Intégration des Tableaux Multidimensionnels en Pig pour l'Entreposage de Données sur les Nuages

Laurent d'Orazio*, Sandro Bimonte**

*Université Blaise Pascal - LIMOS
Campus des Cézeaux
63173 Aubière Cedex, France
dorazio@isima.fr
http://www.isima.fr/ dorazio/
**Cemagref
Campus des Cézeaux
63173 Aubière Cedex, France
sandro.bimonte@cemagref.fr
http://eric.univ-lyon2.fr/ sbimonte/

Résumé. Les entrepôts de données et les systèmes OLAP correspondent à des technologies d'aide à la décision. Ils permettent d'analyser à la volée de gros volumes de données représentés en fonction d'un modèle multidimensionnel. L'informatique dans les nuages, sous l'impulsion des grandes compagnies telles que Google, Microsoft ou encore Amazon, a récemment suscité une attention particulière. Considérer l'interrogation OLAP et les entrepôts de données au sein de telles infrastructures devient alors un enjeu majeur. Les problèmes devant être considérés sont ceux classiques des systèmes largement distribués (interrogation de gros volumes de données, hétérogénéité sémantique et structurelle ou encore variabilité), mais d'un nouveau point de vue devant considérer les spécificités de ces architectures (facturation à l'utilisation, élasticité et facilité d'utilisation). Dans ce papier nous abordons dans un premier temps les règles de facturation à l'utilisation pour le stockage des entrepôts de données. Nous proposons d'utiliser des techniques de stockage pour nuages à base de tableaux multidimensionnels. De premières expérimentations montrent l'intérêt de notre proposition. Ensuite, nous listons des perspectives de recherche.

1 Introduction

Les entrepôts de données et les systèmes OLAP (On-Line Analytical Processing) représentent des technologies d'aide à la décision qui permettent l'analyse en ligne de gros volumes de données [17]. Dans une architecture Relational OLAP (ROLAP), les systèmes OLAP sont déployés en utilisant des Systèmes de Gestion de Bases de Données relationnels pour stocker et analyser les données. Cette approche est pertinente pour les entrepôts de données peu denses. Quand les données sont denses, l'approche Multidimensional OLAP (MOLAP) peut être utilisée [34]. L'architecture MOLAP stocke les données à l'aide de structures multidimensionnelles

ad-hoc, tels que des tableaux multidimensionnels, afin de réduire la taille des données stockées. Enfin, il existe une autre typologie qui combine les deux technologies : Hybrid OLAP (HOLAP). Selon cette approche, une partie des données sont stockées en relationnel et lŠautre en utilisant des techniques multidimensionnelles.

Les Architectures Hautes Performances visent à faire face aux besoins croissants en terme de calcul et de stockage des applications scientifiques et industrielles [8]. Parmi ces architectures, les nuages informatiques (*cloud computing*) sous l'impulsion d'entreprises telles que Google, Microsoft et Amazon suscitent actuellement une attention particulière.

Les entrepôts de données et les systèmes OLAP dans les nuages soulèvent différents problèmes liés à la performance du stockage et de l'interrogation. Les problèmes à aborder sont ceux classiques des systèmes distribués à grande échelle (interrogation de gros volumes de données, hétérogénéité sémantique et structurelle, variabilité, etc.), notamment abordés par les travaux autour des entrepôts de données sur grilles [24, 15, 21], mais qui doivent être considérés d'un nouveau point de vue qui prend en compte les caractéristiques particulières de ces architectures : facturation à l'utilisation, élasticité et facilité d'utilisation [8].

Certains travaux récents abordent les requêtes complexes, telles que les requêtes spatiales ou OLAP, dans les nuages [23, 31, 33]. Néanmoins, à notre connaissance, aucun travail ne définit un modèle de données pour le stockage de données multidimensionnelles dans les nuages considérant la facturation à la demande, les approches existantes étant basés sur de simples fichiers textes.

Ce papier présente une première avancée vers l'implémentation d'une architecture MOLAP dans les nuages afin de réduire les coûts de stockage des données. En particulier, nous présentons un algorithme qui transforme les données stockées sous forme de tableaux multidimensionnels suivant le modèle de données Pig [23]. Ceci permet d'exécuter les requêtes OLAP en utilisant le paradigme MapReduce [11] et de réduire de manière significative les coûts de stockage. Cet article liste ensuite des perspectives de recherche pour l'analyse de données sur nuages informatiques.

Ce papier est organisé de la manière suivante. La section 2 présente le contexte de ce travail. La section 3 introduit la proposition de stockage de tableaux multidimensionnels pour nuages. La section 4 présente les expérimentations qui valident notre approche. La section 5 liste des pistes de recherche. Enfin, la section 6 conclut ce papier.

