L'opérateur CUBE pour les entrepôts de données NoSQL orientés colonnes

Khaled Dehdouh, Fadila Bentayeb, Nadia Kabachi, Omar Boussaid

Laboratoire ERIC, Université de Lyon 2
5 avenue Pierre Mendes-France, 69676 Bron Cedex, France
prénom.nom@univ-lyon2.fr

Résumé. L'émergence de grands volumes de données, imposée par les grands acteurs du web, nécessite de nouveaux modèles de gestion de données et des nouvelles architectures de stockage et de traitement capables de trouver rapidement une information dans une volumétrie considérable de données. Les bases de données NoSQL (Not Only SQL) orientées colonnes offrent pour les *big data*, un modèle approprié aux entrepôts de données et à une structuration multidimensionnelles sous forme de cube OLAP(On-Line Analytical Processing). Cependant, en l'absence d'opérateur de calcul de cube OLAP, nous proposons dans cet article, un nouvel opérateur d'agrégation, baptisé CN-CUBE (*Columnar NoSQL CUBE*), qui permet de calculer des cubes de données à partir d'entrepôts de données stockés dans un système de gestion de base de données NoSQL orientées colonnes. Nous avons implémenté l'opérateur CN-CUBE sous l'interface SQL (*Phoenix* [1]) du SGBD orienté colonnes *Hbase* [2], et réalisé des expérimentations sur un entrepôt de données publiques dans un environnement distribué réalisé avec *Hadoop* [3]. Nous avons pu montrer ainsi que notre opérateur CN-CUBE présente des temps de calcul de cubes OLAP intéressants pour les entrepôts de *big data*.

1 Introduction

Un entrepôt de données est une base de données dédiée à l'analyse en ligne (OLAP) pour l'aide à la décision (Inmon, 1992). Il est souvent implémenté sous un système de gestion de base de données relationnelles (SGBDR) (Codd, 1970). Cependant, dans un monde qui est constamment connecté, les sources de données produisent des données de plus en plus massives, appelées *big data*. Les modèles relationnels classiques ont montré leurs limites quant au stockage et à la gestion des *big data* (Leavitt, 2010). En effet, ce sont les grands acteurs du web tels que Yahoo, Google, Facebook, Twitter et LinkedIn qui ont signalé en premier les limites du modèle relationnel. Le constat était que les SGBDR ne sont pas adaptés aux environnements distribués requis par les volumes gigantesques de données. Pour répondre aux besoins

1. http://phoenix.incubator.apache.org/
2. http://hbase.apache.org/
3. http://hadoop.apache.org/

de stockage, de traitement et d'exploitation des données *big Data*, les systèmes de gestion de bases de données NoSQL sont apparus. Ces derniers ont été construits en abandonnant les contraintes ACID (Atomicité, Cohérence, Isolation et Durabilité) (Cattell, 2011a). Ceci permet de déployer des bases de données dans le nuage et une montée en charge élevée toute en assurant une grande performance.

Nous nous intéressons dans cet article à l'analyse en ligne (OLAP- On-Line Analytical Processing) des *big data*. Les opérateurs OLAP permettent, à partir d'un entrepôt de données, d'extraire des cubes de données correspondant à des contextes d'analyse. Un cube de données est une structure multidimensionnelle qui permet de représenter le fait à observer (mesures) selon plusieurs axes d'observation (dimensions) (Gray et al., 1997). Le calcul de cube permet d'avoir des agrégations au-delà des limites de *Group by*. Cela consiste à calculer tous les agrégats suivant tous les niveaux de toutes les dimensions. Si le cube est composé de trois dimensions A, B et C, les agrégats calculés concernent les combinaisons suivantes : (A, B, C), (A, B, ALL), (A, ALL, C), (ALL, B, C), (A, ALL, ALL), (ALL, B, ALL), (ALL, ALL, C), (ALL, ALL, ALL).

