Omniscience dans la Conception des Entrepôts de Données Parallèles sur un Cluster

Soumia Benkrid*,**, Ladjel Bellatreche** , Alfredo Cuzzocrea ***

* Ecole nationale Supérieure d'Informatique (ESI), Alger, Algérie
s_benkrid@esi.dz
**LIAS/ISAE-ENSMA, Futuroscope, Poitiers, France
(soumia.benkrid, bellatreche)@ensma.fr
***ICAR-CNR - University of Calabria, Italy
cuzzocrea@si.deis.unical.it

Résumé. Généralement, le processus de conception d'un entrepôt de données parallèle passe principalement par deux étapes : (1) la fragmentation des données et (2) l'allocation des fragments générés sur les différents nœuds de traitement. Le principal inconvénient d'une telle approche de conception est le coût élevé de communication pour équilibrer la charge entre les nœuds de traitement, ainsi le nœud coordinateur peut devenir un goulot d'étranglement dans le système. Pour remédier à ces problèmes, la réplication de données (RD) est utilisée. Fréquemment, la fragmentation des données, l'allocation des fragments et la réplication de données sont effectuées de manière isolée. En effet, l'interaction entre ces phases est ignorée. Dans cet article, nous proposons une nouvelle approche de conception d'un entrepôt de données parallèle qui traite conjointement la fragmentation, l'allocation et la réplication. Un algorithme d'allocation redondant basé sur l'algorithme de classification floue "Fuzzy k-means" est proposé. Nous avons également formalisé le problème du traitement parallèle des requêtes comme un Dual Bin Packing, un algorithme glouton est proposé pour la résolution du problème. Enfin, une validation de nos propositions en utilisant le banc d'essai "Star Schema Benchmark" (SSB) est proposée.

1 Introduction

Les nouveaux fournisseurs de données, comme les réseaux sociaux (facebook, linkedin, twitter) les médias numériques, les capteurs, systèmes de commerce électronique, etc. ont largement contribué à la naissance d'une nouvelle ère autour de la gestion des données extrêmement volumineuses. Des conférences et des ateliers autour de cette thématique se sont créés. Nous pouvons ainsi citer Extremely Large Databases Workshop [1], régulièrement sponsorisé par eBay, Facebook et Greenplum. Des applications appuyées par des entreprises se sont créées proposant des solutions d'entreposage, de collection, de traitement, d'analyse de cette mine d'information. De nouveaux métiers ont vu le jour : data analyst, data architect, etc. Un

1. http://www.xldb.org/

nombre important d'entreprises françaises se s'est positionné autour de ce marché pour fournir des solutions de business intelligence. Nous pouvons ainsi citer : Novedia [2], Altic Labs [3], Talend [4], etc. La technologie des entrepôts de données a eu sa part de marché en répondant à des besoins d'analyse des entreprises quelle que soit leur nature (PME ou autre).

En conséquence, ces entreprises font face à un défi de gestion et d'exploitation efficace des données extrêmement volumineuses. Malheureusement, les systèmes de gestion d'entrepôts de données traditionnels ne sont pas bien adaptés pour traiter cette masse de données. Pour atteindre la haute performance, l'évolutivité, la fiabilité et la disponibilité dans les entrepôts de données volumineux, le recours aux traitements parallèles est devenu un enjeu important Apers (1988); Nehme et Bruno (2011). L'architecture sans partage (Shared-Nothing) a été proposée par DeWitt DeWitt et al. comme l'architecture de référence pour la mise en œuvre des entrepôts de données à haute performance modélisés en schéma en étoile. Des solutions commerciales ont été proposées par des entreprises comme Teradata [5] et Netezza [6]. Les solutions de Teradata ont été adoptées par des grandes entreprises françaises comme Carrefour et Banque Populaire Group [7]. Cependant, elles restent coûteuses pour des PME en termes de droits de licence et coûts d'installation et de maintenance. Avec la crise économique, ces dernières cherchent à trouver des solutions efficaces et moins coûteuses.