2 Contexte et motivation

Cette section présente brièvement les entrepôts de données et OLAP en sous-section 2.1, la gestion de données dans les nuages en sous-section 2.2, puis introduit les motivations de notre travail en sous-section 2.3

2.1 Entrepôts de données et OLAP

Les entrepôts de données représentent les données selon le modèle multidimensionnel. Celui-ci définit les concepts de dimension et mesure. Une dimension est composée de hiérarchies et représente l'axe d'analyse. Une hiérarchie organise les données (membres) dans une structure hiérarchique permettant aux décisionnaires d'analyser les mesures à différentes granularités.

Les mesures correspondent à des indicateurs numériques décrivant le sujet d'analyse (fait). Les opérateurs OLAP tels que roll-up et drill-down permettent de naviguer au sein des hiérarchies synthétisant les données à l'aide des fonctions d'agrégation SQL [17]. D'autres opérateurs ont été définis pour la sélection d'une partie de l'entrepôt de données et la permutation des dimensions [27].

Les architectures MOLAP utilisent des structures de données multidimensionnelles telles que les tableaux multidimensionnels construits à partir des données originales, qui sont en général stockées dans des bases de données relationnelles. Les architectures MOLAP permettent d'optimiser le stockage pour les entrepôts de données denses à l'aide de ce modèle de stockage de données particulier [34].

Les tableaux multidimensionnels permettent de stocker uniquement les mesures car elles sont indexées à l'aide des positions des membres des dimensions. Par exemple, la valeur de la mesure à la position tab[2] [3] [4] est associée au second membre de la première dimension, au troisième membre de la deuxième dimension et au quatrième membre de la troisième dimension. Il est important de noter qu'il est possible de passer d'un tableau multidimensionnel à un tableau uni-dimensionnel avec la formule suivante :

Soient d dimensions, N_k les membres de la k^eme dimension, alors la position de la valeur mesurée dans un tableau à une dimension est :

$p(i_1, ..., i_d) = \sum_{j=1}^{d} (i_j x \prod k = j + 1^d N_k)$

2.2 Gestion de données dans les nuages

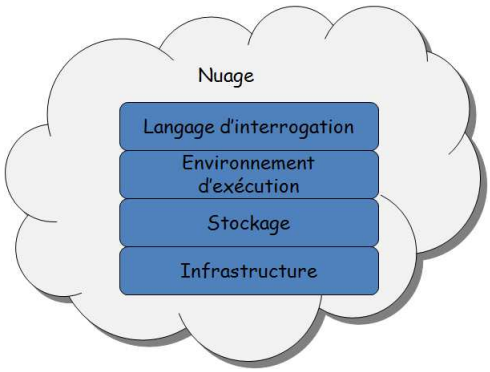

FIG. 1 – *Architecture de la gestion de données sur nuages*

La gestion de données sur nuages se base généralement sur une architecture en couches, comme illustré par la figure 1.

Le premier niveau correspond à l'infrastructure. En général, ce niveau est composé d'un ou plusieurs centres de données, utilisé(s) pour l'analyse de gros volumes de données et mis à disposition par les fournisseurs de nuages. Microsoft Azure [5] et Amazon EC2 [1] sont des exemples de telles infrastructures. La principale caractéristique de ce niveau est son association au modèle de facturation à l'utilisation, où les utilisateurs ne payent que pour les ressources utilisées.

Le deuxième niveau est dédié au stockage des données. Son objectif principal est de proposer un système qui permet le passage à l'échelle et qui soit tolérant aux fautes. Google File System [14], Amazon Simple Storage Service (S3) [2] et Dynamo [12] sont des exemples typiques de ce niveau d'architecture. Au sein des nuages, les données sont stockées sous forme de fichiers gérés par de tels systèmes.

Le troisième niveau représente l'environnement d'exécution. L'exemple probablement le plus connu dans l'informatique dans les nuages est MapReduce [11] de Google et son implémentation de sources libres Hadoop [4], Microsoft proposant un outil similaire appelé Dryad [18]. L'objectif de ce niveau est de fournir la propriété d'élasticité permettant d'ajuster les ressources en fonction des applications, augmentant les capacités de calcul et de stockage en cas de pics d'utilisation, les diminuant lors de périodes creuses. Cette propriété évite d'une part des investissements lourds permettant à des applications de résister à des pics d'utilisation, mais pouvant conduire à une sous-utilisation globale de l'infrastructure. Elle permet d'assurer d'autre part le bon fonctionnement d'une application dont le passage à l'échelle n'aurait pas été prévu correctement, en augmentant les ressources si nécessaire.

