Vers une Conception des Entrepôts de Données Parallèles Autonomes

Soumia Benkrid*, Ladjel Bellatreche**

* Ecole nationale Supérieure d'Informatique (ESI), Alger, Algérie
s_benkrid@esi.dz
**LIAS/ISAE-ENSMA, Futuroscope, Poitiers, France
bellatreche@ensma.fr

Résumé. Les systèmes de stockage de données parallèles (SSDPs) sont devenus une des solutions incontournables pour traiter les données massives pour des fins d'analyse. L'efficacité de ces systèmes dépend fortement des processus de fragmentation et d'allocation des partitions sur l'ensemble des nœuds. En examinant les travaux existants, nous avons constaté qu'ils se basent sur les requêtes, une chose usuelle pour toute méthodologie de conception des systèmes complexes. Les requêtes d'entrée dans le contexte de SSDPs sont souvent connues à l'avance (et statiques) et peuvent évoluer. Les approches utilisées pour concevoir des SSDP sont alors réactives. Dans la BI 2.0, où les utilisateurs (décideurs) sont au centre du système avec *leurs requêtes en lots et ad-hoc,* les approches réactives peuvent facilement montrer leur limite. Pour faire face à cette situation, le recours aux techniques issues des systèmes autonomes *intelligents* (comme l'informatique proactive) est nécessaire. Dans cet article, nous proposons une approche proactive de conception des SBDPs comportant deux phases principales : (i) une phase hors ligne qui génère des schémas de fragmentation et d'allocation à partir des requêtes supposées connues et vues comme une base d'apprentissage et (ii) une phase en ligne qui augmente les schémas obtenus par la phase hors ligne afin de prendre en compte les nouvelles requêtes ad-hoc par lots. La prise en charge de ces requêtes est assurée par l'exploitation des résultats intermédiaires identifiés dans la phase hors ligne. Les résultats de nos expérimentations montrent clairement l'intérêt de l'informatique proactive pour la conception des SSDPs autonomes.

1 Introduction

La Business Intelligence (BI) traditionnelle a été souvent construite pour offrir des mécanismes d'exploitation des données historisées d'entreprises à l'aide des requêtes supposées connues à l'avance. La connaissance préalable des requêtes a été largement utilisée pour déployer (Stöhr et al., 2000), optimiser (Bellatreche, 2018) et exploiter (Gallinucci et al., 2018) des entrepôts de données ($\mathcal{ED}$) de la première génération. La large considération de cette hypothèse par les chercheurs est souvent motivée par la complexité des problèmes associés au

déploiement, optimisation des accès et l'exploitation. Avec les nouvelles exigences des entreprises en termes d'acquisition de plus en plus de sources de données externes et leur volonté de mettre l'utilisateur final au cœur du processus d'analyse, un nouveau concept a ainsi émergé, il s'agit de la nouvelle BI ou la « BI Next Generation » dont la particularité est qu'elle soit une BI « ad-hoc ». Devant cette situation, la proposition des solutions et des services offrant une exploitation évolutive et efficace de l'$\mathcal{ED}$ est nécessaire.

Cette évolution a fortement impacté la composante sources, où des ressources externes sont intégrées dans le processus de construction d'un $\mathcal{ED}$ (Berkani et al., 2019), la phase d'ETL (Gallinucci et al., 2018). A cet égard, la phase de déploiement a été entendue par l'intégration de Hadoop (Chen et Wang, 2018). Plusieurs académiques et industriels (Benkrid et al., 2017; Ordonez et Bellatreche, 2018) proposent la cohabitation des SGBD parallèles et Hadoop. Dans cet article, nous traitons le problème de déploiement d'un $\mathcal{ED}$ sur un SGBD parallèle.

La conception d'un entrepôt de données parallèle ($\mathcal{EDP}$) est un processus complexe. Il consiste d'abord à partitionner l'entrepôt, ensuite allouer les fragments générés sur les nœuds de l'architecture parallèle. Une stratégie de traitement des requêtes doit être définie pour atteindre la haute performance de l'$\mathcal{EDP}$. Un nombre important de travaux existe et couvre les phases de cette conception : le partitionnement des données (Curino et al., 2010; Nehme et Bruno, 2011; Pavlo et al., 2012; Quamar et al., 2013; Shanbhag et al., 2017; Dong et al., 2017), l'allocation de fragments (Apers, 1988; Ozsu et Valduriez, 2011; Rabl et Jacobsen, 2017), le placement de données (Benkrid et al., 2014) et l'ordonnancement des requêtes (Karnagel et al., 2017). La connaissance des requêtes est une précondition de ces travaux. L'acteur en charge de déploiement doit avoir une visibilité sur deux éléments importants afin de bien mener son travail : (a) l'acquisition des requêtes (les requêtes d'entrée sont connues à l'avance ou prédictives) et (b) leur stabilité (statiques ou évolutives).

L'acquisition des requêtes en général, et la prédiction des requêtes d'entrées, en particulier, a été récemment étudiée dans le contexte de pilotage automatique des bases de données (Ma et al., 2018), la sélection des schémas de partitionnement des données (Durand et al., 2018; Dong et al., 2017) et de vues matérialisées (Du et al., 2017) et la sélection des expressions communes (Jindal et al., 2018). Dans ces travaux, les auteurs proposent des mécanismes de prédiction de requêtes, les techniques d'apprentissage d'automatique et d'autres basées sur l'utilité en ligne. Ses travaux sont en quête des requêtes. En conséquence, leurs approches de conception sont réactives basées sur des techniques de réglages. En effet, nous identifions deux aspects limitants les approches. Premièrement, la plupart des travaux effectués autour du placement de données ne se soucient pas de l'exécution de la charge de requêtes par lots. Deuxièmement, la majorité des approches existantes propose des solutions pour des charges, connues au préalable, contenant une dizaine ou une centaine de requêtes. Or, actuellement, la BI offre aux utilisateurs la possibilité de générer des milliers de requêtes ad-hoc par jour.