Dans le même temps, l'évolution technologique a facilité la création des clusters d'ordinateurs à *moindre coût*. En effet, les clusters de base de données ($\mathcal{CDB}$) ont montré leur intérêt pour concevoir des Entrepôts de Données Relationnels Parallèles ($\mathcal{EDRP}$) Lima et al. (2004, 2009). Cette conception hérite les mêmes étapes de conception d'un $\mathcal{EDRP}$, à savoir : **(1)** la fragmentation de l'entrepôt de données, **(2)** l'allocation des fragments générés sur les nœuds du cluster et **(3)** la définition d'une stratégie de traitement des requêtes. Le problème de fragmentation des données consiste à diviser le $\mathcal{EDR}$ en unités disjointes appelées "partitions". Le partitionnement peut se faire horizontalement ($\mathcal{FH}$) ou verticalement ($\mathcal{FV}$). Le $\mathcal{FH}$ est essentiellement utilisé pour la conception des entrepôts de données parallèles. L'allocation des données consiste à placer les fragments générés sur les nœuds du cluster. Cette allocation peut être soit redondante (avec réplication) ou non redondante (sans réplication). Une fois les fragments placés, les requêtes sont exécutées sur les nœuds du cluster. Le traitement parallèle des requêtes englobes : (1) la réécriture des requêtes globale selon le schéma de fragmentation et (2) l'allocation des sous-requêtes générées sur les nœuds du cluster selon le schéma d'allocation pour que les nœuds soient uniformément chargés. D'autres problèmes sous-jacents aux étapes principales ci dessous décrites doivent être considérés durant la conception d'un $\mathcal{EDRP}$. Il s'agit du problème de l'équilibrage de charge entre les nœuds du cluster, qui est primordial pour atteindre la haute performance de l'$\mathcal{EDRP}$. En effet, un déséquilibre de charge peut être causé par l'un (ou la combinaison) des deux problèmes suivants Taniar et al. (2002) : **(i)** la mauvaise répartition des données et **(ii)** la mauvaise répartition de traitement (Processing Skew). Dans la littérature, l'équilibrage de charge est effectué via une redistribution des données des nœuds surchargés sur les nœuds sous chargés Loukopoulos et Ahmad (2004). Cette migration des données peut engendrer un coût de communication élevé et le nœud coordinateur

2. http://www.novediagroup.com/
3. http://www.altic.org/
4. fr.talend.com
5. http://www.teradata.com/
6. www.ibm.com/software/fr/data/netezza/
7. http://www.teradata.com/browsecustomers.aspx?page=All

peut devenir un goulot d'étranglement. En conséquence, la *réplication des données* est devenue une exigence pour éviter le goulot d'étranglement et réduire le coût de communication. En effet, la réplication de données assure : (a) la disponibilité des données et la tolérance aux pannes en cas de défaillance, (b) la localité de traitement et (c) l'équilibrage de charge.

D'après cette discussion, nous concluons que la conception d'un $\mathcal{EDRP}$ peut être définie comme un 5-uplets : $< \mathcal{F}, \mathcal{A}, \mathcal{R}, \mathcal{EC}, \mathcal{TR} >$, où $\mathcal{F}, \mathcal{A}, \mathcal{R}, \mathcal{EC}$ et $\mathcal{TR}$ représentent respectivement les processus de fragmentation, d'allocation, de réplication, d'équilibrage de charge et de traitement de requêtes. Les problèmes associés aux cinq composantes sont tous NP-hard Ahmad et al. (2002); Furtado (2004); Pavlo et al. (2012); Saccà et Wiederhold (1985); Wolfson et Jajodia (1992). De nombreux travaux ont été proposés dans la littérature sur la conception des bases de données parallèles en général et les entrepôts de données parallèles, en particulier. Après une analyse fine, nous avons remarqué que la majorité de ces travaux traitent les problèmes liés à la conception d'une manière indépendante pour contrôler la complexité globale du problème de conception parallèle. En effet, il existe des méthodes qui ne s'intéressent qu'à la phase de fragmentation des données Stöhr et al. (2000); Rao et al. (2002); Zilio et al. (2004); Nehme et Bruno (2011), d'autres focalisent sur la phase d'allocation des données Apers (1988); Ahmad et al. (2002); Menon (2005), la réplication de données Ciciani et al. (1990); i Hsiao et Dewitt (1990); Loukopoulos et Ahmad (2004), et le traitement parallèle des requêtes Akal et al. (2002); Lima et al. (2004). Deux inconvénients majeurs de cette conception sont identifiés : (1) l'ignorance de l'interdépendance entre les différentes phases de conception et (2) le fait que dans chaque phase, une métrique est considérée par son acteur pour identifier la qualité de solution proposée, cela génère une multitude de métriques hétérogènes qui peuvent pénaliser la solution finale. Des efforts ont été envisagés pour considérer cette interaction. La forte interaction entre la fragmentation et l'allocation étaient les deux premières phases de la conception qui étaient traitées d'une manière conjointe, sous le nom du placement de données Copeland et al. (1988); Hua et Lee (1990). Dans l'an 2000, les travaux de Stöhr et al. Stöhr et al. (2000) étaient les pionniers à avoir revisité le problème de data placement dans le contexte des $\mathcal{EDRP}$ sur une machine parallèle à disque partagé. Récemment, une volonté des industriels et des académiciens est de combiner les différentes phases de conception d'un $\mathcal{EDRP}$ Bellatreche et al. (2012).