La dernière couche offre un langage d'interrogation à haut niveau. Une telle couche a pour objectif de proposer une certaine facilité d'utilisation et une totale transparence des autres couches de l'architecture, tout en autorisant le parallélisme. Différents langages d'interrogation ont été proposés, tels que Hive de Facebook [31], Scope de Microsoft [9], Sawzall [26] de Google ou encore Map-Reduce-Merge [32], qui sont basés sur des modèles de données particuliers comme le modèle orienté colonne [29] ou des modèles étendant le modèle relationnel [23]. Le modèle de données Pig et son langage Pig Latin en particulier ont été proposés pour offrir un compromis entre un langage déclaratif de type SQL et le style procédural bas niveau de MapReduce. Un système basé sur Pig a été implémenté, permettant la compilation d'instructions Pig Latin et leur exécution dans des environnements parallèles.

2.3 Motivation

Conformément au principe de facturation à l'utilisation, les utilisateurs ne paient que pour les ressources (cycle processeur consommation de bande passante et stockage) qu'ils utilisent. Par exemple, sur Microsoft Azure [5] le coût CPU pour une heure d'exécution est 0,12 $, le stockage revient à 0,15 $ par Go et par mois, alors que la consommation de bande passante est facturée par Go 0,10 $ en chargement et 0,15 $ en téléchargement.

D'une part des langages d'interrogation de données sur nuages ne permettent qu'indirectement la formulation de requêtes OLAP (puisqu'aucun opérateur [16] ad-hoc n'a été proposé) et d'autre part aucun modèle de données associé ne fournit un stockage ad-hoc pour les données multidimensionnelles.

C'est pourquoi notre travail vise à proposer une organisation spécifique des données multidimensionnelles pour les nuages informatiques afin de réduire les coûts de stockage et de calcul pour les requêtes OLAP en accord avec la contrainte des nuages informatiques de facturation à l'utilisation.

De plus, l'analyse en ligne dans les nuages est un domaine de recherche vraiment nouveau. En effet, si d'un côté l'Université de Californie [8] a identifié des axes de recherches pour nuages (disponibilité et qualité de service, sécurité, etc.), de l'autre côté, personne à notre connaissance n'a exploré les différentes pistes de recherche pour l'OLAP dans les nuages, si l'on excepte quelques produits commerciaux [3] [6] ayant fait leur apparition rapidement et

Year	Month	Day	Country	Region	Department	Profit
2010	04	01	France	Auvergne	Puy-de-Dôme	2000
2010	04	01	France	Auvergne	Allier	500
2010	04	01	France	Auvergne	Haute-Loire	400
...	...	...	...	...		
2005	11	29	France	Rhône-Alpes	Isère	2500

TAB. 1 – *Exemple de données multidimensionnelles représentées à l'aide du modèle Pig*

```
s9091 = FILTER sales BY years = 1990 or years = 1991;
groups = GROUP s9091 BY year, region;
results = FOREACH groups
GENERATE s9091.region, s9091.month, SUM(profit);
```

FIG. 2 – *Requête OLAP avec Pig : FILTER et (GROUP, Aggrégation)*

récemment, ne fournissant que peu d'informations à leur sujet. Dans ce papier nous détaillerons des perspectives portant sur les performances et la modélisation multidimensionnelle sur les nuages.

3 Tableaux multidimensionnels dans les nuages

Dans cette section, nous introduisons un cas d'étude et le stockage à l'aide de tableaux multidimensionnels (sous-section 3.1). Puis nous présentons un algorithme pour la transformation de tableaux multidimensionnels en données Pig et notre proposition d'optimisation des requêtes OLAP en Pig (sous-section 3.2).

3.1 Cas d'étude

Afin de présenter notre travail, nous nous basons sur un cas d'étude simulé portant sur l'analyse OLAP des ventes de magasins d'une chaîne d'approvisionnement, situés dans chaque département français. Dans ce contexte, deux dimensions sont considérées, une dimension spatiale regroupant les départements en régions et une dimension temporelle avec une hiérarchie jour, mois et année, tandis que la mesure est le profit. Le tableau 1 propose un exemple de données, représentées à l'aide du modèle Pig. Dans ce papier pour simplifier la présentation de notre contribution, nous considérons une requête OLAP comme une requête qui porte sur un seul niveau d'agrégation. En effet, une requête OLAP classique [16] est composée par un ensemble de requêtes sur différents niveaux d'agrégation unis par une "union", opérateur qui est également implémenté en Pig Latin.