Dans ce contexte, une base de données NoSQL orientée colonnes constitue un modèle de stockage très adapté aux entrepôts de données et à l'analyse en ligne (Jerzy, 2012). En effet, la modélisation en colonnes est naturellement appropriée à la structure des données multidimensionnelles des cubes OLAP. Malheureusement, nous pouvons déplorer le fait qu'actuellement, les SGBD NoSQL ne disposent pas encore d'opérateurs d'agrégation pour construire des cubes OLAP. Pour pallier ce problème, nous proposons dans cet article un opérateur d'agrégation, appelé CN-CUBE (Columnar NoSQL CUBE), pour les SGBD NoSQL orientés colonnes. Cet opérateur permet de calculer des cubes OLAP à partir d'entrepôts implémentés à l'aide de bases de données NoSQL orientées colonnes. CN-CUBE exploite une vue, résultat d'une requête d'extraction, définie sur les attributs (dimensions et mesures) nécessaires au calcul du cube OLAP correspondant à un besoin d'analyse. Cette stratégie permet de réduire les accès au disque en évitant le retour aux données de l'entrepôt pour le calcul des différents agrégats. En outre, pour calculer tous les agrégats du cube OLAP, CN-CUBE exploite les positions des valeurs dans les colonnes, pour calculer les différents agrégats du cube OLAP. Nous avons implémenté l'opérateur CN-CUBE et évalué ses performances sur un entrepôt de données que nous avons créé au sein du SGBD NoSQL orienté colonnes *HBase* avec *Hadoop*. Le choix du SGBD *HBase* et la plateforme *Hadoop* est motivé par leur contexte distribué nécessaire pour le stockage et l'analyse des *big data*. L'entrepôt est alimenté par des données réelles de type open data qui décrivent les accidents corporels de la circulation.

La suite de cet article est organisée de la manière suivante. La section 2 dresse un état de l'art sur les bases de données NoSQL orientées colonnes. La section 3 est consacrée à la description détaillée du processus de calcul de cube OLAP et à la présentation de l'opérateur CN-CUBE. Nous présentons dans la section 4 l'implémentation de notre opérateur CN-CUBE et les expérimentations d'évaluation que nous avons menées. Enfin, nous concluons cet article et présentons quelques perspectives de notre travail dans la section 5.

2 État de l'art

La base de données NoSQL orientée colonnes stocke les données d'une table colonne par colonne. Les données sont modélisées sous forme d'un couple clé/valeur et sont stockées dans un système de fichiers distribués. Chaque valeur est associée à un horodatage (timestamp), attribué par le système, qui sert à gérer la cohérence des données, la combinaison <clé, nom de colonne, timestamp> représente les coordonnées de la valeur. Cependant, la réplication des données dans différentes machines (noeuds) imposée par la gestion de données dans un environnement distribué, engendre parfois, lors des mises à jour, des versions différentes de la même donnée. Grâce à l'horodatage (timestamp) associé à chaque valeur, c'est la version la plus récente qui sera prise en compte lors d'une interrogation de la base de données (Cattell, 2011b).

Par ailleurs, les interfaces (API [4]) natives des SGBD NoSQL orientés colonnes nécessitent l'écriture de plusieurs lignes souvent complexes pour exploiter les données. Pour simplifier la manipulation des données aux utilisateurs, des interfaces (API) SQL ont été développées et adaptées au stockage orienté colonnes. Cela a permis de réduire considérablement la quantité de code à écrire et d'offrir la possibilité d'intégrer des opérateurs d'agrégation. On peut citer *CQL* [5] (Cassandra Query Language) et *Phoenix* [6] respectivement pour *Cassandra* [7] (la base de données NoSQL de la fondation Apache, initialement développée par Facebook) et *HBase* (la base de données utilisée par le projet Hadoop). Ces interfaces SQL fournissent la meilleure façon de combiner les fonctions de manipulation de données pour exprimer une sélection de données en appliquant des filtres (James, 2013).

Cependant, même si tout le monde s'accorde à dire que le stockage en colonnes est bien adapté aux données multidimensionnelles et par conséquent au calcul de cubes de données, les SGBD NoSQL orientés colonnes ne disposent malheureusement pas d'opérateur de calcul de cubes OLAP.