En se basant sur cette discussion la mise en place d'une approche de conception d'un $\mathcal{EDP}$ autonome est devenue un enjeu important. Autonome signifie que la plateforme de déploiement doit être capable de reconnaître, de détecter et de diagnostiquer les écarts par rapport aux conditions normales. Dans cet article, nous proposons une approche proactive de conception d'un $\mathcal{EDP}$ comportant deux phases principales : (1) **hors-ligne** dans laquelle, des techniques d'apprentissage automatique sont utilisées pour identifier la meilleure conception et (2) **en-ligne** dans laquelle un planning basé sur des vérifications quantitatives pour déterminer la meilleure adaptation en fonction de l'ensemble des configurations optimales identifiées lors de

la phase hors-ligne. Il est à noter que la phase hors-ligne joue un rôle crucial dans la conception d'un système autonome car son résultat représente la base de connaissance pour le système en ligne.

Le papier est structuré comme suit : une discussion de la littérature est présentée dans la section 2. Dans la section 3, nous présentons la vision de notre approche puis dans la section 4, nous présentons notre méthodologie qui vise à améliorer la performance des requêtes défavorables (ad-hoc et non fréquentes). 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 État de l'Art

Cette section a pour objectif de présenter brièvement les travaux relatifs à la conception d'un entrepôt de données parallèle, la réutilisation des résultats intermédiaires et les systèmes autonomes.

Conception des entrepôts de données parallèles. La conception d'un entrepôt de données parallèle est liée principalement au problème de placement de données qui étudie comment trouver la meilleure distribution de données sur une plate-forme parallèle. Le schéma de placement de données à un impact sur plusieurs mesures de performance tels que la tolérance aux pannes, la capacité de stockage et le temps de réponse. Ainsi, une bonne stratégie de placement de données doit trouver un compromis entre cet ensemble de critères contradictoires. La littérature fait état d'un nombre important de stratégies de placement de données sous différentes formalisations. De nombreux travaux, particulièrement dans le monde industriel, proposent des variantes des stratégies de partitionnement de données conventionnel (circulaire, par intervalle et par hachage). Cependant, le problème de placement est divisé en deux sous-problèmes : le partitionnement de données et l'allocation de données. Les deux problèmes ont eu le mérite d'être largement étudiés, de manière isolée, sur toutes les générations de bases de données (Akal et al., 2002; Menon, 2005; Pavlo et al., 2012; Stöhr et al., 2000; Durand et al., 2018). Récemment, des efforts ont été fait par (Benkrid et al., 2014) pour traiter conjointement les problèmes de partitionnement et d'allocation de données pour améliorer la performance d'un EDP.

Réutilisation des résultats intermédiaires. La réutilisation des résultats intermédiaires vise à exploiter les similitudes de traitement entre plusieurs requêtes en cours d'exécution et à réduire le temps d'exécution en réutilisant les résultats intermédiaires de certaines requêtes pour optimiser d'autres requêtes (Sellis, 1988). Elle a été largement étudiée dans la littérature. Certains chercheurs ont utilisé cette idée pour recommander des vues matérialisées (Jindal et al., 2018), des stratégies d'ordonnancement de requêtes (Phan et Li, 2008) et des stratégies de gestion de cache (Gunda et al., 2010). Récemment, d'autres travaux ont proposé de prendre en considération des résultats intermédiaires pour le partitionnement de données (Du et al., 2017; Boukorca et al., 2015).

Systèmes autonomes. Un système autonome (Self-aware System) est un système capable d'adapter automatiquement son comportement aux conditions internes et externes de son environnement. IBM (Horn, 2001) définit un système autonome comme un système doté de quatre propriétés de base : auto-gestion, l'auto-configuration, l'auto-optimisation et l'auto-protection. Pour réaliser ce type de systèmes, des modèles de référence ont été proposés

(Cámara et al., 2017), nous citons principalement MAPE-K (Monitor-Analyze-Plan-Execute-Knowledge), PLA (Proactive Latency-aware Adaptation) et les systèmes multi-agents. La communauté de base de données s'est intéressée principalement à l'auto-optimisation (Gunda et al., 2010; Nehme et Bruno, 2011; Pavlo et al., 2012) .

De nos jours, *la mise en place d'un système autonome est devenue une obligation*. Cependant, les travaux existants se focalisent sur une seule propriété (optimisation). De plus, la réutilisation des résultats intermédiaires doit être prise en compte dès la conception d'un EDP et pas uniquement lors de l'optimisation. En effet, les résultats intermédiaires jouent le rôle d'une alliance entre le hors-ligne (charge de requêtes initiale) et l'en-ligne (requêtes ad-hoc).

3 Notre approche

Notre défi est de mettre en place un entrepôt de données parallèle autonome composé de deux phases. Comme illustré dans la Figure 1. La première phase est hors-ligne ; elle consiste à déterminer le meilleur schéma de placement de données. En revanche, la seconde phase est en ligne et elle représente le gestionnaire d'adaptation de notre EDP.