Une requête OLAP classique sur ces données est : "quel est le profit total pour chaque région entre 1990 et 1991 ?". Une telle requête peut facilement être exprimée à l'aide d'instructions Pig sur le modèle de données présenté par le tableau 1, comme l'illustre la figure 2.

```
Dimensions
  Temps
    Time_Dim[0]=2010-04-01
    Time_Dim[1]=2010-03-31
    Time_Dim[2]=2010-03-30
    Time_Dim[3]=2010-03-29
    ...
    Time_Dim[1826]=2005-11-29
  Localisation
    Location_Dim[0]=France,Auvergne,Puy-de-Dôme
    Location_Dim[1]=France,Auvergne,Allier
    Location_Dim[2]=France,Auvergne,Haute-Loire
    Location_Dim[3]=France,Auvergne,Cantal
    ...
    Location_Dim[99]=France,Rhône-Alpes,Isère

Mesures
  Facts Profit_Fact[0]=2000
  Facts Profit_Fact[1]=500
  Facts Profit_Fact[2]=400
  ...
```

FIG. 3 – *Exemple de tableaux multidimensionnels*

Le modèle de données Pig est une extension du modèle relationnel avec les concepts atomiques de *Bag* (ensemble de valeurs), de *Map* (fonctions de hachage), des tableaux imbriqués et des *UDF* (User Defined Functions ou fonctions définies par l'utilisateur). Ainsi, une requête OLAP en Pig peut facilement être définie en utilisant trois instructions. La première sélectionne les membres des dimensions (1990 et 1991 sur notre exemple). Une autre permet de regrouper les données (regroupement par année et région pour cet exemple), alors qu'une dernière réalise la tâche d'agrégation (une somme dans notre cas particulier). Il est à noter que contrairement à celles écrites en SQL pour lesquelles un optimiseur est utilisé afin de choisir un plan d'exécution à partir d'éléments d'optimisation, les requêtes en langage Pig Latin correspondent à un ensemble d'instructions dont l'ordre est laissé à la responsabilité de l'utilisateur, ce qui peut conditionner le temps de calcul et par conséquent le prix à payer.

La figure 3 présente les données de la table 1 stockées sous la forme de tableaux multidimensionnels. Un tableau à une dimension est fourni pour chaque dimension et pour les mesures. Les mesures sont alors associées aux membres des dimensions en utilisant leur indice et la formule présentée précédemment en section 2.1. Par exemple, Profit_Fact[0] est la mesure associée aux membres 01/04/2010 (Time_Dim[0]) et France, Auvergne, Puy-de-Dôme (Location_Dim[0]).

3.2 Pig et tableaux muldimensionnels pour requêtes OLAP

Cette section présente comment les tableaux multidimensionnels peuvent être utilisés comme structure de stockage pour Pig et l'optimisation de requêtes OLAP écrites en langage Pig Latin.

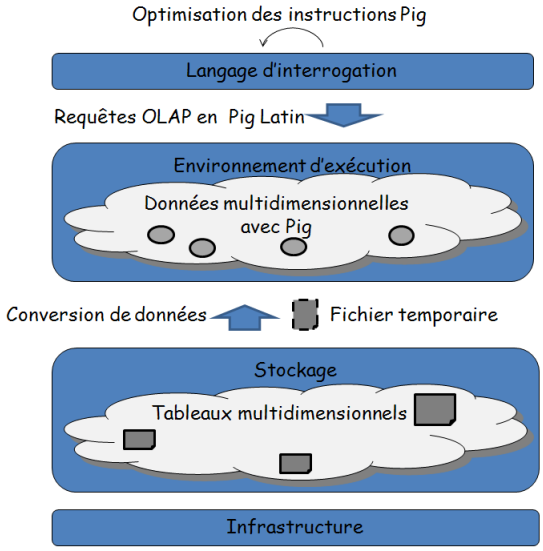

FIG. 4 – *Aperçu du processus d'interrogation*

Le processus d'interrogation, illustré par la figure 4 peut être décomposé en deux étapes :

1. Les données stockées dans des fichiers textes sous forme de tableaux multidimensionnels sont traduites en données Pig via un fichier temporaire. Ceci permet de réduire la taille des fichiers stockés et, par conséquent, le prix à payer par les clients. De plus, ces données peuvent ensuite être manipulées dans un environnement parallèle, à l'aide du paradigme MapReduce, respectant ainsi la propriété d'élasticité.
2. Les requêtes OLAP formulées en Pig Latin sont optimisées.

La conversion des tableaux multidimensionnels en données Pig est faite via l'algorithme 1 Les entrées de cet algorithme sont les fichiers stockant les tableaux des dimensions et les mesures (voir figure 3). La sortie correspond à un fichier représentant l'entrepôt de données basé sur le modèle de donnée Pig (cf. table 1). Le principe de cet algorithme est de construire le produit cartésien de $n-1$ dimensions. Un produit cartésien est alors réalisé entre ces données, la dimension n et les valeurs mesurées sont ajoutées pour générer les n-uplets de la manière suivante : le i^{eme} n-uplet avec la i^{eme} valeur du tableau.

Nous proposons une optimisation simple mais efficace des requêtes OLAP en réécrivant les requêtes Pig. Les requêtes exprimées dans le langage Pig Latin correspondent à un ensemble d'instructions. Le paradigme MapReduce ne fournit pas de moteur d'exécution de plan de requête. Ainsi, les requêtes OLAP peuvent être formulées de deux manières différentes : (i) FILTER et (GROUP et Agrégation) (comme illustré par la figure 2), (ii) (GROUP et Agrégation) et FILTER (comme illustré par la figure 5).

