Vers des Optimiseurs Verts de Requêtes en Mode Parallèle

Simon Pierre Dembele*, Amine Roukh** Ladjel Bellatreche*

* LIAS/ISAE-ENSMA - Université de Poitiers
Futuroscope, France
(simon-pierre.dembele, bellatreche)@ensma.fr
** University of Mostaganem
Mostaganem, Algérie
roukh.amine@univ-mosta.dz

Résumé. Les récentes Conférences des Parties (connues sous le nom de COP) ont contribué au lancement des initiatives par l'ensemble des acteurs de la société autour de développement de solutions durables pour réduire le réchauffement climatique lié aux gaz à effet de serre. Dans le domaine informatique, les *data centers* sont l'un des principaux consommateurs d'énergie. Cette consommation est associée principalement aux différents composants des SGBD hébergés par ces data centers. Les optimiseurs de requêtes sont les plus énergivores, car ils sont traditionnellement conçus pour améliorer la performance de requêtes intensives comme le cas des entrepôts de données. Le peu de travaux existants intégrant l'énergie dans la conception des optimiseurs de requêtes considèrent le mode séquentiel de traitement des opérations d'un plan d'exécution d'une requête. Récemment, les SGBD centralisés ont proposé le mode parallèle impliquant plusieurs processeurs (ou coeurs) pour optimiser le temps de réponse de ces requêtes, mais négligent la dimension énergétique. Dans cet article, nous proposons une démarche de conception d'optimiseurs économiques en mode parallèle. Premièrement, nous présentons un état de l'art riche couvrant les principaux travaux sur l'intégration de l'énergie dans les SGBD. Deuxièmement, nous proposons un modèle de coût analytique estimant la consommation énergétique. Ces paramètres sont obtenus par des techniques d'apprentissage automatique. Troisièmement, nous intégrons notre modèle de coût dans le SGBD PostgreSQL 10.3. Finalement, nous validons nos propositions en utilisant les données et les requêtes du banc d'essai TPC-H.

1 Introduction

Depuis quelques années, la communauté internationale regroupant les états, les gouvernements, les associations, et les usagers s'intéresse de près aux changements climatiques en proposant des initiatives pour limiter le réchauffement climatique. En 2015 et 2016, la France et le Maroc ont organisé respectivement à Paris et à Marrakech, la Conférence mondiale Des Parties (COP21 et COP22) portant sur cette problématique, où plus de 195 pays étaient représentés.

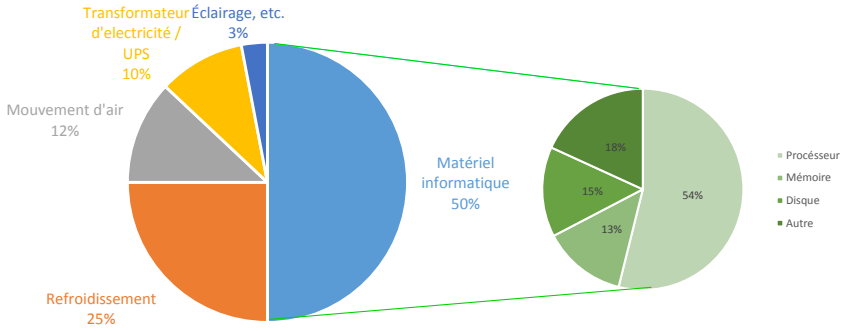

FIG. 1: La consommation d'énergie dans un centre de données.

L'un des objectifs primordiaux de cette convention est de sensibiliser les *pays pollueurs* et proposer des initiatives accompagnées par des solutions durables pour réduire le réchauffement climatique lié aux gaz à effet de serre (émissions de carbone ou CO_2). Si aucun changement de la part de la communauté internationale et les usagers ne s'opère alors les conséquences néfastes sur la planète comme : **(a)** la multiplication des vagues de chaleur, **(b)** des sécheresses et des inondations, **(c)** la fonte accélérée des glaciers auront lieu.

Le domaine informatique, avec ses composantes (matérielles et logicielles) et ses usagers participent intensivement au phénomène de réchauffement climatique. L'une des causes principales de cette situation est la demande continuelle de calcul intensif du *déluge de données*. En conséquence, les centres de données deviennent l'épine dorsale pour toute entreprise, organisme et fournisseur de stockage de données comme *Amazon*, *Facebook*, *Google*, etc. Citons l'exemple du centre de données construit par *Interxion*, l'installation consomme 64 megawatts, soit l'équivalent d'une ville de 50 000 habitants [1]. Avec la chaleur qu'il dégage, il génère 200 millions de tonnes de gaz à effet de serre Council (2014).

La consommation énergétique électrique de ces centres est répartie de la manière suivante (Figure 1) : **(i)** le matériel informatique déployé sur ces centres tels que les serveurs de traitement, **(ii)** le stockage et la communication, **(iii)** des équipements de refroidissement d'infrastructure (pour un kilowatt dépensé par un serveur, un autre kilowatt serait nécessaire pour le refroidir Liebert (2007); Tsirogiannis et al. (2010)) et **(iv)** leur fonctionnement en continu. Nous constatons que le processeur, la mémoire et le disque consomment la moitié de l'énergie totale. Les applications de bases de données hébergées sur des SGBDs sont les plus grosses consommatrices d'énergie électrique. Cela est dû au fait qu'ils effectuent des opérations coûteuses impliquant les processeurs, les mémoires et les réseaux.