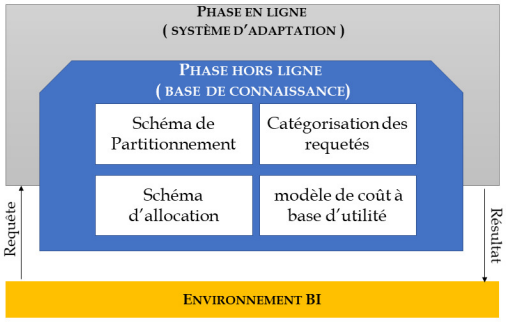

FIG. 1 – *Aperçu de notre vision*

A la base des défis décrits précédemment, nous décrivons dans cette section notre méthodologie pour réaliser un placement de données « proactif » sur une grappe de bases de données. Nous nous sommes intéressés au placement par intervalle. Ce mode de placement divise l'entrepôt de données en N fragments et les alloue d'une manière égale sur les nœuds de la grappe de bases de données. La conception du placement par intervalle peut être définie comme suit : *Étant donné un schéma en étoile d'un entrepôt de données $\mathcal{DWS}$, une charge de requêtes de jointure en étoile $\mathcal{Q}$ et une grappe de bases de données DBC, la sélection de la conception la plus appropriée pour $\mathcal{DWS}$ consiste à trouver le meilleur schéma de partitionnement de la charge $\mathcal{Q}$, soit $\mathcal{Q} = \{Q_1, Q_2, \ldots, Q_k\}$ le schéma de partitionnement obtenu, et également fragmenter la table de faits F de $\mathcal{DWS}$ en NF fragments et de les allouer sur les différents nœuds de la grappe de bases de données $\mathcal{DBC}$ pour réduire le coût d'exécution de chacune des requêtes de $Q_i (1 \leq i \leq k)$ sur $\mathcal{DBC}$ et satisfaire les contraintes de maintenance.*

La figure 2 montre l'architecture générale de notre approche composée principalement de trois composants clés :

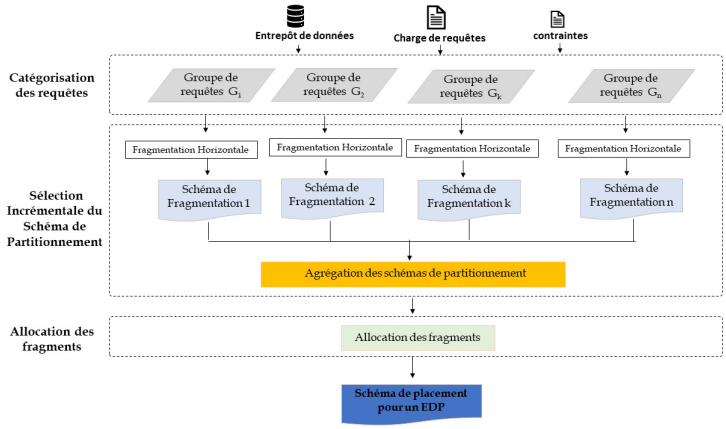

FIG. 2 – *Aperçu de notre approche hors-ligne*

Le principe général de notre méthodologie, nommée "EDP hors ligne Proactif" est de diviser la charge de requêtes $\mathcal{Q}$ en un ensemble de groupes homogènes disjoints. Pour chaque élément de $\mathcal{Q}$, un schéma de fragmentation est généré. Une fois que tous les schémas de partitionnement ont été générés, une agrégation est faite pour obtenir un schéma de partitionnement unique. Soulignons le fait que l'étape d'agrégation est primordiale pour diminuer le nombre de fragments. Cette phase cherche la meilleure manière de combiner les schémas de fragmentation générés pour obtenir un schéma de fragmentation unique et global. Enfin, les fragments ainsi générés sont alloués sur un cluster de base de données à l'aide d'un algorithme d'allocation. Chaque étape a pour objectif de maximiser la réutilisation des sous-expressions communes et de minimiser le coût de la charge de requêtes. Dans les sections suivantes, nous détaillons chaque étape de notre approche.

3.1 Catégorisation des requêtes

La catégorisation des requêtes consiste à diviser une charge de requêtes donnée en plusieurs groupes de manière à ce que les requêtes du même groupe soient liées les unes aux autres. Dans la tâche de catégorisation automatique des requêtes, il a été prouvé à travers la littérature que la sélection des caractéristiques est un élément important car il faut avoir une représentation formelle et efficace pour les requêtes en fonction des caractéristiques choisies avant d'effectuer une analyse. Jusqu'à présent, la sélection des caractéristiques pour la catégorisation des requêtes est effectuée par exploitation de trois principales caractéristiques : (1) *Lexicales* (Du et al., 2017), (2) *physiques* (Ghosh et al., 2002) et (3) *la fréquence d'arrivée* (Ma et al., 2018). Dans notre contexte, nous proposons de représenter chaque requête sous forme d'un ensemble de caractéristiques *lexico-sémantique*. Autrement dit, l'idée fondamentale de notre méthode consiste à enrichir les caractéristiques lexicales par une sémantique liée à l'exé-

cution des requêtes (physique). Le schéma 3 récapitule l'ensemble des traitements effectués sur les requêtes, en vue de leur catégorisation :

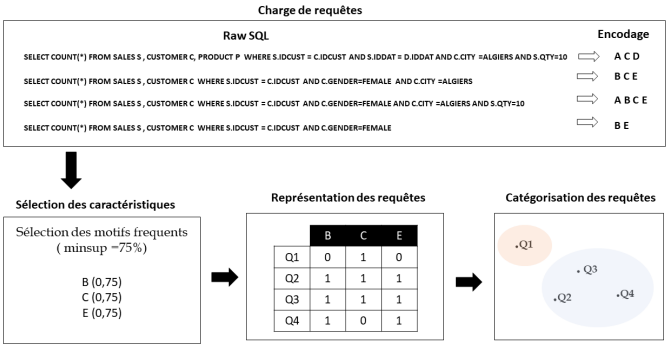

FIG. 3 – *Processus de catégorisation*

— **Sélection des caractéristiques**. Notre choix s'est intuitivement porté sur l'utilisation des résultats intermédiaires comme caractéristique de regroupement des requêtes. Ce choix se justifie par le rôle crucial que jouent les sous-expressions communes dans l'optimisation des requêtes. Plusieurs études antérieures ont examiné le problème de sélection des résultats intermédiaires (Jindal et al., 2018; Zhou et al., 2007), qui est reconnu comme un problème NP-complet. Dans ce travail, nous assimilons un résultat intermédiaire à un motif fréquent. En effet, les résultats intermédiaires représentent des sous-séquences qui apparaissent fréquemment ensemble dans la charge de requêtes. Dans notre cas d'étude, les objets sont des requêtes et les items sont les prédicats de sélection et de jointures. Plusieurs algorithmes traitent le problème de la recherche des motifs fréquents. Nous utilisons l'algorithme Apriori pour extraire les meilleurs m itemsets fréquents.