Algorithme 1 Conversion de tableaux multidimensionnels en données Pig

ENTRÉES : Fichiers de tableaux
SORTIES : Fichier Pig

```
int i ← 1 ;
int n ;
file cartProdFile ; {initialisé par le produit cartésien des deux dernières dimensions}
file pigFile ;
file mAFile ;
array dimensions ; {ensemble des dimensions}
tantque i<=n-1 faire
   cartProdFile ← produitCartesien(dimensions[i],cartProdFile) ;
   i ← i + 1 ;
fintantque
i ← 0 ;
tantque non fin de mAFile faire
   rolapFile.insert ← (produitcartesien(dimension(N),cartProdFile )+','+mAFile(i)) ;
   i ← i + 1 ;
fintantque
retourner pigFile ;
```

```
groups = GROUP sales BY years ;
avgprof = FOREACH groups GENERATE region, SUM(profits) ;
results = FILTER avgprof BY years = 1990 or years = 1991 ;
```

FIG. 5 – *Requête OLAP avec Pig : (GROUP, Aggrégation) et FILTER*

Evidemment, une telle séquence influe grandement sur les temps de réponse. Sur un tel exemple, si la taille de la source de données est très grande, agréger tous les résultats et ensuite sélectionner les membres des dimensions peut être très coûteux. C'est pourquoi, une optimisation intuitive des requêtes OLAP en Pig Latin consiste à utiliser le patron d'interrogation "FILTER, GROUPE, Agrégation".

4 Validation

Notre proposition a été validée à l'aide de données simulées. Toutes les expérimentations ont été réalisées sur un ordinateur possédant un processeur Intel Core 2 duo à 2,2 GHz disposant de 4Go de RAM. L'objectif principal de ces expériences était d'illustrer la réduction des coûts de stockage obtenue à l'aide de notre approche, sans détériorer les performances, en particulier en terme de temps de réponse. La sous-section 4.1 s'intéresse ainsi au stockage, la sous-section 4.2 à la conversion de données, alors que la sous-section 4.3 étudie l'impact de notre optimisation sur le temps de réponse.

4.1 Espace de stockage

La figure 6 présente la consommation d'espace de stockage en Go en fonction du modèle de données, les modèles étudiés étant Pig et les tableaux multidimensionnels. Les résultats montrent très clairement que les tableaux multidimensionnels permettent une réduction drastique (environ 90%) du volume de stockage nécessaire pour la gestion des sources de données denses. Nous pouvons ainsi conclure que notre système est moins coûteux en terme de stockage. Par exemple, avec la tarification de l'infrastructure Amazon EC2 (0,15$ par Go et par mois) et pour une source de données d'un To basée sur le modèle de données relationnel, les coûts seraient d'environ 1850 $ par an, alors qu'avec notre approche, la facturation serait d'environ 230 $.

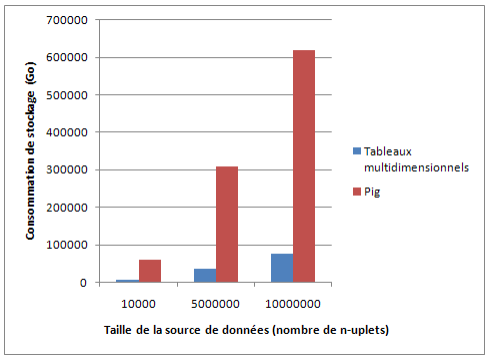

FIG. 6 – *Consommation d'espace de stockage*

4.2 Conversion de données

La figure 7 présente le temps de réponse moyen du processus de conversion de tableaux multidimensionnels en données représentées selon le modèle Pig en fonction de la taille de la source de données exprimée en nombre de n-uplets. Une telle expérimentation vise à illustrer les coûts supplémentaires, en particulier en utilisation processeur, induits par notre proposition. Les résultats montrent que l'exécution d'un tel processus est inférieure à une minute pour une source de données contenant jusqu'à dix millions de n-uplets. Par conséquent, ce coût additionnel peut être considéré comme négligeable. En effet, avec la tarification de l'infrastructure Amazon EC2 (0,12 $ par heure d'utilisation d'une instance standard), un tel processus coûterait au pro rata temporis approximativement 0,001 $ pour une source de données contenant dix millions de n-uplets.