3 L'opérateur CN-CUBE

L'opérateur CN-CUBE que nous proposons permet de calculer le cube OLAP à partir des entrepôts de données NoSQL orientés colonnes suivant trois phases :

Première phase : Elle consiste à définir une vue sur les attributs (dimension et mesures) nécessaires au calcul du cube OLAP. Les données qui satisfont tous les prédicats sont extraites à partir de l'entrepôt de données stocké en colonnes. Seules les valeurs qui ont l'horodatage le plus récent sont prises en compte. Le résultat obtenu est donc, une relation R composée des colonnes qui représentent les axes d'analyse et la (les) colonne(s) représentant la (les) mesure(s) à agréger. Cette relation est un résultat intermédiaire pour constituer l'ensemble des parties du cube OLAP. Cette stratégie permet à CN-CUBE d'éviter le retour aux données de l'entrepôt

4. Application Programming Interface
5. http ://www.datastax.com/docs/1.1/references/cql/index
6. http ://www.orzota.com/sqlforhbase/
7. http ://cassandra.apache.org/

pour le calcul d'agrégats. A ce stade, la relation R permet déjà d'obtenir l'agrégation totale et celle en fonction de toutes les colonnes représentant les dimensions.

Deuxième phase : Dans cette phase, chaque colonne dimension de la relation R est hachée avec les valeurs qui la composent pour obtenir la liste des positions de ces valeurs. Les valeurs de ces listes sont binaires, elles peuvent correspondre à "1" ou à "0" ; le "1" indique que la valeur de hachage existe à cette position et "0" sinon. Ces listes permettent d'avoir les agrégats de chaque dimension séparément.

Troisième phase : Au niveau de cette phase, les listes de positions des dimensions au niveau de R sont associées via l'opérateur "ET logique". Cette opération permet d'identifier les valeurs des dimensions à combiner et les valeurs de la mesure à agréger qui correspondent aux différentes combinaisons. Le regroupement des sous résultats (totaux et sous totaux) des trois phases permet le calcul du cube.

3.1 Processus d'exécution de l'opérateur CN-CUBE dans un environnement distribué

Pour exploiter les *big data*, les entrepôts de données ont besoin d'opter pour une solution forcément distribuée pour pouvoir passer à l'échelle. Cette solution repose sur une architecture de stockage de données répartie sur plusieurs machines et un traitement de données parallélisé. *Hadoop* offre un système de gestion de fichiers distribué, appelé *HDFS* (*Hadoop Distributed File System*), conçu pour stocker de très gros volumes de données sur plusieurs machines (Shvachko et al., 2010) et un système de traitement de données distribué, appelé *MapReduce* (Dean et Ghemawat, 2008). Ce dernier est un modèle de traitement massivement parallèle adapté au traitement de très grandes quantités de données. Il s'articule sur deux étapes principales, *Map* et *Reduce*. La fonction *Map* découpe le traitement en sous traitements sur les différents noeuds composant le cluster et les résultats sont regroupés via la fonction *Reduce*. L'intégration d'un système de gestion de base de données NoSQL tel que *HBase* avec *Hadoop*, permet de mieux structurer et indexer les données pour éviter de parcourir, lors de l'accès aux données, tout le cluster (full scan).

Cependant, sur les trois phases qui caractérisent l'exécution de l'opérateur CN-CUBE, le processus d'exécution de l'opérateur CN-CUBE sur un cluster à plusieurs noeuds s'effectue en déclenchant deux principaux *jobs MapReduce*. Le premier exécute la première phase qui consiste à construire le résultat intermédiaire nécessaire aux calculs de l'ensemble des parties du cube OLAP et fournir par conséquent les agrégats pouvant être calculés à ce niveau. Le résultat du premier *job*, constitue l'entrée du deuxième *job* qui se charge de calculer à partir du résultat intermédiaire, le reste des agrégats composant le cube (phases 2 et 3). Ainsi, les phases 2 et 3 ne peuvent avoir lieu qu'après achèvement de la phase 1. Le regroupement des résultats des jobs, constitue le cube OLAP.