— **Représentation des requêtes**. Chaque requête est représentée par un vecteur booléen dans l'espace à m dimensions (m est le nombre des motifs fréquents sélectionnés). Le vecteur représentant une requête prend la forme d'un vecteur de 0 et de 1 où, pour chaque résultat intermédiaire (motif fréquent) sélectionné m_j, le poids lui correspondant $w_{ij} = 1$ si la requête Q_j utilise m_i, sinon $w_{ij} = 0$.

— **Création des groupes de requêtes**. La famille d'algorithmes d'apprentissage non supervisé automatique comporte un large ensemble d'exemples qui varient en implémentation et en performances en fonction du contexte de leur utilisation. Nous citons, à titre d'exemple, K-means, Mean-Shift, Hierarchical Clustering, DBSCAN. Ce dernier nous intéresse plus particulièrement car il n'est pas nécessaire de spécifier le nombre de clusters au préalable.

3.2 Sélection Incrémentale du Schéma de Partitionnement

L'idée principale de notre approche de partitionnement nommée « UtilitéFrag » est de construire plusieurs schémas de partitionnement de données et de les fusionner pour obtenir

un schéma de partitionnement des données global plus précis et plus dynamique.

3.2.1 Génération des Schéma de Partitionnement Candidats.

Pour chaque groupe de requêtes, nous appliquons un algorithme de partitionnement et nous obtenons ainsi un ensemble d'attributs partitionnées. Le partitionnement d'un attribut est représenté par un ensemble de sous-domaines. Il est à noter qu'il est tout à fait possible d'utiliser plusieurs algorithmes de partitionnement de données pour sélectionner les schémas de fragmentation de tous les groupes de requêtes. Nous utilisons l'approche proposée par (Boukorca et al., 2015) qui est en accord avec notre stratégie qui vise la maximisation de la réutilisation des résultats intermédiaires.

3.2.2 Sélection d'un Schéma de Partitionnement Global global.

Une fois que les schémas de partitionnement des groupes de requêtes ont été générés, leurs sorties doivent être combinées dans un schéma de partitionnement unique. La sélection d'un schéma de partitionnement global est très importante, une simple fusion des schémas de partitionnement générés produit un grand nombre de fragments. À cette fin, nous formalisons ce problème comme un *Set-Union Knapsack Problem* (Goldschmidt et al., 1994) :

Considérons

— un ensemble des sous-domaines $S = \{x_{11}, x_{12}, \ldots, x_{1n}, \ldots, x_{L1}, x_{L2}, \ldots, x_{Ln}\}$, tel que x_{ij} représente un sous-domaine de l'attribut $A_j (j \leq L)$ et l'union $\cup_{(u \leq n)} x_{uj}$ représente le domaine de l'attribut A_j. Chaque x_{ij} est caractérisé par un seuil w_i,

— un seuil de fragmentation W qui présente le nombre de fragments autorisées par le concepteur

— un seuil d qui présente l'utilité minimale souhaitée

— une collection $SF = \{S_1, S_2, ..., S_m\}$ tel que $S_i \subseteq S$, chaque S_i représente un schéma de fragmentation avec une utilité de p_i

Le SUKP implique la recherche d'une sous-collection $SF^* = \{S_{i1}, S_{i2}, ..., S_{it}\}$ de SF $SF^* \subseteq SF$ telle que

$$P(SF^*) = max \sum_{j=1}^{t} p_{ij} \geq d \quad (t \leq m) \tag{1}$$

$$s.t. W(SF^*) = \prod_{j=1}^{t} w_j \leq W \quad (t \leq m) \tag{2}$$

Ce problème a été montré comme un problème NP-Complet (Goldschmidt et al., 1994). Pour produire une solution quasi-optimale pour notre problème, nous avons opté pour la proposition d'un algorithme glouton (greedy algorithm) qui génère une solution réalisable.

3.2.3 Initialisation

Cette étape consiste, pour chaque attribut de partitionnement A_i, à extraire l'ensemble de sous-domaines générés par les schémas de partitionnement puis, afin de réduire la dimension de l'espace de recherche, ne garder que ceux qui se produisent fréquemment ensemble. Nous

exploitons les méthodes de recherche des motifs fréquents fermés (nous utilisons l'algorithme Close) pour générer l'ensemble de sous-domaines candidats pour chaque attribut de fragmentation.

3.2.4 phase d'exploration

L'objectif principal de la phase d'exploration est de rechercher les sous-domaines dotés d'une *"utilité" élevée*. L'utilité est un indicateur important pour mesurer la qualité de service, nous avons défini la mesure de l'utilité comme le profit (nombre des entrées sorties) généré à l'aide du partitionnement d'attributs. À cette fin, nous proposons un schéma de sélection de partitionnement d'attributs incrémentale en décomposant l'ensemble de données généré en k problèmes dépendants.

Comme la stratégie de sélection incrémentale est sensible à l'ordre dans lequel les instances de données sont présentées à l'algorithme, nous trions tout d'abord les attributs de partitionnement en fonction de leur fréquence d'apparition dans les schémas de fragmentation générés. Le schéma de partitionnement des attributs est mesuré et les sous-domaines qui attribuent l'utilité maximale sont maintenus. L'idée est de sélectionner les sous-domaines en fonction du partitionnement d'attribut le plus important (le premier est noté $Ap1$). Ensuite, les sous-domaines qui marquent la valeur maximale d'utilité sont choisis pour le second attribut $Ap2$. Le deuxième attribut doit maximiser le profit de l'ensemble d'attributs $Ap1, Ap2$ et ainsi de suite jusqu'à ce que les $c$fragments soient choisis. Certains groupes de requêtes sont appelés *victimes* car ils ne bénéficieront pas du nouveau schéma. Dans le cas du possible, nous les traitons de manière individuelle.