Dans cet article, nous proposons une méthode de conception d'un $\mathcal{EDRP}$ sur un cluster. L'idée principale est d'exploiter l'interaction entre les différentes phases de conception d'un $\mathcal{EDRP}$et d'utiliser un seul modèle de coût unifiant l'ensemble des phases. Cette unification permet de *cimenter les phases* et augmente l'omniscience des acteurs de cette conception. Nous suggérons que pour avoir cette unification des phases de conception, la fragmentation joue un rôle primordial, étant considérée comme la première étape importante de cette conception. Durant la phase de fragmentation, le partitionneur doit avoir en tête qu'il décompose l'$\mathcal{EDR}$ pour qu'il soit bon pour l'allocateur, le répliqueur, l'équilibreur et le traiteur de requêtes. La qualité de la conception est mesurée par le modèle de coût unifié. L'idée de base derrière notre proposition est que lors de la fragmentation, chaque solution potentielle de fragmentation est testée pour l'allocation, la réplication et l'équilibrage de charge. La solution ayant un coût minimal est retenue pour la conception de l'$\mathcal{EDRP}$. Nous sommes conscients que cette approche peut être coûteuse. Pour réduire cette complexité, le développement des algorithmes moins coûteux est envisagé dans cet article.

Cet article comporte les sections suivantes : La section 2 est consacrée à la description de

l'approche proposée. La section 3 décrit la stratégie proposée pour le traitement parallèle des requêtes. La section 4 présente une validation de l'approche à travers des expérimentations obtenues en utilisant le banc d'essai SSB. Enfin, la section 5 récapitule les principaux résultats et donne quelques perspectives à explorer.

2 Solution proposée : Approche $\mathcal{F}$&$\mathcal{A}$&$\mathcal{R}$

Dans cette section, nous décrivons en détail notre approche proposée pour la conception d'un EDP, que nous appelons $\mathcal{F}$&$\mathcal{A}$&$\mathcal{R}$. L'approche $\mathcal{F}$&$\mathcal{A}$&$\mathcal{R}$ consiste à partitionner l'entrepôt de données et en même temps, elle alloue les fragments d'une manière redondante, il nous faut une procédure de fragmentation et un autre d'allocation redondante. Pour la fragmentation, nous adaptons notre algorithme génétique proposé dans nos propositions Bellatreche et al. (2010, 2012). La principale adaptation concerne la stratégie d'allocation des fragments. Nous proposons une stratégie d'allocation redondante.

2.1 Procédure de fragmentation

Pour sélectionner le schéma de partitionnement horizontal, l'approche $\mathcal{F}$&$\mathcal{A}$&$\mathcal{R}$ adopte notre algorithme génétique proposé dans Bellatreche et al. (2010). A partir de la charge de requêtes, nous devons identifier quelles tables de dimension participant dans la fragmentation de la table des faits. Pour se faire, nous procédons de la façon suivante : (1) extraire tous les prédicats de sélection utilisés par les requêtes de Q, (2) attribuer à chaque table de dimension D_i ($1 \leq i \leq n$) son ensemble de prédicats de sélection , noté par SPD_i, (3) Eliminer les tables de dimension D_i ayant un SPD_i vide , (4) identifier l'ensemble des attributs de fragmentation candidats, (5) Eliminer les attributs ayant un degré de skew élevé (6) décomposer le domaine de valeurs de chaque attribut de fragmentation en plusieurs sous-domaines. En conséquence, le schéma de fragmentation (chromosome) peut être représenté par un tableau multidimensionnel. Au démarrage de l'algorithme génétique, nous générons une population initiale d'une manière aléatoire. Pour chaque individu généré, nous vérifions s'il satisfait la contrainte de maintenance $(NF_i \leq W)$ tel que NF_i représente le nombre des fragments. Si c'est le cas, il est gardé dans la population. Dans le cas contraire, des opérations de fusions sont effectuées afin de réduire son nombre de fragments. Une fois la population initiale créée, notre algorithme effectue des opérations génétiques (croisement et mutation) afin d'améliorer la population. L'application de ses opérations est contrôlée par une fonction d'évaluation. Cette dernière alloue les fragments de chaque individu d'une manière redondante sur les différents nœuds en calculant le coût d'exécution de la charge de requêtes sur les fragments placés en utilisant un modèle de coût (section 3). A la fin, le chromosome qui offre le coût minimal représente le schéma de fragmentation. Dans la section suivante, nous décrivons la procédure d'allocation utilisée pour placer chaque chromosome.