Face à cette situation, la communauté de bases de données n'est pas resté indifférente. Deux articles de vision ont été publiés par des chercheurs provenant à la fois du milieu académique (Stavros Harizopoulos et al. Harizopoulos et al. (2009) de MIT) et industriel (Goetz Graefe Graefe (2008) de HP). Ces derniers ont mis l'accent sur l'intégration de l'énergie dans la conception et l'exploitation des bases de données. Les recommandations de ces auteurs coïncident avec ce que la communauté scientifique a identifié, à savoir l'utilisation des matériels et logiciels économiques Poess et Nambiar (2008). Les SGBD ont bénéficié des matériels éco-

1. http://www.interxion.com/fr/Implantations/france/paris/par7/

nomiques existants (comme les supports de stockage amovible Schall et al. (2010), les cartes graphiques Hurson et Azad (2016).

La consommation énergétique d'un SGBD est associée à ses composants (l'optimiseur de requêtes, le gestionnaire de buffer, le contrôleur de concurrence, le gestionnaire des méthodes d'accès, le gestionnaire de stockage). L'optimiseur de requêtes est le composant le plus gourmand, car il traite des requêtes exécutées sur des masses de données et doit satisfaire le besoin non fonctionnel historique des usagers à savoir la performance de ces requêtes. Ces derniers cherchent un temps de réponse raisonnable quelle que soit la complexité des requêtes et des données . Par conséquent, la conception des optimiseurs économiques devient alors un enjeu climatique. Pour concevoir des optimiseurs économiques, les concepteurs doivent prendre en compte deux aspects importants : **(i)** les paramètres pertinents et sensibles à l'énergie de tout l'environnement de la base de données. Ces paramètres couvrent plusieurs entités logicielles et matérielles tels que son schéma (par ex. la taille des tables, la longueur de tuples, etc.), la charge de requêtes (par ex. le type de requêtes, les facteurs de sélectivité des jointures et des prédicats de sélections, tailles des résultats intermédiaires, etc.), le matériel (par ex. la taille de tampon, la taille de page du disque, etc.) et la plate-forme de déploiement (par ex. RAM, la taille du disque, le nombre de nœuds d'une machine parallèle, etc.). **(ii)** Les algorithmes utilisés par ces optimiseurs de requêtes pour sélectionner les plans d'exécution de requêtes optimaux ou quasi-optimaux doivent être également conçus dans une perspective énergétique. Les optimiseurs de requêtes traditionnels utilisent massivement les modèles de coût (MdC) mathématiques pour estimer le coût d'exécution de chaque plan de requête. Ces modèles prennent en compte dans leur définition les paramètres de l'environnement de bases de données et les algorithmes de sélection des plans Ouared et al. (2016). Afin d'intégrer l'énergie dans les optimiseurs, la définition des modèles de coût estimant la consommation énergétique est primordial.

Les récents efforts sur l'intégration de la dimension énergétique dans les optimiseurs de requêtes ont considéré le mode séquentiel d'exécution de requêtes Roukh et al. (2017). Les SGBD modernes proposent le mode parallèle pour exécuter des requêtes. Ce mode permet d'accélérer le processus d'exécution de requêtes. Il est alors opportun de revisiter les travaux existants afin de les adapter pour considérer le mode parallèle. Dans cet article, nous répondons à cette problématique, en proposant un MdC estimant la consommation énergétique d'une requête exécutée par un optimiseur d'un SGBD offrant le mode parallèle. Nos propositions sont inspirées par nos travaux précédents développés pour le mode séquentiel Roukh et al. (2017).

Cet article est structuré comme suit : La section 2 présente un état de l'art sur l'intégration de l'énergie dans les bases de données. La section 3 décrit les concepts fondamentaux. La section 4 présente un audit sur le mode séquentiel et le mode parallèle d'exécution de requête. La section 5 présente notre modèle de coût mathématique ainsi que ses paramètres. Une étude expérimentale est présentée en section 6. La section 7 conclut notre article.

2 Etat de l'Art

Dans cette section, nous discutons les principaux travaux sur l'intégration de l'énergie dans les bases de données qui se déclinent en deux approches principales : (i) les approches logicielles et (ii) les approches matérielles.

Les approches matérielles concernent à la fois les dispositifs de stockage des données et les techniques de traitement de ces données. Lang et Patel (2009) ont proposé le mécanisme PVC (Processor Voltage/Frequency Control) pour équilibrer la consommation d'énergie et la performance de requêtes. Une technique similaire est utilisée dans Tu et al. (2014) qui ajuste dynamiquement le niveau DVFS du processeur en fonction des performances du SGBD ou du plan de requête choisi. Xu et al. (2013b) utilisent la technique DVFS avec un mécanisme de contrôle de rétroaction pour les charges de requêtes. Afin d'économiser l'énergie, ils gèrent le niveau DVFS en fonction du débit de travail et des caractéristiques de la charge de requêtes. Une co-ingénierie entre Oracle et Intel pour réduire les coûts d'exploitation et de répondre efficacement aux objectifs d'informatique durable (*green computing*) a donné lieu à une nouvelle version de *Oracle Exadata Database Machine* Intel et Oracle (2011). Certains travaux ont proposé de substituer les processeurs classiques par les Field-Programmable Gate Array (FPGA) et les unités de traitement graphique (GPU) Woods et al. (2014) en profitant de leur consommation énergétique modérée Rofouei et al. (2008). D'autres travaux militent au couplage CPU-GPU pour optimiser l'ensemble de requêtes de bases de données avancées Hurson et Azad (2016); Breß et al. (2014). Les Smart Solid State (SSSDs) sont également utilisés Do et al. (2013). Certes, ils ont beaucoup de potentiel, mais les capacités de calcul des appareils embarqués dans les SSSD actuels sont trop limitées pour gérer des données volumineuses.