3.2.5 Évaluation

Notre objectif consiste à maximiser les profits en fonction d'un ensemble de contraintes. Principalement, pour chaque groupe de requêtes Q_i ($1 \leq i \leq k$), nous estimons le nombre d'entrées/sorties (noté $InitialI(O_i)$) nécessaires à l'exécution de leurs requêtes. Ensuite, nous évaluons le nombre d'entrées/sorties (noté $CurrentI(O_i)$) du schéma de partitionnement global et nous calculons l'utilité p_i :

$$p_i = InitialI(O_i) - CurrentI(O_i) \ \ \forall i 1 \leq i \leq k \tag{3}$$

L'utilité du schéma de partitionnement global est la somme des utilités des groupes de requêtes. Cette utilité $Utility$ doit dépasser un seuil d fixé par le concepteur

$$Utility = \sum_{i \in k} p_i \tag{4}$$

3.3 Génération du schéma d'allocation de Fragments

Le problème d'allocation de données sur une grappe de bases de données peut être formalisé sous la forme d'un *problème de classification* (Benkrid et al., 2014). Pour apporter une solution à ce problème, nous proposons une procédure d'allocation visant l'amélioration du

parallélisme inter-requêtes. L'idée de base consiste à former, à partir des fragments non étiquetés, M groupes qui soient les plus homogènes et la plus corrélés. Les étapes de notre procédure proposée sont les suivantes.

— **Représentation des fragments dans** $\Re^k$. Chaque fragment de l'ensemble $\mathcal{F}$ est représenté par un ensemble de k caractéristiques $\mathcal{A} = \{A_1, A_2, \ldots, A_k\}$, ces caractéristiques sont les requêtes de la charge de requêtes initiale Q et la valeur de chaque $A_t(F_i, Q_j)$ $(1 \leq t \leq k)$ est égale à

$$A_t(F_i, Q_j) = \begin{cases} 1 & Q_j \ est \ en \ matching \ total \ avec \ F_i \\ idxjoin & Q_j \ est \ en \ matching \ partiel \ avec \ F_i \\ 0 & sinon \end{cases}$$

— **Construction de la Matrice d'Appartenance des Fragments**. Le résultat délivré par cette étape est une matrice $MAF(N \times M)$ des degrés d'appartenance, où N est le nombre de fragments et M le nombre de nœuds (nombre de classes à obtenir). Pour affecter les fragments sur les nœuds de la grappe de bases de données, nous utilisons le principe suivant : "*les fragments fortement et positivement corrélés appartiennent à la même classe* ". Pour cela, nous combinons une méthode de réduction de dimension (par exemple, ACP) et une méthode de classification (par exemple, K-means++).
La valeur $MAF[i][j]$, telle que $(1 \leq i \leq Net1 \leq j \leq M)$, appartient à l'intervalle [0, 1]. Cette valeur correspond à la distance entre un fragment F_i et le centroïde de la classe c_j (chaque classe représente un nœud) .

— **Construction de la matrice de placement de fragment**. Une fois MAF créée, chaque classe est allouée sur un nœud de la grappe de base de données. Si une distribution de données biaisées se présente, nous déplaçons certains fragments selon leur degré d'appartenance. Le résultat de cette phase est une matrice, nommée MP, ses lignes et ses colonnes sont associées aux N fragments et les M nœuds associés à la grappe, respectivement. Les éléments de la matrice MP sont binaires (0 ou 1). $MP[i][m] = 1$ si le fragment F_i est alloué sur le nœud N_m, sinon $MP[i][m] = 0$

4 Expérimentations

Cette section présente les résultats d'une évaluation expérimentale de l'approche proposée. Les expériences ont été menées sur une grappe de bases de données de 10 nœuds de traitement, où chaque nœud est doté d'un processeur 3, 33 GHz Intel Core $i7$, d'une mémoire principale de 16 Go, et un Microsoft serveur SQL 2016 comme SGBD. Nos algorithmes sont implémentés en langage de programmation Java. En ce qui concerne la couche de données, nous avons considéré le banc d'essai SSB avec un facteur d'échelle de 1 ($SF = 1$). Pour la charge de requêtes nous avons généré d'une manière aléatoire 1000 requêtes en fonction des 13 requêtes d'origine de SSB.

Dans la première expérience, nous étudions la performance de notre approche de catégorisation des requêtes. Pour cela, nous utilisons la méthode décrite dans la section 3.1 et nous comparons l'efficacité de nos caractéristiques lexico-sémantiques à celles des caractéristiques lexicales. La catégorisation basée sur des caractéristiques lexicales fournit 6 classes de requêtes. Cependant, la catégorisation basée sur des caractéristiques lexico-sémantiques produit 5,8 et 13 pour un minsup de $\geq 30\%$, 20% et 10% respectivement. Les résultats obtenus

sont illustrés dans la Figure 4 et ils montrent que l'augmentation du nombre des groupes de requêtes améliore considérablement les performances de la charge de travail exécutée en lot. D'après les résultats obtenus, nous observons aussi que l'utilisation des caractéristiques lexico-sémantiques surpasse d'une manière significative (33%) l'utilisation des caractéristiques lexicales. En conséquence, nous pouvons conclure qu'à travers la sémantique apportée aux caractéristiques lexicales, la réutilisation des sous-expressions communes pour l'exécution par lots est largement exploitée, ce qui améliore considérablement les performances d'une exécution de la charge de requêtes.