2.2 Procédure de d'Allocation Redondante des fragments

Dans sa forme générique, le problème de l'allocation des données consiste à déterminer le meilleur emplacement pour un ensemble de fragments sur les nœuds d'un cluster afin de minimiser le coût de réponse d'une charge de requêtes, il peut être assimilé à un problème de

classification. En effet, le problème de classification place un ensemble d'objets de données dans un certain nombre de classes en fonction d'une métrique.

L'allocation des fragments est fortement liée à la réplication des fragments. Pour cette raison, nous proposons de formaliser le problème d'allocation comme un problème de *classification floue* et le résoudre par *l'algorithme fuzzy k-means* Bezdek et Full (1984). Dans les techniques de classification floue, chaque point dans l'ensemble des données appartient à chaque classe avec un certain degré nommé " degré d'appartenance ". Ces degrés d'appartenance sont entre 0 et 1. Le principe sous-jacent de classification floue est l'attribution d'éléments de données à plusieurs clusters, avec divers degrés d'appartenance. Ainsi, le problème d'allocation des fragments est formalisé comme suit :
Considérons un ensemble de fragments $\mathcal{F} = \{F_1, F_2, \ldots, F_{NF}\}$ avec d dimensions dans l'espace euclidien R^d, i.e. $F_j \in R^d$. Le problème consiste à affecter chaque fragment F_i à R (R représente le degré de réplication) sous-ensembles flous en minimisant une fonction objectif. *Le résultat de la classification floue peut être exprimé par une matrice d'appartenance U dont la valeur $U[i][j] = u_{ij}$, tel que $i = 1..M$ et $j = 1..NF$, où u_{ij} appartient à [0,1] et satisfait la contrainte suivante :*

$$\forall 1 \leq j \leq NF : \sum_{i=1}^{M} u_{ij} = 1 \tag{1}$$

La fonction objective f_O à optimiser est définie comme suit :

$$f_O = \sum_{k=1}^{NF} \sum_{i=1}^{M} u_{ij}^m \, ||X_k - V_i||^2 \tag{2}$$

Où : (*i*) $m \succ 1$ est le cœfficient de flou, (*ii*) X_k est le vecteur des points de données (*iii*) V_i est le centre des clusters C_i, (*iv*) $||X_k - V_i||^2$ représente la distance Euclidienne entre X_k et V_i.

Les étapes de notre procédure d'allocation sont les suivants :

1. *Construction de la Matrice d'Usage des Fragments(MUF)*. Comme son nom l'indique, cette matrice représente l'utilisation des fragments par les requêtes. Les lignes et les colonnes de cette matrice sont associées aux n requêtes de départ et les NF fragments obtenus par le schéma de fragmentation en cours d'évaluation, respectivement. La valeur $MUF[i][j]$ tel que ($1 \leq i \leq L$ and $1 \leq j \leq NF$) est égale à 1, si la requête Q_i utilise le fragment F_j. sinon, elle égale à 0. Nous ajoutons à cette matrice une colonne représentant la fréquence d'accès de chaque requête.
2. *Représentation de chaque Fragment dans R^2* : chaque fragment F_i est représenté dans l'espace vectoriel à deux-dimensions R^2 par les coordonnés (x, y). Les coordonnés du fragment F_i dans R^2 sont établies à partir du poids de classification comme suit : Le calcul d'un point d'un fragment F_i est égal à la somme des fréquences d'accès de toutes les requêtes qui n'utilisent pas le fragment. Une fois le poids calculé, les coordonnées dans R^2 de chaque fragment F_i sont spécifiées comme suit : $(x, y) = (i, poids(F_i))$.
3. *Construction de la Matrice d'Appartenance des Fragments (MAF)*. Cette matrice représente le degré d'appartenance d'un fragment F_k à une classe C_i selon l'ensemble des