Les approches logicielles. Les travaux existants dans cette catégorie ont suivi les recommandations de deux articles de vision sur l'incorporation de l'énergie dans les bases de données Graefe (2008); Harizopoulos et al. (2009). Ils ont recommandé de revisiter les techniques d'optimisation des requêtes, les algorithmes d'ordonnancement, la conception physique et les techniques de mise à jour de base de données en tenant compte de l'énergie. La mise en œuvre de ces recommandations passe par les chemins importants : (i) la définition des modèles de coûts pour prédire la consommation énergétique et (ii) la proposition des techniques basées sur ces modèles de coûts. Généralement, le développement de modèles de coût est une tâche difficile Ouared et al. (2016), mais dans le contexte énergétique, en plus du modèle de coût, ces paramètres doivent être également estimés Roukh et al. (2017); Binglei et al. (2017). Dans Xu et al. (2013a, 2010), les auteurs étendent le modèle de coût de PostgreSQL pour prédire la consommation d'énergie d'une seule requête exécutée d'une façon isolée (pas de parallélisme inter-requêtes). Le coût énergétique d'un plan d'exécution d'une requête prend en compte des opérations du SGBD (par ex. le coût de la puissance du processeur pour accéder à un tuple, coût du disque pour la lecture/écriture d'une page) et les différentes méthodes d'accès utilisées. Ils utilisent une technique de régression linéaire simple pour estimer les paramètres sensibles à l'énergie. Dans Kunjir et al. (2012), une recherche profonde dans la modélisation de la puissance de pic des opérations de requête est donnée. Un modèle de plans d'exécution sur la base de pipeline a été développé pour identifier les sources de consommation d'énergie de pic pour une requête isolée et recommander des plans avec une faible puissance de pic. Pour chacun de ces pipelines, une fonction mathématique est appliquée, qui prend en entrée les taux et les tailles des données circulant à travers les opérations de pipelines. En tant que sortie, une estimation de la consommation d'énergie de pic est donnée. Les auteurs ont utilisé la technique de régression multiple affine par morceaux pour construire leurs modèles de coût. Dans le même sens, le travail de Lang et al. (2011) propose un framework pour un traitement éco-énergétique des requêtes. Il étend les plans de requêtes produites par l'optimiseur de re-

quête traditionnelle avec une prédiction de la consommation d'énergie pour certains opérateurs spécifiques comme la sélection, projection et la jointure en utilisant la technique de régression linéaire. Dans Roukh et al. (2017), nous avons proposé un modèle de coût analytique considérant le pipeline lors de l'exécution de requêtes. Ces paramètres sont identifiés par un algorithme de régression polynomiale. Ce modèle de coût est utilisé dans l'optimiseur PostgreSQL dans le système EnerQuery Roukh et al. (2016) disponible sur la forge du laboratoire LIAS [2] et a guidé nos algorithmes de sélection des structures d'optimisation comme les vues matérialisées. Dans une autre ligne de recherche, les auteurs de Lang et al. (2012); Schall et Härder (2014) ont abordé le cas des bases de données parallèles et distribués sur un cluster de nœuds pour économiser l'énergie et concevoir des SGBDs avec une énergie proportionnelle. À cette fin, les auteurs ont exposé les décisions de conception qui doivent-être pris en considération, tels que la taille de cluster, l'allocation de stockage et le traitement de données dynamique sur les nœuds, et l'augmentation/réduction des nœuds du cluster en fonction des changements de la charge de travail.

3 Concepts Fondamentaux

Dans cette section, nous présentons des notions liées à l'énergie.

Définition 1 *L'énergie est une mesure de la capacité d'un système à modifier un état, à produire un travail entraînant un mouvement, un rayonnement électromagnétique ou de la chaleur.*

L'énergie peut exister sous diverses formes telles que l'énergie mécanique, l'énergie thermique, l'énergie de la lumière, l'énergie sonore, l'énergie magnétique, l'énergie électrique, l'énergie chimique et l'énergie nucléaire. Nous nous focalisons sur l'énergie électrique. Le transfert d'énergie en une seconde est défini comme la puissance électrique. Plus précisément :

Définition 2 *La puissance est la quantité d'énergie d'un système par unité de temps, ou la vitesse à laquelle un travail est fourni.*

L'énergie est généralement mesurée en *Joules* tandis que la puissance est mesurée en *Watts*. Formellement, l'énergie et la puissance peuvent être définies comme suit :

$$P = \frac{W}{t} \tag{1}$$

$$E = P \times t \tag{2}$$

où P, t, W et E représentent respectivement, une puissance, une période de temps, le travail total effectué dans ce laps de temps, et de l'énergie. Ces concepts de puissance, travail et énergie sont utilisés différemment dans divers contextes. Dans le contexte de la technologie d'information, le travail implique des activités associées à l'exécution de programmes (par exemple : addition, soustraction ou opération dans la mémoire), la puissance est le taux pour lequel l'ordinateur consomme de l'énergie électrique (ou la dissipe sous forme de chaleur) et l'énergie est l'énergie électrique totale que consomme l'ordinateur (ou se dissipe en chaleur) au fil du temps Venkatachalam et Franz (2005).

2. `https://forge.lias-lab.fr/projects/ecoprod`

Cependant, l'énergie peut être réduite soit par la minimisation du laps du temps ou de la puissance consommée. En général, la consommation de la puissance électrique d'un système donné peut-être divisée en deux parties :

1. *Puissance de base* : La consommation de la puissance lorsque le système est inactif. Cela inclut la consommation des ventilateurs, processeur, mémoire, périphériques d'E/S et les autres composants de la carte mère dans un état d'inactivité.
2. *Puissance active* : La consommation de la puissance lors de l'exécution d'une charge de travail. Les composants qui influencent sur la puissance active sont déterminés par le type de charge de travail qui s'exécute sur la machine, et la façon avec laquelle elle utilise le processeur, la mémoire et les dispositifs d'E/S.