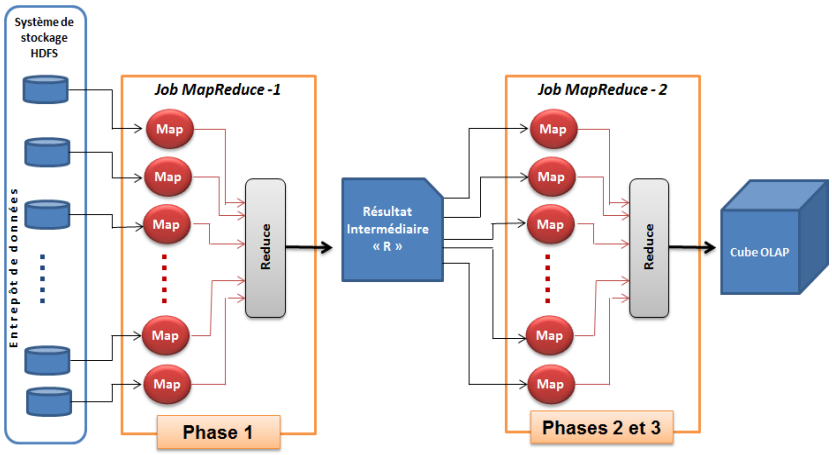

FIG. 1 – *Processus d'exécution de l'opérateur CN-CUBE avec MapReduce*

4 Implémentation et expérimentations

Pour implémenter l'opérateur CN-CUBE dans un environnement NoSQL orienté colonnes avec *MapReduce*, nous avons mis en place un environnement de stockage et de traitement non relationnel et distribué (Taylor, 2010). Cet environnement est basé sur une architecture de Cloud Computing privé réalisée via la plateforme *Hadoop-1.2.0* et un SGBD NoSQL *HBase-0.94.14*, dédié à la gestion des données dans un environnement distribué. Pour simplifier la manipulation des données et augmenter la performance du SGBD *HBase*, nous avons renforcé cette configuration avec une interface SQL dédiée à *HBase*, appelée *Phoenix-2.2.2*, ce dernier est en source libre, il permet de combiner les fonctions de manipulation de données au niveau de *HBase* (*scan, put et get*) pour exprimer une sélection de données et appliquer des filtres (prédicats) (James, 2013).

Pour calculer des cubes OLAP, nous avons implémenté l'opérateur CN-CUBE sous forme d'application cliente basée sur une fonction de calcul de cube (UDF : user define function), développée en JAVA 1.7 (Java SE Runtime Environment) sous Netbeans IDE 7.4. Celle-ci se connecte au serveur de données *HBase* via l'API *Phoenix*.

4.1 Environnement de tests

L'environnement de tests est un cluster composé de 15 machines (noeuds). Chaque machine est équipée d'un processeur intel-Core TMi3-3220 CPU@3.30 GHZ avec une mémoire *RAM* de $4Go$. Ces machines fonctionnent avec le système d'exploitation *Ubuntu-12.10* et elles sont interconnectées via un réseau local ethernet de 100 Mbps. Parmi ces 15 machines, une d'entre elles est configurée pour jouer le rôle de *Namenode* du système *HDFS*, le *master* et le *Zookeper* de *HBase* (Hunt et al., 2010). Les autres machines sont configurées pour être des *DataNode* du système *HDFS* et les *RegionServers* de *HBase*. Bien que l'architecture de Cloud privé que nous avons adoptée soit limitée en termes de capacités, elle est suffisante pour nous

permettre de déployer un entrepôt de données non-relationnel avec un passage à l'échelle et d'implémenter l'opérateur CN-CUBE dans un environnement distribué.

Pour évaluer l'opérateur CN-CUBE, nous avons intégré à cet environnement un logiciel dédié à la gestion des données massives, appelé *HIVE*[8]. Ce dernier dispose d'une interface SQL, appelée *HQL* (*Hive Query Language*) pour exploiter ses données (Thusoo et al., 209). *Hive* n'est pas orienté colonnes, cependant, une fois intégré à *HBase*, il offre la possibilité de manipuler en *HQL* les données stockées dans *HBase* (Apache Hive, 2012). Nous avons choisi d'utiliser *Hive* pour comparer les performances de l'opérateur CN-CUBE, car il intègre dans sa version 0.10.0, l'opérateur CUBE pour le calcul de cube OLAP (Apache Hive, 2014). Pour réaliser nos expérimentations, nous avons utilisé le jeu de données présenté à la section 4.2.

4.2 Jeu de données

Afin de valider notre approche de calcul de cube OLAP via l'opérateur CN-CUBE, nous avons choisi d'utiliser un jeu de données réelles, de type open data, relatif aux accidents corporels de la circulation. Ce jeu de données est extrait du fichier national des accidents corporels de la circulation, administré par l'observatoire national interministériel de la sécurité routière[9](Ministère de l'Intérieur, 2013). Il regroupe des informations essentielles recueillies dans un bulletin d'analyse d'accident corporel de la circulation par les forces de l'ordre, à la suite de chaque accident corporel de la circulation en France métropolitaine et impliquant au moins un véhicule et ayant fait au moins une victime ayant nécessité des soins. Les informations collectées dans ce jeu de données ont été rapportées par les différentes unités des forces de l'ordre (police, gendarmerie, pompiers, etc.) qui sont intervenues sur le lieu de l'accident.