requêtes $\mathcal{Q}$. Les lignes et les colonnes de cette matrice sont associées aux NF fragments et les M nœuds respectivement. La valeur $MAF[i][j]$, tel que ($1 \leq i \leq NF$ and $1 \leq j \leq M$), appartient à l'intervalle $[0, 1]$. Cette valeur est calculée comme suit :

$$MAF_{ij} = \left[\sum_{l=1}^{NF} \left(\frac{d_{ij}}{d_{il}}\right)^{\frac{2}{m-1}}\right]^{-1}$$

tel que : d_{ij} représente la distance Euclidienne entre X_k et V_i et m représente le cœfficient flou de la partition.

4. *Regroupement des Clusters*. Afin de générer des groupes de fragments, nous faisons usage du principe que les valeurs d'appartenance les plus élevées indiquent la confiance d'attribution des objets au cluster. Ainsi, nous avons trié les valeurs d'appartenance dans l'ordre décroissant et nous avons assigné à chaque fragment F_k $(1 \leq k \leq NF)$ aux R (R étant le degré de réplication) premiers groupes, tel que la contrainte de placement de données est satisfaite. À la fin de cette étape, un ensemble de cluster $C = C_1, \ldots, C_M$ est généré, de telle sorte que chacun d'eux représente un sous-ensemble de fragments.
5. *Construction de la Matrice de Placement des Fragments (MPF)*. Cette matrice représente la présence des fragments sur les nœuds. Les lignes et les colonnes de cette matrice sont associées aux N fragments obtenus par le schéma de fragmentation en cours d'évaluation et les M nœuds associés au cluster, respectivement. Les éléments de la Matrice MPF sont binaires (0 ou 1). $MPF[i][m] = 1$, avec $1 \leq i \leq NF$ et $1 \leq m \leq M$, si le fragment F_i est alloué sur le nœud N_m, sinon $MPF[i][m] = 0$. Notre procédure d'allocation considère le cluster comme étant l'unité d'allocation. Les clusters sont placés d'une manière circulaire sur les nœuds.

3 Stratégie de Traitement des requêtes

Une fois le schéma de fragmentation généré et les fragments alloués, les requêtes globales seront évaluées sur le cluster de base de données $\mathcal{DBC}$. Le principal objectif est de minimiser le temps d'exécution total de l'ensemble des requêtes. Pour évaluer une requête de jointure en étoile, il faut identifier d'abord les fragments valides et leurs localisations sur les nœuds. Comme notre allocation est redondante (chaque fragment est alloué sur R nœuds), Nous utilisons un ordonnanceur pour trouver l'allocation la plus judicieuse pour chaque sous-requête. Il est à noter que chaque fragment valide donne lieu à une sous-requête. Le problème de traitement parallèle pour l'approche $\mathcal{F}\&\mathcal{A}\&\mathcal{R}$ a pour but la minimisation du coût d'exécution total des requêtes de $\mathcal{Q}$ exécutées sur le cluster en satisfaisant la contrainte d'équilibrage de charge δ.

Le problème introduit ci-dessus est similaire à un dual du problème de remplissage *(Dual Bin Packing Problem (DBPP))*. Il est défini comme NP-complet Jr. et al. (1978). Ainsi, pour produire une solution quasi-optimale pour ce problème, nous proposons un algorithme glouton qui doit trouver le meilleur placement des requêtes qui minimise le coût d'exécution moyen de la charge de requêtes.

Tout d'abord, nous identifions les fragments valides et leurs sous-requêtes associées, le nombre de fragments valides et le nombre des nœuds valides pour l'exécution de l'ensemble des sous-requêtes . Ensuite, nous estimons le temps de traitement, noté $PTSQ_i$, nécessaire

pour évaluer chaque sous-requête Q_j et on initialise le seuil de traitement moyen par la constante LB. Nous avons ensuite trié la liste des fragments valides en ordre décroissant selon leurs taille et, pour chaque sous-requête générée , nous procédons comme suit : (1) sélectionner les nœuds valides, (2) calculer la charge de chaque nœud valide ; (3) affecter la sous-requête au nœud ayant la plus grande capacité résiduelle . Ceci assure l'allocation des sous-requêtes sur des fragments et leurs répliques de telle sorte que la contrainte d'équilibrage de charge est satisfaite.