4.3 Optimisation de requête

Enfin, le tableau 2 illustre l'impact de l'optimisation des instructions Pig pour les requêtes OLAP. Il présente le temps de réponse pour l'évaluation d'une requête naïve et d'une requête optimisée, sur une source de données contenant environ 500 000 n-uplets. Les résultats

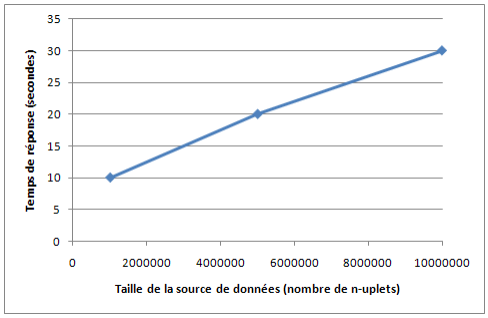

FIG. 7 – *Consommation d'espace de stockage*

montrent clairement que l'optimisation proposée accélère le processus d'évaluation (dans le contexte d'expérimentation étudié, une réduction d'environ 30 %).

	Temps de réponse moyen (secondes)
Requête optimisée FILTER, GROUP et Agrégation)	130
Requête naïve GROUP, Agrégation, et FILTER	190

TAB. 2 – *Influence de l'optimisation d'une requête Pig sur le temps de réponse*

5 Perspectives de recherche

Cette section dresse une liste de perspectives de recherche que nous considérons comme particulièrement cruciales à aborder afin de fournir des requêtes OLAP et des entrepôts de données performants sur nuages. Nous décomposons ces pistes selon deux axes : optimisation des performances (sous-section 5.1), puis modélisation et interrogation (sous-section 5.2).

5.1 Optimisation des performances

Afin de fournir une amélioration des performances des requêtes OLAP, notamment afin de considérer les problèmes liés à la navigation dans des cubes de données volumineux, les pistes suivantes devraient être abordées :

1. *Introduction d'optimiseurs de plan d'exécution pour les requêtes OLAP écrites en Pig Latin.* En effet, comme surligné dans l'article, Pig ne définit aucun plan d'exécution des requêtes. Nous pensons que des fortes possibilités d'amélioration sont possibles grâce à l'adaptation des optimiseurs des SGDB classiques au modèle de données de Pig. De tels optimiseurs pourraient alors être intégrés au niveau du client de manière transparente pour les utilisateurs, afin de respecter la propriété de facilité d'utilisation des nuages informatiques.
2. *Implémentation d'index pour les entrepôts de données en utilisant le paradigme MapReduce.* Les index tels que bitmap, etc. sont utilisés dans les entrepôts de données

pour optimiser les opérations coûteuses comme la jointure [20] ou l'agrégation [30]. Par conséquent, la parallélisation de ces index à travers le paradigme de MapReduce, nous semble être pertinente afin d'exploiter toute la puissance des nuages. Pour cela, nous considérons comme fondamentales lŠextension et/ou lŠadaptation de l'implémentation de ces indexes dans les architectures OLAP distribuées [24, 15, 21] en prenant en compte le paradigme MapReduce et les particularités des architectures sur nuages : la facturation à l'utilisation, la facilité d'utilisation et l'élasticité.

3. *Définition d'algorithmes de sélection de vues matérialisées basés sur le modèle de facturation à l'utilisation.* Les vues matérialisées représentent une technique fondamentale pour l'optimisation des requêtes OLAP dans les architectures ROLAP. Plusieurs travaux ont été proposés pour la sélection "intelligente" des vues matérialisées à calculer [7]. Ces approches ne tiennent pas compte du modèle de facturation à l'utilisation. Nous pensons donc à définir des techniques de matérialisation et de "dématérialisation" qui s'adaptent aux changements des requêtes des utilisateurs et des coûts de stockage et calcul des fournisseurs des nuages.

4. *Intégration de caches pour l'amélioration de la qualité de service et la réduction des coûts d'utilisation.* L'utilisation de cache est cruciale pour améliorer les performances de nombreux systèmes informatiques et plus particulièrement des systèmes décisionnels [28]. Notre objectif est de proposer des techniques de caches sophistiquées et plus précisément des caches sémantiques [19, 10] afin d'améliorer la qualité de service (réduction des temps de réponse, augmentation de la disponibilité) et de réduire les coûts. De tels mécanismes seraient utilisés pour copier les requêtes fréquemment posées (et ainsi économiser les cycles de calcul et la consommation de bande passante).

5.2 Modélisation et interrogation

D'un point de vue modélisation et interrogation des entrepôts de données dans les nuages, d'autres pistes de recherche restent à explorer :

1. *Intégration des opérateurs SQL pour OLAP comme opérateurs natifs de Pig Latin.* En effet, une des caractéristiques les plus importantes des nuages est la facilité d'utilisation. Donc, puisque, comme montré dans ce papier, la définition des requêtes OLAP sur Pig Latin n'est pas immédiate, nous pensons à intégrer l'opérateur Cube [16] dans Pig Latin. Cela nous permettra d'introduire des fonctionnalités typiques du serveur OLAP directement dans le gestionnaire de données sur nuages [13] en facilitant l'utilisation et la définition des systèmes pour l'analyse en ligne dans les nuages.