Charge de requêtes : Pour évaluer l'opérateur CN-CUBE que nous proposons pour le calcul de cube OLAP dans l'entrepôt de données orienté colonnes, nous avons défini quatre requêtes de calcul des cubes OLAP. Celles-ci impliquent graduellement le nombre de dimensions dans le calcul des cubes. Ces requêtes agrègent le nombre d'accidents de la route (mesure) ayant un indice de gravité supérieur ou égal à 100 (au moins un mort) et qui sont survenus dans le département du RHONE (69). Les agrégats sont calculés en fonction de différents axes d'analyse (dimensions), comme suit :

La première requête agrège le nombre d'accidents et calcule les différents agrégats par rapport à l'organisme ayant réalisé le constat (ORG) et les circonstances d'éclairage (LUM) lors de l'accident. La seconde requête calcule les mêmes agrégats avec une dimension supplémentaire à savoir, les conditions atmosphériques (ATM). La troisième requête rajoute la dimension type de collusion (COL) dans les calculs de ces agrégats. La quatrième requête s'intéresse en plus de toutes les dimensions précédentes, à la catégorie des routes (CATR).

8. http ://hive.apache.org
9. http ://www.data.gouv.fr/

4.3 Expérimentations

Nous avons évalué les performances de l'opérateur CN-CUBE en matière de temps de calcul de cube OLAP. Pour cela, nous avons réalisé deux expérimentations. La première est consacrée à évaluer l'opérateur CN-CUBE en augmentant graduellement le nombre de dimensions lors du calcul de cube OLAP. La deuxième évalue le temps de calcul de cube face au passage à l'échelle.

4.3.1 Calcul du cube OLAP sur un cluster à un seul noeud

L'objectif de cette expérimentation est d'évaluer empiriquement l'opérateur CN-CUBE face aux variations du nombre de dimensions. Pour cela, nous avons comparé le temps de calcul des cubes OLAP de deux à cinq dimensions sur un échantillon de données composé de 60 millions de n-uplets, et nous avons exécuté les quatre requêtes décisionnelles de la section 4.2. Rappelons que ces requêtes génèrent des cubes OLAP avec un nombre de dimensions qui augmente progressivement. Les résultats obtenus sont présentés dans la figure 2.

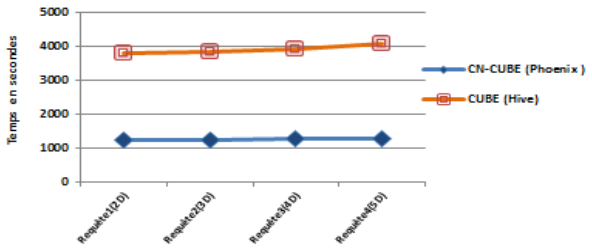

FIG. 2 – *Comparaison du temps de calcul de cubes OLAP à 2, 3, 4 et 5 dimensions entre CN-CUBE et CUBE.*

Nous constatons une légère variation du temps de calcul de cubes OLAP avec les opérateurs CN-CUBE et CUBE, face à l'augmentation du nombre de dimensions. Cependant, CN-CUBE affiche une meilleure performance que CUBE. En effet, l'opérateur CUBE enregistre des temps entre 3801 et 4064 secondes, alors que l'opérateur CN-CUBE enregistre des temps entre 1240 et 1279 secondes. L'inconvénient de l'opérateur CUBE de *Hive* réside dans l'exploitation des données. En effet, cette dernière, via l'interface *HQL*, nécessite la migration des données de *HBase* vers *Hive*. Cette migration consiste à exporter les données d'une table *HBase* vers *Hive*. Cet export nécessite un parcours de la table, ce qui engendre un temps supplémentaire qui se traduit par un temps d'exécution plus important. En revanche, la performance de l'opérateur CN-CUBE réside dans l'utilisation et l'exploitation des listes de positions pour le calcul des cubes OLAP. Ces listes occupent peu d'espace mémoire. Elles conviennent parfaitement à un traitement au niveau de la mémoire (*MEMETABLE*) sans pour autant solliciter de multiples accès au disque (*FileStore*). En effet, l'augmentation du nombre de dimensions dans le calcul de cubes de données se traduit techniquement par la création et la manipulation des vecteurs de bits qui représentent des listes des positions des valeurs. Ce procédé n'impacte pas lourdement la mémoire. De plus, les combinaisons sont réalisées avec des listes (vecteur de bits

composé de "1" ou "0") et non pas avec des valeurs, ces dernières ne sont extraites qu'après avoir construit la liste des positions y afférente à une combinaison.