4 Experimental Evaluation and Analysis

Dans cette section, nous avons mené une série d'expérimentations afin de valider et mettre en évidence le gain apporté par notre approche $\mathcal{F}\&\mathcal{A}\&\mathcal{R}$. Nous commençons par la présentation des paramètres d'expérimentation puis nous présentons les résultats obtenus.

Afin de valider l'approche proposée, nous avons mené une série d'expérimentations basée sur le banc d'essai Star Schema Benchmark ($\mathcal{SSB}$). Nous avons également généré une charge de 36 requêtes de jointure en étoile à partir des 13 requêtes proposées par $\mathcal{SSB}$. Pour les paramètres de l'algorithme génétique, nous avons fixé le taux de croisement à (70%) et le taux de mutation à (30%) pour améliorer 40 chromosomes en 20 générations.

Dans la première expérience, nous étudions les performances de notre approche proposée. Nous avons comparé $\mathcal{F}\&\mathcal{A}\&\mathcal{R}$ avec trois autres approches de conception d'un EDP : (1) le partitionnement, l'allocation et la réplication sont traitées de manière isolée, (2) la réplication de données est effectuée une fois que la fragmentation et d'allocation sont effectuées conjointement (3) l'allocation et la réplication sont traitées de manière conjointe et séparément du partitionnement. La figure 1 compare les performances relatives des quatre méthodes de conception d'un EDP en fixant le seuil de fragmentation à 100 et le skew de valeur d'attribut à $0,5$. Pour chaque approche de conception, nous faisons varier le degré de réplication de 1 à 10 et pour chaque valeur, nous avons calculé le nombre d'E/S nécessaire pour exécuter la charge des requêtes sur un cluster de 10 nœuds. Nous remarquons que l'approche $\mathcal{F}\&\mathcal{A}\&\mathcal{R}$ est plus adaptée à la conception d'un EDP que l'approche itérative et ses variantes. A partir de ces résultats, nous constatons également que l'augmentation du degré de réplication implique une augmentation de la performance du système en minimisant le coût d'exécution des requêtes.

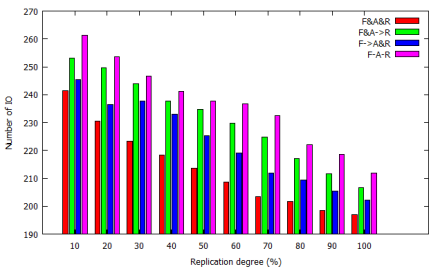

FIG. 1: Comparaison entre les approches de conception d'un EDP

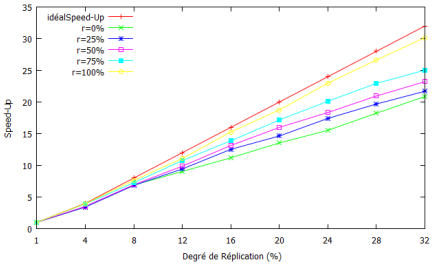

FIG. 2: Effet du degré de réplication sur le speed-up de $\mathcal{F}\&\mathcal{A}\&\mathcal{R}$

Dans la deuxième expérience, nous avons étudié l'impact de la réplication sur la scalabilité de notre approche. Pour cela, nous calculons le facteur de rapidité (speed up). Pour un seuil de fragmentation de 100, nous avons fait varier le nombre de nœuds de 1 à 32 et pour chaque valeur, nous calculons le speed up pour les degrés de réplication suivant : $\mathcal{R}$: $8(25\%)$, $16(50\%)$, $24(75\%)$ et $32(100\%)$. Les résultats obtenus sont présentés dans la Figure 2 et confirment que l'approche proposée est bien adaptée à la conception d'un EDP. De plus, l'augmentation du degré de réplication permet d'offrir une meilleure accélération. Dans le cas $\mathcal{R} = 100\%$, l'accélération est approximativement linéaire. Cela est dû au fait que la réplication donne des avantages supplémentaires découlant de l'équilibrage de charge qui n'élimine pas complètement les effets du skew. Cependant, la réplication requiert plus de mémoire pour le stockage et la maintenance des répliques (que nous négligeons dans cet article). Le degré de réplication doit donc être bien paramétré.