2. *Implantation des propriétés avancées de modélisation multidimensionnelle grâce à Pig.* Les applications multidimensionnelles peuvent présenter des caractéristiques de modélisation avancée, comme les relations n-n entre faits et dimensions, les mesures complexes, etc. [25] qui sont difficiles à implémenter dans les SGBD relationnels [22]. L'exploration de la puissance du modèle de données de Pig (en ce qui concerne les concepts de bag, map, et nested query) pour la modélisation avancée des entrepôts de données reste une piste importante.

6 Conclusion

Ce papier présente l'intégration des tableaux multidimensionnels dans Pig pour l'entreposage de données sur nuages afin de réduire les coûts de stockage. En particulier, nous avons présenté un algorithme qui permet de convertir les données stockées dans les tableaux multidimensionnels en données basées sur le modèle Pig. Ainsi, les requêtes OLAP peuvent facilement être exécutées en utilisant le langage Pig Latin. Enfin, nous avons proposé une optimisation simple et intuitive des requêtes OLAP en ordonnant les instructions Pig Latin. Les premières expérimentations en local sur des données simulées illustrent la pertinence d'une telle solution. Les résultats montrent clairement que notre proposition permet de réduire la consommation d'espace de stockage et permet ainsi aux utilisateurs des nuages de diminuer les coûts, en utilisant un algorithme de conversion dont le coût est négligeable par rapport aux coûts de l'analyse décisionnelle. Nous avons ensuite listé des perspectives de recherche devant être considérées afin d'intégrer efficacement au sein des nuages informatiques les requêtes OLAP et des entrepôts de données. Actuellement, nous travaillons sur la parallélisation de la conversion à l'aide du paradigme MapReduce afin d'exploiter la puissance de calcul des infrastructures, ainsi que sur l'évaluation de performance de notre proposition sur des centres de données mis à disposition par les fournisseurs de nuages.

Remerciements

Merci à Boussad Mebarki, Ilyas Brahmia, Abdelaziz Merabet, ainsi qu'aux équipes *APIS* du LIMOS et *COPAIN* du cemagref pour les discussions sur les entrepôts de données et les nuages informatiques.

Références

[1] Amazon ec2. web page. http ://aws.amazon.com/ec2/.

[2] Amazon s3. web page. http ://aws.amazon.com/s3/.

[3] Gooddata. web page. http ://www.gooddata.com/products/technology/cloud-bi-platform/.

[4] Hadoop. web page. http ://hadoop.apache.org/.

[5] Microsoft azure. web page. http ://www.microsoft.com/windowsazure/.

[6] Pentaho. web page. http ://www.pentaho.com/cloud_bi/.

[7] K. Aouiche and J. Darmont. Data mining-based materialized view and index selection in data warehouses. *Journal of Intelligent Information Systems*, 33(1) :65–93, 2009.

[8] M. Armbrust, A. Fox, R. Griffith, A. D. J. R. H. Katz, A. Konwinski, G. Lee, D. A. Patterson, A. Rabkin, I. Stoica, and M. Zaharia. Above the clouds : A berkeley view of cloud computing. Technical Report UCB/EECS-2009-28, Berkeley, 2009.

[9] R. Chaiken, B. Jenkins, P.-Å. Larson, B. Ramsey, D. Shakib, S. Weaver, and J. Zhou. Scope : easy and efficient parallel processing of massive data sets. *PVLDB*, 1(2) :1265–1276, 2008.

[10] S. Dar, M. J. Franklin, B. T. Jonsson, D. Srivastava, and M. Tan. Semantic data caching and replacement. In *Proceedings of the international conference on Very Large Data Bases*, pages 330–341, Bombay, India, 1996.

[11] J. Dean and S. Ghemawat. Mapreduce : simplified data processing on large clusters. *Communications of the ACM*, 51(1) :107–113, 2008.

[12] G. DeCandia, D. Hastorun, M. Jampani, G. Kakulapati, A. Lakshman, A. Pilchin, S. Sivasubramanian, P. Vosshall, and W. Vogels. Dynamo : amazon's highly available key-value store. In *Proceedings of the ACM Symposium on Operating Systems Principles*, pages 205–220, Stevenson, USA, 2007.

[13] J.-P. Dittrich, D. Kossmann, and A. Kreutz. Bridging the gap between olap and sql. In *Proceedings of the International Conference on Very Large Data Bases*, pages 1031–1042, 2005.

[14] S. Ghemawat, H. Gobioff, and S.-T. Leung. The google file system. In *Proceedings of the ACM Symposium on Operating Systems Principles*, pages 29–43, Bolton Landing, USA, 2003.

[15] M. Gorawski and R. Malczok. Materialized ar-tree in distributed spatial data warehouse. *Intelligent Data Analysis*, 10(4) :361–377, 2006.