Deux concepts de puissance électrique doivent être pris en considération lors de l'évaluation de l'utilisation de l'énergie dans un système : *la puissance moyenne* consommée au cours de l'exécution de la charge de requête, et *le pic* représentant la puissance maximale (*peak power*). L'énergie est une grandeur physique dépendante du temps, et de la puissance consommée en une seconde notée par la formule suivante :

$$Energie = puissance * temps \tag{3}$$

L'efficacité énergétique (EE) exprime l'utilisation optimale de l'énergie pour offrir le même service. Elle est exprimée par

$$EE = \frac{Travail}{Energie} = \frac{Performance}{Puissance} \tag{4}$$

4 Du Mode Séquentiel au Mode Parallèle

Certes, nous avons une certaine expérience sur le développement d'optimiseurs économiques en mode séquentiel ; mais le passage au mode parallèle n'est pas intuitif. Cela nous a obligé d'expérimenter ce dernier afin d'identifier son environnement et ses paramètres pertinents.

Rappelons qu'une requête exprimée dans un langage de requêtes de haut niveau comme SQL doit d'abord être analysée syntaxiquement. L'analyseur identifie les mots clés de la requête (les mots clés SQL, les noms d'attributs et les noms de relations) qui apparaissent dans le texte de la requête, et vérifie la syntaxe de requête pour déterminer si elle est formulée selon les règles de syntaxe (règles de grammaire) du langage de requête. La requête doit également être validée en vérifiant que tous les noms d'attributs et de relations sont des noms valides et sémantiquement significatifs dans le schéma de la base de données. En utilisant des règles de transformation, une représentation interne de la requête est ensuite créée, habituellement sous la forme d'une structure de données arborescente appelée arbre algébrique. Le SGBD doit alors concevoir une stratégie d'exécution ou un plan de requête pour extraire les résultats de la requête à partir des fichiers de base de données. Une requête a de nombreuses stratégies d'exécution possibles, et le processus consistant à la stratégie appropriée pour traiter une requête est connue sous le nom d'optimisation de requêtes.

Dans un environnement mono-processeur, le plan d'exécution d'une requête est séquentiel (l'ensemble de ses opérations est traité l'une à la suite de l'autre) tout en exploitant le pipeline

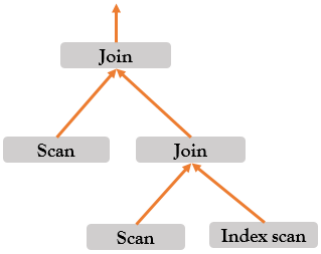

FIG. 2: Le Mode Séquentiel

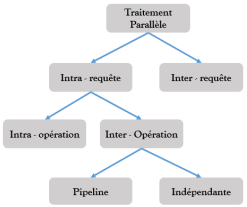

FIG. 3: Le Mode Parallèle

entre les opérations (Figure 2). Pour accélérer l'exécution d'un plan séquentiel, le recours aux techniques de parallélisation est indispensable. Deux formes principales de parallélisme existent : inter-opération et intra-opération (Figure 3).

4.1 Difficulté du parallélisme

La mise en place du parallélisme dans le contexte des bases de données a été largement étudiée. Les travaux existants ont identifié ces difficultés liées aux aspects suivants Bouganim et al. (1996) : (i) l'initialisation, (ii) le choix du nombre de processeurs (degré de parallélisme), (iii) le contrôle de l'exécution, (iv) l'équilibrage de charge, (v) les interférences.

— Initialisation : est un ensemble de procédés nécessaires avant le début du traitement parallèle. L'initialisation consiste à localiser les processeurs impliqués dans le traitement parallèle ou à la création des processus ou threads. Le temps de l'initialisation explique en grande partie les mauvais Speed-up (mesure de performance lors du traitement parallèle) Hong et Stonebraker (1991).

— Degré de parallélisme définit le nombre de nœuds (processeurs ou processus) utilisé pour exécuter un fragment du plan. L'augmentation de ce nombre complique la gestion des interférences en diminuant les performances. La diminution de ce nombre entraine également in-exploitation des ressources donc une perte de performance Bouganim et al. (1996).

— Équilibrage de charges : afin d'obtenir de bonnes performances, il est nécessaire que les opérations soient reparties équitablement entre les différents nœuds (processus ou threads). Dans le cas contraire la mesure de performance du Speed-up sera affectée Bouganim et al. (1996).

4.2 Le Cas de PostgreSQL

La parallélisation pour le cas du PostgreSQL intègre un noeud Gather ou Gather merge [3]. Un Gather (ou Gather merge) possède un ensemble de nœud fils qui représente la portion du plan parallélisé ; il peut être placé à l'intérieur du plan ou au début du plan. La figure 4 illustre le plan parallélisé de la requête suivante :

```
select sum(achats)
```