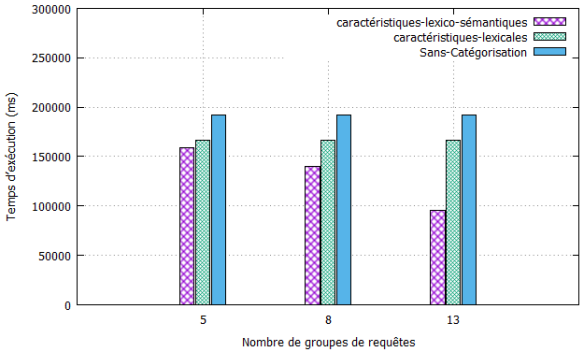

FIG. 4 – *Performance de la catégorisation basée sur des caractéristiques lexico-sémantiques.*

Dans la deuxième expérience, nous étudions l'impact de notre approche sur les performances d'un EDP. Pour cela, nous calculons le facteur de rapidité (speed-up). Pour un seuil de fragmentation de 100 et une charge de requêtes catégorisée en 5 groupes de requêtes, nous faisons varier le nombre de nœuds de 1 à 10 et pour chaque valeur ; nous calculons le speed-up. Comme le montre la Figure 5, notre approche détient un speed-up linéaire, mais ce n'est pas le speed-up idéal, cela est dû certainement à un déséquilibrage de charge entre les nœuds de traitement. La figure 6 confirme notre hypothèse.

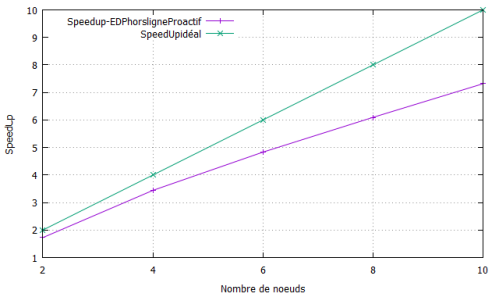

FIG. 5 – *SpeedUp*

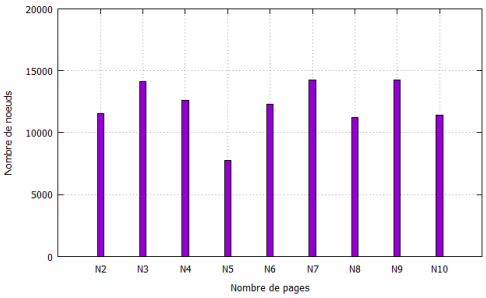

FIG. 6 – *Distribution des données*

Dans la troisième expérience, nous étudions les performances de notre approche en matière de placement de données en discutant de ce qui se passe dans la mémoire lors de l'exécution

d'un lot de requêtes. Ainsi, nous menons la même configuration que la seconde expérience et nous comparons les métriques de mémoire de notre approche proposée avec le placement circulaire et le placement par affinité. Précisément, nous exécutons des requêtes et nous analysons les métriques de la mémoire du SGBD sur le nœud ayant le maximum d'E/S. Nous concentrons notre analyse sur les indicateurs les plus importants :

— **Buffer Cache Hit Ratio** qui indique le nombre de pages constituant le cache de données ;
— **Page Read/sec** qui indique le nombre de lectures disque de pages de base de données par seconde ;
— **Page Life Expectancy** qui indique, en moyenne, combien de secondes une page de données reste dans le cache ;
— **Page Lookups/sec** qui indique la fréquence de recherche d'une page dans le cache.

Les résultats obtenus, illustrés dans la figure 7, montrent que notre approche dépasse largement le placement circulaire par 20% d'avantage

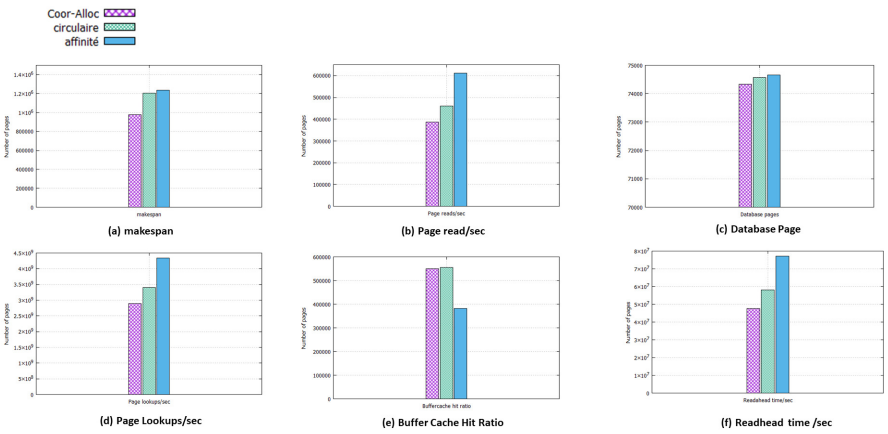

FIG. 7 – *Analyse des indicateurs de la mémoire.*

Le nombre de pages trouvées dans le cache par le placement circulaire est similaire à celui de notre politique de placement. Cependant, notre temps d'exécution (Figure 7 (a)) est inférieur à celui de l'approche basée sur le placement circulaire, ce qui signifie que notre approche traite moins de données et lit moins de pages à partir du disque comme illustré dans la Figure 7 (b)(e)(f). En outre, la métrique "Database Pages" (Figure 7 (c)) indique que le lot de requêtes généré par notre placement est plus intéressant que les autres stratégies (circulaire et affinité). En effet, un faible seuil signifie que le tampon peut ne pas avoir à libérer les pages du cache de données pour pouvoir transférer des données du disque vers le cache de données. Cet indicateur confirme l'efficacité de notre placement de données basé sur la corrélation entre les fragments. Dans un second temps, nous mesurons l'indicateur "Page Lookups/sec". Comme le montre la figure 7 (d), le nombre de requêtes effectuées par le SGBD pour trouver une page du pool de mémoire tampon n'est pas très élevé. C'est un résultat prometteur. Tous ces résultats confirment que notre politique d'allocation des données est meilleure que celle du placement circulaire et elle a donné des résultats prometteurs. De plus, l'approche par affinité donne un

résultat médiocre par rapport aux autres politiques de placement car elle ne traite pas bien les grandes charges de requêtes.