[16] J. Gray, A. Bosworth, A. Layman, and H. Pirahesh. Data cube : A relational aggregation operator generalizing group-by, cross-tab, and sub-total. In *Proceedings of the International Conference on Data Engineering*, pages 152–159, New Orleans, USA, 1996.

[17] W. Inmon. *Building the Data Warehouse*. Wiley, New York, 1996.

[18] M. Isard, M. Budiu, Y. Yu, A. Birrell, and D. Fetterly. Dryad : distributed data-parallel programs from sequential building blocks. In *Proceedings of the EuroSys Conference*, pages 59–72, Lisbon, Portugal, 2007.

[19] A. M. Keller and J. Basu. A predicate-based caching scheme for client-server database architectures. *The Very Large Data Bases Journal*, 5(1) :35–47, 1996.

[20] R. Kimball. *The data warehouse toolkit : practical techniques for building dimensional data warehouses*. John Wiley & Sons, Inc., 1996.

[21] H. Mahboubi and J. Darmont. Enhancing xml data warehouse query performance by fragmentation. In *Proceedings of the ACM Symposium on Applied Computing*, pages 1555–1562, Honolulu, USA, 2009.

[22] E. Malinowski and E. Zimnyi. *Advanced Data Warehouse Design : From Conventional to Spatial and Temporal Applications (Data-Centric Systems and Applications)*. Springer Publishing Company, Incorporated, 2008.

[23] C. Olston, B. Reed, U. Srivastava, R. Kumar, and A. Tomkins. Pig latin : a not-so-foreign language for data processing. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 1099–1110, 2008.

[24] D. Papadias, P. Kalnis, J. Zhang, and Y. Tao. Efficient olap operations in spatial data warehouses. In *Proceedings of the International Symposium on Advances in Spatial and Temporal Databases*, pages 443–459, 2001.

[25] T. B. Pedersen, C. S. Jensen, and C. E. Dyreson. A foundation for capturing and querying complex multidimensional data. *Information Systems*, 26(5) :383–423, 2001.

[26] R. Pike, S. Dorward, R. Griesemer, and S. Quinlan. Interpreting the data : Parallel analysis with sawzall. *Scientific Programming*, 13(4) :277–298, 2005.

[27] M. Rafanelli. Operators for multidimensional aggregate data. In *Multidimensional Databases : problems and solutions*, pages 116–165. 2003.

[28] L. Savary, G. Gardarin, and K. Zeitouni. Geocache : A cache for gml geographical data. *IJDWM*, 3(1) :67–88, 2007.

[29] M. Stonebraker, D. J. Abadi, A. Batkin, X. Chen, M. Cherniack, M. Ferreira, E. Lau, A. Lin, S. Madden, E. J. O'Neil, P. E. O'Neil, A. Rasin, N. Tran, and S. B. Zdonik. C-store : A column-oriented dbms. In *Proceedings of the International Conference on Very Large Data Bases*, pages 553–564, 2008.

[30] Y. Tao and D. Papadias. Historical spatio-temporal aggregation. *ACM Transaction Information Systems*, 23(1) :61–102, 2005.

[31] A. Thusoo, J. S. Sarma, N. Jain, Z. Shao, P. Chakka, S. Anthony, H. Liu, P. Wyckoff, and R. Murthy. Hive - a warehousing solution over a map-reduce framework. *PVLDB*, 2(2) :1626–1629, 2009.

[32] H.-c. Yang, A. Dasdan, R.-L. Hsiao, and D. S. Parker. Map-reduce-merge : simplified relational data processing on large clusters. In *Proceedings of the international conference on Management of data*, pages 1029–1040, Beijing, China, 2007.

[33] S. Zhang, J. Han, Z. Liu, K. Wang, and S. Feng. Spatial queries evaluation with mapreduce. In *Proceedings of the International Conference on Grid and Cooperative Computing*, pages 287–292, 2009.

[34] Y. Zhao, P. Deshpande, and J. F. Naughton. An array-based algorithm for simultaneous multidimensional aggregates. In J. Peckham, editor, *Proceedings of the International Conference on Management of Data*, pages 159–170, Tucson, USA, 1997.

Summary

Data warehouses and OLAP systems are business intelligence technologies. They allow decision-makers to analyze on the fly huge volume of data represented according to the multidimensional model. Cloud computing on the impulse of ICT majors like Google, Microsoft and Amazon, has recently focused the attention. OLAP querying and data warehousing in such a context consists in a major issue. Indeed, problems to be tackled are basic ones for large scale distributed OLAP systems (large amount of data querying, semantic and structural heterogeneity) from a new point of view, considering specificities from these architectures (pay-as-you-go rule, elasticity, and user-friendliness). In this paper we address the pay-as-you-go rules for warehousing data storage. We propose to use the multidimensional arrays storage techniques for clouds. First experiments validate our proposal. Then we list research perspectives.