3. https://www.postgresql.org/docs/10/static/how-parallel-query-works.html

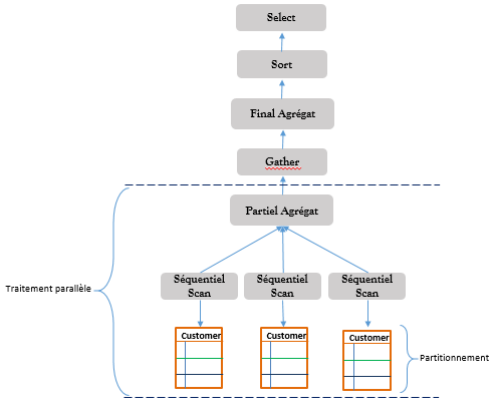

FIG. 4: Un Plan Parallèle

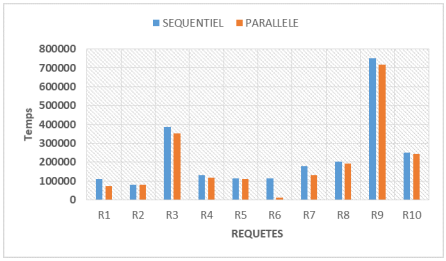

FIG. 5: Temps d'exécution du mode séquentiel et le mode parallèle

```
from customers
```

Dans ce traitement parallèle, le degré de parallélisme est trois. En conséquence, trois nœuds (processeurs ou processus) parcourent la table chacun sur une portion en adoptant l'une des techniques de partitionnement. Chaque nœud fait une agrégation partielle dont le résultat est collecté par le *gather* pour réaliser l'agrégation finale.

4.3 Prise en Main de Mode Parallèle

Dans cette section, nous évaluons la consommation énergétique des modes séquentiel et parallèle. Pour ce faire, nous utilisons le banc d'essai TPC-h [4] avec un facteur d'échelle de 10 Go en utilisant le SGBD PostgreSQL (version 10.3). Nous utilisons le power mètre pour mesurer l'énergie moyenne consommée lors de l'exécution de la requête. Nous avons fixé le degré de parallélisme à deux. Figure 5 le temps de l'exécution de 10 requêtes de notre banc d'essai.

Les figures 6, 7 répertorient l'énergie moyenne calculée par le temps d'exécution multiplié par la puissance moyenne lors de l'exécution des requêtes et l'énergie maximale ou (pic) calculée par la puissance maximale multiplié par le temps pour le mode séquentiel (dénoté par $E0$) et le mode parallèle (dénoté par $E2$, avec le degré de parallélisme fixé à 2).

Les résultats obtenus montrent l'efficacité énergétique du mode parallèle.

5 Notre Modèle de Coût

Les expériences que nous avons menées montrent l'intérêt du mode parallèle lors d'exécution de requêtes. Mais cela ne permet de conclure pas que l'énergie consommée est faible car la variation de la puissance est en fonction du nombre de processus concurrents. L'observation

4. http://www.tpc.org/tpch

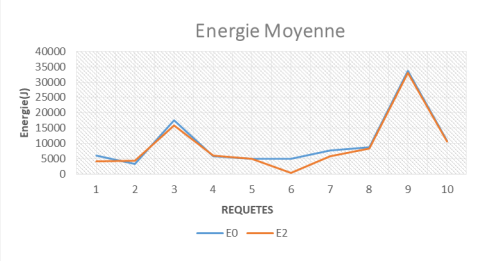

FIG. 6: Énergie moyenne

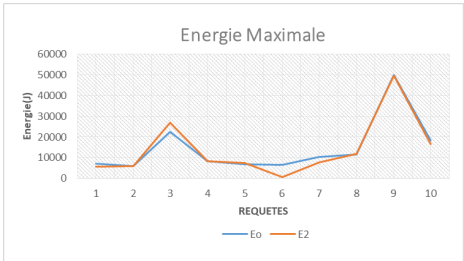

FIG. 7: Énergie maximale

nous montre que la puissance consommée n'est pas liée au nombre de processus en parallèle. Plus précisément, si nous avons une puissance P_1 watts pour un plan séquentiel avec un seul processeur, nous ne pouvons pas conclure que nous aurons $n * P_1$ watts sur un plan parallélisé, mais plutôt une valeur approximative de P_1+P_2+...+P_{n-1}+P_n<=$n * P_1$, où n représente le nombre de processus parallèles.

Pour répondre à ces questionnements, nous avons eu recours aux modèles de coût (MdC). Un MdC est un élément central dans les systèmes de base de données. Il est utilisé par l'optimiseur de requêtes pour évaluer le coût d'un plan d'exécution de la requête, selon une métrique particulière (entreées/sorties (E/S), CPU et communication).

D'une manière générale, un MdC correspondant à une métrique MET_m d'une opération d'interrogation élémentaire $ElemOp$ donné par l'équation suivante :

$$CM_{ElemOp}^{MET_m} : \mathbb{P}^n \longrightarrow Valeur\ \ de\ \ MET_m \in \mathbb{R} \tag{5}$$

où :

- $ElemOp$: est une opération de requête SQL (par exemple, restriction, projection, jointure, etc.)
- $\mathbb{P}$: est le domaine des paramètres couvrant l'ensemble de l'environnement de l'application de base de données en termes de conception, implémentation et déploiement (la couche de système d'exploitation, la couche de système de données, la couche de méthode d'accès, la couche de gestion de tampon et couche de système de stockage),
- n : est la cardinalité de l'ensemble de paramètres pris en considération par l'équation mathématique d'un MdC.
- $Valeur$: représente le résultat de sortie. Ce résultat est un nombre réel associé à la métrique utilisée.