Enfin, dans la quatrième expérience, nous étudions la performance de notre approche de partitionnement basée sur l'utilité. Pour une charge de requêtes catégorisée en 5 groupes de requêtes, nous fixons le nombre de nœuds à 10 et nous faisons varier le seuil de fragmentation W dans l'intervalle $[100 - 300]$. Nous calculons le temps d'exécution nécessaire pour le traitement en lot de la charge de requêtes. Pour cela, nous comparons notre approche basée principalement sur la catégorisation et l'approche de partitionnement sans catégorisation (nous utilisons la même approche que celle utilisée pour le partitionnement de chaque classe de requêtes). Comme illustré à la Figure 8, il ressort clairement que l'augmentation du seuil de fragmentation et de catégorisation améliore généralement la performance des requêtes car en relâchant W, plus d'attributs sont utilisés pour fragmenter l'entrepôt. De plus, Lorsque le nombre de groupes de requêtes est relativement grand, le schéma de fragmentation global favorise plus de requêtes. Cela implique une utilité meilleure pour chaque groupe de requêtes. Nous soulignons que le traitement des requêtes ad-hoc est pris en compte par le modèle de catégorisation des requêtes. Plus précisément, lorsqu'une nouvelle requête arrive, nous l'affectons au meilleur groupe en calculant la similarité de la nouvelle requête avec les groupes existants et le groupe correspondant le mieux à la requête est celui qui présente la similarité la plus élevée et en conséquent il partage avec la nouvelle requête des résultats intermédiaires.

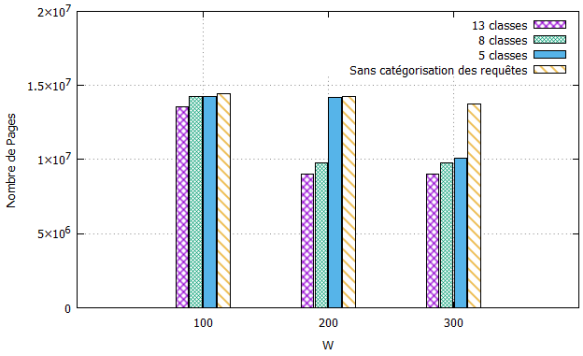

FIG. 8 – *Performance de l'approche de partitionnement UtilitéFrag*

5 Conclusion

Ce travail est une première brique permettant d'intégrer les caractéristiques des systèmes autonomes dans le monde des entrepôts de données. Cette autonomie peut toucher toutes les phases de leur cycle de vie. Nous nous sommes focalisés sur la phase de déploiement largement impactée par la nouvelle BI. Cette dernière implique des requêtes OLAP connues et des requêtes ad hoc. Cependant, les travaux existants sur le placement de données ne sont pas adaptés à de telles situations et les travaux récents proposent des solutions en mode interactif (en ligne) pour répondre efficacement aux requêtes ad hoc. Pour résoudre ce problème, nous avons proposé la mise en œuvre d'une approche proactive en mode hors connexion pour concevoir un

EDP de données sur une grappe de bases de données. Plus concrètement, notre approche vise à réduire le temps d'exécution d'une charge de requête par la maximisation de la réutilisation des résultats intermédiaires.

Ce travail ouvre plusieurs perspectives : (i) la proposition des solutions d'agrégation des schémas de partitionnement des groupes de requêtes, (ii) la définition d'un modèle de catégorisation de requêtes incrémental car la charge de requêtes peut évoluer après avoir accumulé suffisamment de nouvelles données et requêtes, et (iii) reproduire notre raisonnement sur d'autres phases de cycle de vie.

Références

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 Trans. Database Syst. 13*(3), 263–304.

Bellatreche, L. (2018). Optimization and tuning in data warehouses. In *Encyclopedia of Database Systems, Second Edition*.

Benkrid, S., L. Bellatreche, et A. Cuzzocrea (2014). A global paradigm for designing parallel relational data warehouses in distributed environments. *Trans. Large-Scale Data- and Knowledge-Centered Systems 15*, 64–101.

Benkrid, S., R. Boucenna, D. Boulegane, Y. Sennadj, C. Zakaria, et L. S. L'Hadj (2017). Vers l'amélioration du processus décisionnel par l'intégration des données sociales. In *EDA*, pp. 123–138.

Berkani, N., L. Bellatreche, S. Khouri, et C. Ordonez (2019). Value-driven approach for designing extended data warehouses. In *DOLAP*.

Boukorca, A., L. Bellatreche, et S. Benkrid (2015). HYPAD : hyper-graph-driven approach for parallel data warehouse design. In *Algorithms and Architectures for Parallel Processing - 15th International Conference, ICA3PP 2015, Zhangjiajie, China, November 18-20, 2015. Proceedings, Part IV*, pp. 770–783.

Cámara, J., K. L. Bellman, J. O. Kephart, M. Autili, N. Bencomo, A. Diaconescu, H. Giese, S. Götz, P. Inverardi, S. Kounev, et al. (2017). Self-aware computing systems : related concepts and research areas. In *Self-Aware Computing Systems*, pp. 17–49. Springer.

Chen, J. et H. Wang (2018). Guest editorial : Big data infrastructure I. *IEEE Trans. Big Data 4*(2), 148–149.

Curino, C., E. Jones, Y. Zhang, et S. Madden (2010). Schism : A workload-driven approach to database replication and partitioning. *Proc. VLDB Endow. 3*(1-2), 48–57.