Eu égard aux résultats obtenus, nous avons constaté que l'opérateur CN-CUBE optimise jusqu'à trois fois le temps de calcul de cube OLAP comparé à l'opérateur CUBE de *Hive*. Pour cela, il était très intéressant d'évaluer le comportement de CN-CUBE face à un passage à l'échelle dans un environnement distribué.

4.3.2 Calcul du cube OLAP sur un cluster avec plusieurs noeuds (passage à l'échelle)

L'objectif de cette expérimentation est de confronter l'opérateur CN-CUBE au passage à l'échelle. Ce dernier représente la clé de voute de l'avènement des bases de données NoSQL et son adoption par les grands acteurs du web. L'expérimentation consiste à évaluer le temps d'exécution de la requête 2 de la section 4.2 qui calcule un cube OLAP à trois dimensions, sur quatre configurations différentes et avec des échantillons de données de taille différentes (100 Go, 500 Go et 1 To). Les quatre configurations consistent en un cluster à un seul noeud, un cluster à cinq noeuds, un cluster à dix noeuds et un cluster à quinze noeuds. Les résultats que nous avons obtenus sont présentés dans la figure 3.

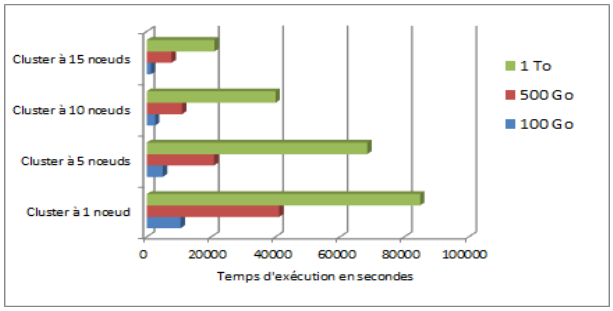

FIG. 3 – *Résultat de la monté en charge sur un cluster à plusieurs noeuds*

Nous constatons que le temps de calcul de cube OLAP des différentes tailles de l'entrepôt diminue en augmentant le nombre de noeuds dans la composition de cluster. C'est le cluster à 15 noeuds qui enregistre les meilleurs temps de calcul. L'écart entre les résultats des temps de calcul du cube OLAP entre les différents clusters apparait plus clair pour les entrepôts de données volumineux (500 Go et 1 To) et les avantages d'ajouter des noeuds au cluster sont appréciables au fur et à mesure que la taille de l'entrepôt augmente. Pour une taille d'entrepôt de 100 Go, l'augmentation du nombre de noeuds réduit le temps de calcul de cube OLAP jusqu'à 9188 secondes (2h et 55mn). En revanche, pour une taille de 1 To de l'entrepôt, l'augmentation du nombre de noeuds réduit le temps de calcul jusqu'à 63900 secondes (17h et 45mn).

En effet, le SGBD *HBase* adopte nativement un système de stockage et de traitement des données qui repose respectivement sur le système de fichier *HDFS* et le paradigme *MapReduce*. La charge de travail en termes de mémoire et de calcul (CPU) ainsi que le stockage est distribuée sur toutes les machines du cluster. Cela présente deux avantages, d'une part, le

stockage des données de l'entrepôt est distribué physiquement sur les machines (noeuds) composant le cluster. Cela évite de parcourir toutes la table (full scan) pour extraire des données. Et d'autre part, la requête de calcul du cube OLAP est répartie en sous traitements sur l'ensemble des machines du cluster donnant lieu à des sous-résultats. Ces derniers sont regroupés, constituant ainsi le résultat final de la requête qui correspond dans ce cas au cube OLAP.

De ce constat, il est clair que l'opérateur CN-CUBE tire profit de l'apport de l'environnement distribué des bases de données NoSQL pour calculer des cubes OLAP.