Le MdC dédié à la consommation énergétique doit prendre en compte les trois métriques principales à savoir CPU, E/S et communication. Lors du traitement parallèle d'une requête, deux types de pipelines sont distingués : les pipelines séquentiels et les pipelines parallèles. Soit un plan d'une requête P associé à k pipelines séquentiels $\{PL_1, PL_2, \ldots PL_k\}$. Le temps d'exécution de ce plan sur un seul processeur (dénoté par $Temps(plan, 1)$) est donné par la formule suivante :

$$Temps(plan, 1) = \sum_{i=1}^{k} Temps(PL_i, 1) \tag{6}$$

Le coût énergétique de ce plan (dénoté par $Energie(plan, 1)$)) est donné par l'équation suivante :

$$Energie(plan, 1) = \sum_{i=1}^{k} Energie(PL_i, 1) \tag{7}$$

Dans le cas où un plan d'une requête est associé à k pipelines séquentiels et m parallèles, on obtient les modèles de coût suivant :

$$Temps(plan, 1) = \sum_{i=1}^{k} Temps(PL_i, 1) + \sum_{j=1}^{m} Temps(PPL_j, 1) \tag{8}$$

$$Energie(plan, 1) = \sum_{i=1}^{n} Energie(PL_i, 1) + \sum_{i=1}^{m} Energie(PPL_i, 1) \tag{9}$$

Dans un SGBD avec un mode parallèle impliquant p processeurs ou cœurs, nous obtiendrons les formules suivantes :

$$Temps(plan, p) = \sum_{i=1}^{k} Temps(PL_i, 1) + \frac{1}{P} * \sum_{j=1}^{m} Temps(PPL_j, 1) + Temps(Com, P) \tag{10}$$

$$Energie(plan, p) = \sum_{i=1}^{n} Energie(PL_i, 1) + \sum_{i=1}^{m} Energie(PPL_i, P) + Energie(Com, P) \tag{11}$$

avec $Temps(Com, P)$ et $Energie(Com, P)$ représentant respectivement le temps de communication entre les p processeurs et l'énergie consommée lors de la communication des processeurs.

Le coût énergétique impliquant des pipelines parallèles est donné par :

$$Energie(PPL_i, P) = Puissance(PPL_i, P) * \frac{1}{P} * Temps(PPL_i, 1) \tag{12}$$

La puissance dissipée lors du traitement d'une requête selon le modèle proposé par Roukh et al. (2017) sur un plan séquentiel à la suite de la combinaison des ressources principales responsables de la consommation de l'énergie est :

$$Puissance(PPL_i, 1) = \beta_{CPU} * \sum_{k=1}^{n} CT_{CPU_k} + \lambda_{IO} * \sum_{k=1}^{n} CT_{ES_k} \tag{13}$$

avec β_{CPU} : l'unité énergétique du CT_{CPU}, λ_{IO} :l'unité énergétique du CT_{ES}, n : le nombre d'opérations,CT_{CPU_k} : le nombre de coût CPU, CT_{IO_k} : le nombre d'entrées-sorties pour une opération.

Le traitement parallèle réduit le temps d'exécution des opérations. En mode parallèle, l'opération est repartie entre les différents processeurs, le temps d'exécution de l'opération est

le temps pris par le dernier processeur pour terminer sa tâche. Pour estimer la consommation énergétique en mode parallèle, nous proposons d'introduire un facteur énergétique noté fc. Ce facteur en fonction du degré de parallélisme exprime la différence énergétique de la puissance consommée lors du traitement en mode parallèle par rapport au traitement séquentiel.

$$Puissance(PPL_i, P) = (f_{c_P} + 1) * \beta_{CPU} * \sum_{k=1}^{n} CT_{CPU_k} + \lambda_{IO} * \sum_{k=1}^{n} CT_{ES_k} \quad (14)$$

Les paramètres à estimer dans notre équation seront : β_{CPU}, λ_{IO}, f_{c_P}. Pour un plan séquentiel f_{c_P}=0. f_{c_P} est un paramètre défini en fonction du nombre de processeurs. À partir de la formule (14) qui représente la puissance $Puissance(PPL_i, P)$ dissipée en P processeurs, nous avons : $Puissance(PPL_i, P) = (f_{c_P} + 1) * Puissance(PPL_i, 1)$. On peut écrire encore : $P_{m_P} = (f_{c_P} + 1) * P_0$, où P_{m_P} exprime la puissance moyenne sur p noeuds(processeurs) et P_0 décrit la puissance consommée sur le plan séquentiel sans parallélisme. Le facteur fc est alors défini par l'équation suivante :

$$f_{c_P} = \frac{P_{m_P} - P_0}{P_0} \quad (15)$$

Notre approche part de l'identification des paramètres clés qui jouent un rôle sur la consommation de l'énergie. L'identification est faite suite à l'exécution de plusieurs requêtes dans différentes bases de données de taille variable, nous varions aussi le degré de parallélisme de 0 à 4. Après la phase d'identification arrive la détermination du niveau de modélisation, quand un système de gestion de base de données reçoit un ordre de traitement d'une requête, il choisit un plan dont l'exécution se fait en segmentant le plan en un ensemble d'opérateurs physiques bloquant et non bloquant. Partant de la notion des opérateurs bloquant/non bloquant, le plan d'exécution est décomposé en un ensemble de pipelines délimités par ces opérateurs bloquants. Dans le traitement parallèle on distingue les pipelines séquentiels et les pipelines parallèles‘(comportant des opérateurs qui sont parallèlisables). Suite à cette observation le niveau de modélisation retenue dans notre approche est le Pipeline.

Partant des données du banc d'essai TPC-H sur un facteur d'échelle de 10 Go, nous avons considéré une charge de 44 requêtes caractérisées par des opérations coûteuses en ressources (CPU, ES). Pour chacune des requêtes, nous segmentons le plan en un ensemble de pipelines, au sein de chaque pipeline nous déterminons le $cout_{CPU}$ de chaque opérateurs, son degré de parallélisme et le nombre d'entrées-sorties qui constituent les paramètres d'entrées pour la construction de notre modèle.