Dong, L., W. Liu, R. Li, T. Zhang, et W. Zhao (2017). Replica-aware partitioning design in parallel database systems. In *Euro-Par 2017 : Parallel Processing - 23rd International Conference on Parallel and Distributed Computing, Santiago de Compostela, Spain, August 28 - September 1, 2017, Proceedings*, pp. 303–316.

Du, J., R. J. Miller, B. Glavic, et W. Tan (2017). Deepsea : Progressive workload-aware partitioning of materialized views in scalable data analytics. In *Proceedings of the 20th Inter-*

national Conference on Extending Database Technology, EDBT 2017, Venice, Italy, March 21-24, 2017., pp. 198–209.

Durand, G. C., M. Pinnecke, R. Piriyev, M. Mohsen, D. Broneske, G. Saake, M. S. Sekeran, F. Rodriguez, et L. Balami (2018). Gridformation : Towards self-driven online data partitioning using reinforcement learning. In *Proceedings of the First International Workshop on Exploiting Artificial Intelligence Techniques for Data Management*, aiDM'18, New York, NY, USA, pp. 1 :1–1 :7. ACM.

Gallinucci, E., M. Golfarelli, S. Rizzi, A. Abelló, et O. Romero (2018). Interactive multidimensional modeling of linked data for exploratory OLAP. *Inf. Syst. 77*, 86–104.

Ghosh, A., J. Parikh, V. S. Sengar, et J. R. Haritsa (2002). Plan selection based on query clustering. In *Proceedings of the 28th International Conference on Very Large Data Bases*, VLDB '02, pp. 179–190. VLDB Endowment.

Goldschmidt, O., D. Nehme, et G. Yu (1994). Note : On the set-union knapsack problem. *Naval Research Logistics (NRL) 41*(6), 833–842.

Gunda, P. K., L. Ravindranath, C. Thekkath, et and (2010). Nectar : Automatic management of data and computation in datacenters. In *Proceedings of the 9th Symposium on Operating Systems Design and Implementation (OSDI)* (Proceedings of the 9th Symposium on Operating Systems Design and Implementation (OSDI) ed.).

Horn, P. (2001). *Autonomic computing : IBM\'s Perspective on the State of Information Technology*. IBM.

Jindal, A., K. Karanasos, S. Rao, et H. Patel (2018). Selecting subexpressions to materialize at datacenter scale. *Proc. VLDB Endow. 11*(7), 800–812.

Karnagel, T., D. Habich, et W. Lehner (2017). Adaptive work placement for query processing on heterogeneous computing resources. *Proc. VLDB Endow. 10*(7), 733–744.

Ma, L., D. Van Aken, A. Hefny, G. Mezerhane, A. Pavlo, et G. J. Gordon (2018). Query-based workload forecasting for self-driving database management systems. In *Proceedings of the 2018 International Conference on Management of Data*, SIGMOD '18, New York, NY, USA, pp. 631–645. ACM.

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

Nehme, R. et N. Bruno (2011). Automated partitioning design in parallel database systems. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data*, SIGMOD '11, New York, NY, USA, pp. 1137–1148. ACM.

Ordonez, C. et L. Bellatreche (2018). A survey on parallel database systems from a storage perspective : Rows versus columns. In *Database and Expert Systems Applications Workshops*, pp. 5–20.

Ozsu, M. T. et P. Valduriez (2011). *Principles of Distributed Database Systems* (3rd ed.). Springer Publishing Company, Incorporated.

Pavlo, A., C. Curino, et S. Zdonik (2012). Skew-aware automatic database partitioning in shared-nothing, parallel oltp systems. In *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, SIGMOD '12, New York, NY, USA, pp. 61–72. ACM.

Phan, T. et W.-S. Li (2008). Load distribution of analytical query workloads for database cluster architectures. In *11th International Conference on Extending Database Technology (EDBT)*, pp. 169–180.

Quamar, A., K. A. Kumar, et A. Deshpande (2013). Sword : Scalable workload-aware data placement for transactional workloads. In *Proceedings of the 16th International Conference on Extending Database Technology*, EDBT '13, New York, NY, USA, pp. 430–441. ACM.

Rabl, T. et H.-A. Jacobsen (2017). Query centric partitioning and allocation for partially replicated database systems. In *Proceedings of the 2017 ACM International Conference on Management of Data*, SIGMOD '17, New York, NY, USA, pp. 315–330. ACM.

Sellis, T. K. (1988). Multiple-query optimization. *ACM Transactions on Database Systems 13*(1), 23–52.

Shanbhag, A., A. Jindal, S. Madden, J. Quiane, et A. J. Elmore (2017). A robust partitioning scheme for ad-hoc query workloads. In *Proceedings of the 2017 Symposium on Cloud Computing*, SoCC '17, New York, NY, USA, pp. 229–241. ACM.

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

Zhou, J., P.-A. Larson, J.-C. Freytag, et W. Lehner (2007). Efficient exploitation of similar subexpressions for query processing. In *Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data*, SIGMOD '07, New York, NY, USA, pp. 533–544. ACM.

Summary

Parallel Database Systems have become one of the essential solutions for processing massive data. The effectiveness of these systems depends heavily on the placement of data across all nodes. In today's business intelligence applications deployed on parallel systems involve two types of query loads: queries known in advance and ad-hoc queries. In reviewing the literature, we find a significant number of parallel solution designs to optimize the first type of queries. Some recent work has been proposed to optimize ad-hoc queries using machine learning. In this article, we propose a proactive approach to designing an offline parallel database. The basic idea of our approach is to use static queries as a learning base to handle ad-hoc queries. To carry out our approach, firstly, a method of categorization of queries is proposed, followed by an incremental partitioning technique. Secondly, an allocation algorithm to maximize throughput static queries and maximize reuse of intermediate results for ad-hoc and batch queries is proposed. Finally, the validation of our proposals is presented, and the results obtained are promising.