5 Conclusion

L'intérêt majeur de ce travail est d'étendre l'utilisation des bases de données NoSQL orientées colonnes au domaine décisionnel. Pour cela, nous avons proposé dans cet article, CN-CUBE, un opérateur de calcul du cube OLAP. L'avantage de cet opérateur est qu'il exploite les positions de valeurs pour le calcul des agrégats du cube OLAP. Cette manière de procéder réduit considérablement les flux d'entrée et sortie des données. L'implémentation de cet opérateur au sein de l'interface SQL (*Phoenix*) du SGBD NoSQL orienté colonnes *HBase* et les expérimentations que nous avons menées sur l'entrepôt de données réelles de type open data ont montré clairement la performance de l'opérateur CN-CUBE dans un environnement distribué comparée à celui de l'opérateur CUBE de *Hive*. Ce travail ouvre plusieurs perspectives intéressantes. L'une des pistes de recherche consiste à créer d'autres opérateurs OLAP relatifs à l'exploitation du cube.

Références

Apache Hive (2012). *https ://cwiki.apache.org/confluence/display/Hive/HBaseIntegration*.

Apache Hive (2014). *https ://cwiki.apache.org/confluence/display/Hive/LanguageManual*.

Cattell, R. (2011a). Scalable SQL and NoSQL Data Stores. *Association for Computing Machinery ACM SIGMOD Record 39*, 12–27.

Cattell, R. (2011b). Zookeeper: Wait-free Coordination for Internet-scale Systems. *Association for Computing Machinery ACM SIGMOD Record 39*, 12–27.

Codd, E. (1970). A Relational Model of Data for Large Shared Data Banks. *Association for Computing Machinery ACM 1*, 377–387.

Dean, J. et S. Ghemawat (2008). MapReduce: Simplified Data Processing on Large Clusters. *Association for Computing Machinery ACM Commun 51*, 107–113.

Gray, J., S. Chaudhuri, A. Bosworth, A. Layman, D. Reichart, M. Venkatrao, F. Pellow, et H. Pirahesh (1997). Data cube : A relational aggregation operator generalizing group-by, cross-tab, and sub totals. *Journal of Data Mining and Knowledge Discovery 1*, 29–53.

Hunt, P., M. Konar, F. P. Junqueira, et B. Reed (2010). Zookeeper: Wait-free Coordination for Internet-scale Systems. *Proceedings of the 2010 USENIX Conference on USENIX Annual Technical Conference*, 11–24.

Inmon, W. (1992). *Building the Data Warehouse*. Information Sciences, Inc Wellesley, MA, USA.

James, T. (2013). *https://github.com/forcedotcom/phoenix/wiki/Performance*.

Jerzy, D. (2012). Business Intelligence and NoSQL Databases. *Information Systems in Management 1*, 25–37.

Leavitt, N. (2010). Will NoSQL databases live up to their promise ? *IEEE Computer Society 43*, 12 – 14.

Ministère de l'Intérieur (2013). Plateforme ouverte des données publiques françaises. *http ://www.data.gouv.fr/fr/dataset/base-de-donnees-accidents-corporels-de-la-circulation-sur-6-annees*.

Shvachko, K., H. Kuang, S. Radia, et R. Chansler (2010). The Hadoop Distributed File System. *IEEE Computer Society*, 1–10.

Taylor, R. (2010). An overview of the Hadoop-MapReduce-Hbase framework and its current applications in bioinformatics. *BMC Bioinformatics Journal 11*, S1.

Thusoo, A., J. Sarma, N. Jain, Z. Shao, P. Chakka, S. Anthony, H. Liu, P. Wyckoff, et R. Murthy (209). Hive: A Warehousing Solution over a Map-reduce Framework. *International Conference on Very Large Databases Endowment 2*, 1626–1629.

Summary

The emergence of great volumes of data, imposed by the large actors on the Web, requires new models of data management and new architectures of storage and treatment able to quickly find information in a considerable volumetry. NoSQL columnar databases are suitable for data warehouses and multidimensional data structures storage and offer for big data the appropriate model. However, Columnar DBMS have not an appropriate operator for calculating OLAP cubes. In this paper, we propose a new OLAP operator for columnar DBMS, CN-CUBE (Columnar NoSQL CUBE), that allows to calculate OLAP data cubes from NoSQL columnar oriented-data warehouses. We have then implemented CN-CUBE under *HBase* DBMS and curried out some experimentations on open data.