Dans la troisième expérience, nous nous sommes intéressés à la dépendance entre le paramètre de l'approche $\mathcal{F}\&\mathcal{A}\&\mathcal{R}$. Nous avons étudié la dépendance entre le degré de skew des valeurs d'attribut et le degré de skew de placement des fragments. Nous avons fixé le seuil de fragmentation à 100 et le nombre de nœuds à 10. Nous faisons varier le degré de skew des valeurs d'attribut de $0,2$ à 1 et pour chaque valeur, nous avons calculé le degré de skew de placement des fragments. Figure 3 montre les résultats obtenus et confirme que la mauvaise répartition de données augmente considérablement quand le degré de skew des valeurs d'attribut augmente.

Dans la quatrième expérience, nous avons étudié l'impact du degré de réplication sur le traitement en parallèle. Nous faisons varier le degré de réplication de 1 à 10 et, pour chaque valeur, nous avons calculé le degré de d'équilibrage de charge. Comme le montre la figure 4, l'augmentation du degré de réplication réduit les effets négatifs du skew de données. D'autre part, l'augmentation du degré de réplication facilite l'atteinte de la haute performance de l'EDP.

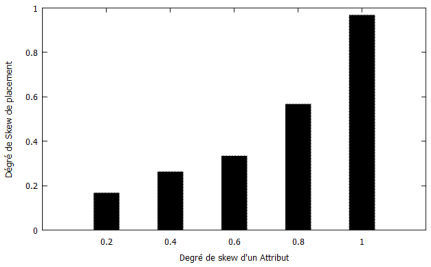

FIG. 3: Dépendance entre attribute skew des valeurs d'un attribut et skew de partitioning de données

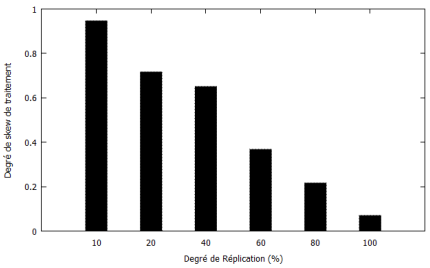

FIG. 4: Dépendance entre degré de réplication et skew de traitement processing

5 Conclusion

Dans cet article, nous avons montré l'intérêt d'unifier les phases de conception d'un $\mathcal{EDRP}$. Nous avons présenté une méthode de conception nommée $\mathcal{F}\&\mathcal{A}\&\mathcal{R}$, qui est une extension de nos propositions dans Bellatreche et al. (2012). Une méthode de réplication originale, basée

sur la logique floue est intégrée dans $\mathcal{F}\&\mathcal{A}\&\mathcal{R}$. Le modèle de coût évaluant la qualité de notre solution intègre les différents concepts de phases de la conception. Nous avons également évalué notre approche en utilisant le banc d'essai SSB. Les résultats expérimentaux sont encourageants et montrent la faisabilité de notre approche.

Pour réduire la complexité de notre solution, nous avons considéré des algorithmes de faible coût d'exécution. Il serait intéressant de proposer des algorithmes avancés tout en parallélisant les différentes étapes de la conception.

Références

Ahmad, I., K. Karlapalem, Y.-K. Kwok, et S.-K. So (2002). Evolutionary algorithms for allocating data in distributed database systems. *Distributed and Parallel Databases 11*(1), 5–32.

Akal, F., K. Böhm, et H.-J. Schek (2002). Olap query evaluation in a database cluster : A performance study on intra-query parallelism. In *ADBIS*, pp. 218–231.

Apers, P. M. G. (1988). Data allocation in distributed database systems. *ACM Transactions on database systems 13*(3), 263–304.

Bellatreche, L., S. Benkrid, A. Crolotte, A. Cuzzocrea, et A. Ghazal (2012). The *f*&*a* methodology and its experimental validation on a real-life parallel processing database system. In *CISIS*, pp. 114–121.

Bellatreche, L., A. Cuzzocrea, et S. Benkrid (2010). *F*&*a* : A methodology for effectively and efficiently designing parallel relational data warehouses on heterogenous database clusters. In *DaWak*, pp. 89–104.

Bezdek, J. C., R. et W. Full (1984). Fcm : The fuzzy c-means clustering algorithm. *Computers and Geo-sciences) 10*(2-3), 191–203.