La difficulté dans la résolution de l'équation (14) est la détermination des valeurs des paramètres β_{CPU} et λ_{IO}. Tandis que la difficulté de l'équation 15 est de trouver un facteur représentant la différence énergétique en fonction du degré de parallélisme. Pour résoudre ces équations, nous utilisons les techniques de la régression. La régression étudie la relativité entre les variables quantitatives dite expliquée et explicative. Dans nos travaux, nous utilisons la régression linéaire simple pour déterminer la valeur du facteur du paramètre f_{c_P} puis La régression polynomiale pour trouver les valeurs des paramètres β_{CPU}, λ_{IO}.

La régression linéaire simple cherche à relier les y en fonction des x par la relation $y = \beta_0 + \beta_1 x + \epsilon$ où ϵ représente la variable de bruit. L'estimation de ces coefficients peut se faire avec plusieurs méthodes. Parmi les plus utilisées on trouve la méthode des moindres carrés.

En appliquant la relation définie par la technique de la régression linéaire simple sur l'équation 15, le facteur de la consommation énergétique en fonction du degré est :

$$f_{c_P} = \alpha * degre + \beta + \epsilon \tag{16}$$

où $degre$ désigne le degré de parallélisme et ϵ est un terme d'erreur ou de perturbation. Les paramètres α et β sont des coefficients de régression qui seront estimés.

La régression polynomiale est une analyse statistique qui décrit la variation d'une variable expliquée y, en fonction d'une variable explicative x. La régression polynomiale de degré p s'écrit de la forme suivante : $y = \beta_0 + \beta_1 x + \beta_2 x^2 + \ldots + \beta_p x^p + \epsilon$ En appliquant la technique de la régression polynomiale sur l'équation 14 de notre modèle, le coût de l'énergie Puissance du pipeline deviendra :

$$\begin{aligned} P(PPL_i, P) &= \beta_1 * CT_{IO} + \beta_2 * CT_{CPU} * f_{c_P} \\ &+ \beta_3 * CT_{IO}^2 + \beta_4 * CT_{CPU}^2 * f_{c_P} \\ &+ \beta_5 * CT_{IO} * CT_{CPU} * f_{c_P} + \beta_0 + \epsilon \end{aligned} \tag{17}$$

où CT_{CPU},CT_{IO} désignent les coûts CPU et Entrées-Sorties du pipeline, respectivement. Ces coûts sont fournis par le module de statistiques SGBD, et ϵ représente les erreurs de mesure. Après avoir collecté la consommation de puissance, les coûts CPU et E/S des requêtes d'apprentissage, nous calculons les valeurs des paramètres $(\beta_1, \beta_2, \beta_3, \beta_4, \beta_5)$ en utilisant le logiciel de statistique R.

6 Expérimentation

Nous avons mené certaines expériences afin d'évaluer la qualité de notre modèle de coût. Notre environnement de test consiste en une machine Dell précision T1700 (3.40 GHz 4 coeurs 8 processeurs logiques) de 8 Go de Ram et 1TB de mémoire de stockage sur lequel est installé le système d'exploitation ubuntu 17.10 avec PostgreSQL Version 10.3. Nous avons utilisé le banc d'essai TPC-H avec différente taille de données (1 Go, 5 Go, 10 Go, 30 Go) pour déterminer la valeur du paramètre f_{c_P}. La consommation énergétique du système lors du traitement des requêtes est mesurée à l'aide du multimètre #Watts UP ? Pro ES# à une fréquence de $1HZ$. La figure 8 montre la disposition matérielle utilisée pour réaliser l'apprentissage.

Nous avons effectué l'apprentissage automatique sur 44 requêtes afin d'identifier les valeurs de nos paramètres.

Nous avons mené des expérimentations afin d'identifier le comportement du facteur f_{c_P} sur les différents facteurs d'échelle de données $1, 5, 10, 30$ GO pour des degrés de parallélisme $0, 2, 4$. A l'aide d'un script, nous exécutons l'ensemble des requêtes. Pour chaque exécution, nous : (i) vidons le cache de la base de données,(ii) vidons le cache du système d'exploitation, (iii) fixons le degré de parallélisme,(iv) enregistrons les valeurs de l'énergie à l'aide du multimètre. A la fin de la phase d'apprentissage, nous cherchons le facteur énergétique de degré $2, 4$ de chaque requête sur les différentes échelles de données à l'aide de l'équation 15. Nous calculons ensuite la moyenne du facteur énergétique des requêtes. Cette phase d'apprentissage nous a permis d'identifier la valeur de notre facteur de la consommation énergétique f_{c_P} : $f_{c_P} = 0.0552 * degre + 0.00783$.