Ciciani, B., D. M. Dias, et P. S. Yu (1990). Analysis of replication in distributed database systems. *IEEE Trans. on Knowl. and Data Eng.*, 247–261.

Copeland, G. P., W. Alexander, E. Boughter, et T. Keller (1988). Data placement in bubba. In *ACM SIGMOD*, pp. 99–108.

DeWitt, D., S. Madden, et M. Stonebraker. How to build a high-performance data warehouse. *http ://db.lcs.mit.edu/madden/high_perf.pdf*.

Furtado, P. (2004). Experimental evidence on partitioning in parallel data warehouses. In *ACM DOLAP*, pp. 23–30.

Hua, K. A. et C. Lee (1990). An adaptive data placement scheme for parallel database computer systems. In *VLDB*, pp. 493–506.

i Hsiao, H. et D. J. Dewitt (1990). Chained declustering : A new availability strategy for multiprocssor database machines. In *in Proceedings of 6th International Data Engineering Conference*, pp. 456–465.

Jr., E. G. C., J. Y.-T. Leung, et D. W. Ting (1978). Bin packing : Maximizing the number of pieces packed. *Acta Informatica 9*, 263–271.

Lima, A. A. B., M. Mattoso, et P. Valduriez (2004). Adaptive Virtual Partitioning for OLAP Query Processing in a Database Cluster. In *SBBD*, pp. 92–105.

Lima, A. B., C. Furtado, P. Valduriez, et M. Mattoso (2009). Parallel olap query processing in database clusters with data replication. distributed and parallel databases. *Distributed and Parallel Database Journal 25*(1-2), 97–123.

Loukopoulos, T. et I. Ahmad (2004). Static and adaptive distributed data replication using genetic algorithms. *in Journal of Parallel and Distributed Computing 64*(11), 1270–1285.

Menon, S. (2005). Allocating fragments in distributed databases. *IEEE Transactions on Parallel and Distributed Systems 16*(7), 577–585.

Nehme, R. V. et N. Bruno (2011). Automated partitioning design in parallel database systems. In *SIGMOD Conference*, pp. 1137–1148.

Pavlo, A., C. Curino, et S. Zdonik (2012). Skew-aware automatic database partitioning in shared-nothing, parallel oltp systems. In *ACM SIGMOD*, pp. 61–72.

Rao, J., C. Zhang, G. Lohman, et N. Megiddo (2002). Automating physical database design in a parallel database. pp. 558–569.

Saccà, D. et G. Wiederhold (1985). Database partitioning in a cluster of processors. *ACM Transactions on Database Systems 10*(1), 29–56.

Stöhr, T., H. Märtens, et E. Rahm (2000). Multi-dimensional database allocation for parallel data warehouses. In *VLDB*, pp. 273–284.

Taniar, D., Y. Jiang, K. H. Liu, et C. H. C. Leung (2002). Parallel aggregate-join query processing. *Informatica (Slovenia) 26*(3).

Wolfson, O. et S. Jajodia (1992). Distributed algorithms for dynamic replication of data. In *Proceedings of the eleventh ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems*, pp. 149–163.

Zilio, D. C., J. Rao, S. Lightstone, G. M. Lohman, A. Storm, C. Garcia-Arellano, et S. Fadden (2004). Db2 design advisor : Integrated automatic physical database design. pp. 1087–1097.

Summary

The process of designing a parallel data warehouse has two main steps: (1) the data fragmentation and (2) the fragment allocation of generated fragments at various nodes. Usually, we split the data warehouse horizontally, allocate fragments over nodes, and Finally balance the load over the nodes of the parallel machine. The main drawback of such design approach is that the high communication cost and the coordinator node can be a bottleneck. Therefore, *Data Replication* (DR) may play a great role to avoid the bottleneck and minimizing the communication cost on the other hand. Usually, the data fragmentation, fragment allocation and data replication are performed in an isolated way. As a consequence, interaction between these phases is ignored. In this paper, we first propose to package these phases into one design methodology. Secondly, a *redundant allocation* algorithm based on the well-known *fuzzy k-means clustering algorithm* is given. Thirdly, the query processing problem in a unified methodology is formalized as a Dual Bin Packing Problem and a greedy algorithm is proposed. Finally, our proposal is also experimentally assessed and validated against the widely-known data warehouse benchmark Star Schema Benchmark (SSB)