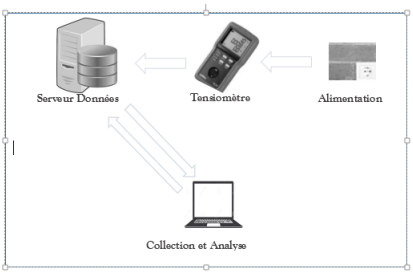

FIG. 8: *Configuration*

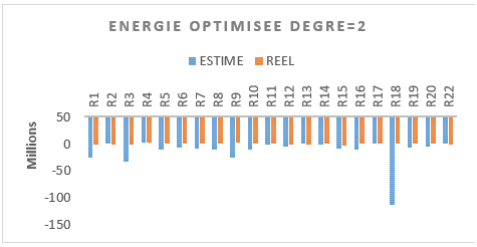

FIG. 9: Écart énergétique de degré 2

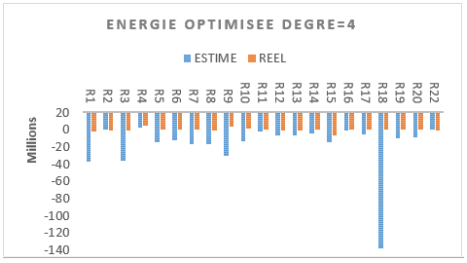

FIG. 10: Écart énergétique de degré 4

Après l'identification des paramètres du modèle de régression simple, nous estimons les paramètres de notre modèle de régression polynomiale de l'équation (14) avec le logiciel R. Les paramètres β_0, β_1, β_2, β_3, β_4, β_5 correspondent respectivement aux valeurs : $-5.783 * 10^{-6}$, $1.270 * 10^{-5}$,$2.469 * 10^{-12}$,$1.219 * 10^{-14}$,$-3.900 * 10^{-12}$.

Notre modèle de coût issu de l'apprentissage automatique est injecté dans le noyau du PostgreSQL pour la prédiction de l'énergie. Dans l'optimiseur du Postgresql, le fichier en charge du calcul de l'estimation des coûts des opérateurs est costsize.c localisé dans le répertoire path. Nous ajoutons des lignes d'instructions aux différents opérateurs pour calculer les coûts E/S et CPU. Ces valeurs sont ensuite envoyées comme paramètres à la fonction ayant pour responsabilité de calculer l'énergie.

Pour évaluer notre modèle, nous exécutons tous les 22 requêtes du banc d'essai TPC-H avec une taille d'échelle de 10 Go. Nous comparons l'énergie estimée par notre modèle de coût à l'énergie proprement consommée lors du traitement du plan choisi. Avec un degré de parallélisme fixé à 2 et 4, les figures 9 et 10 donnent l'écart énergétique observé entre l'énergie estimée et l'énergie réelle.

En observant les figures 9 et 10, l'estimation de l'écart énergétique faite par notre modèle de coût s'approche significativement de l'énergie réelle consommée. Néanmoins pour les 4 requêtes $R1, R3, R4, R9$, notre modèle fait des erreurs dans l'estimation en s'éloignant du réel. Après analyse des résultats, nous avons identifié que cette erreur vient du fait de la mauvaise estimation faite par l'optimiseur au niveau des données intermédiaires entre les pipelines.

7 Conclusion

Dans cet article, nous avons proposé une démarche de conception des optimiseurs de requêtes économiques pour faire face au déluge de données et à la complexité de requêtes qui nécessitent un traitement parallèle. Par cet article, nous voulons sensibiliser les communautés d'entrepôt de données et bases de données francophones à la dimension énergétique. Motivés par nos travaux précédents sur l'optimisation de requêtes, nous avons proposé une intégration énergétique de l'énergie dans les optimiseurs de requêtes via les modèles de coût mathématiques. Nous avons alors défini un modèle de coût dont ses paramètres sont obtenus par des techniques d'apprentissage automatique en utilisant le système R. Notre modèle a été intégré dans le SGBD PostgreSQL Version 10.3 qui offre le mode parallèle pour exécuter des requêtes. Nous avons effectué des expérimentations afin de mesurer la qualité de notre modèle de coût. Les résultats obtenus sont encourageants pour une grande partie des requêtes de banc d'essai TPC-H.

Dans les travaux futurs, nous envisageons d'affiner notre modèle de coût et d'augmenter la base d'apprentissage pour avoir une bonne estimation de ses paramètres.

Références

Binglei, G., J. Yu, B. Liao, D. Yang, et L. Lu (2017). A green framework for DBMS based on energy-aware query optimization and energy-efficient query processing. *J. Network and Computer Applications 84*, 118–130.

Bouganim, L., D. Florescu, et P. Valduriez (1996). Dynamic load balancing in hierarchical parallel database systems. In *VLDB*, pp. 436–447. Morgan Kaufmann.

Breß, S., N. Siegmund, M. Heimel, M. Saecker, T. Lauer, L. Bellatreche, et G. Saake (2014). Load-aware inter-co-processor parallelism in database query processing. *Data Knowl. Eng. 93*, 60–79.

Council, T. N. R. D. (2014). America's data centers consuming and wasting growing amounts of energy. Issue Paper, `http://www.nrdc.org/`.

Do, J., Y.-S. Kee, et al. (2013). Query processing on smart ssds : opportunities and challenges. In *ACM SIGMOD*, pp. 1221–1230. ACM.

Graefe, G. (2008). Database servers tailored to improve energy efficiency. In *EDBT*, pp. 24–28. ACM.

Harizopoulos, S., M. Shah, J. Meza, et P. Ranganathan (2009). Energy efficiency : The new holy grail of data management systems research. *arXiv preprint arXiv :0909.1784*.

Hong, W. et M. Stonebraker (1991). Optimization of parallel query execution plans in XPRS. In *PDIS*, pp. 218–225. IEEE Computer Society.

Hurson, A. et H. Azad (2016). *Energy Efficiency in Data Centers and Clouds*. Academic Press.

Intel et Oracle (2011). Oracle exadata on intel® xeon® processors : Extreme performance for enterprise computing. White paper.

Kunjir, M., P. K. Birwa, et J. R. Haritsa (2012). Peak power plays in database engines. In *EDBT*, pp. 444–455. ACM.

Lang, W., S. Harizopoulos, J. M. Patel, M. A. Shah, et D. Tsirogiannis (2012). Towards energy-efficient database cluster design. *Proceedings of the VLDB Endowment 5*(11), 1684–1695.

Lang, W., R. Kandhan, et J. M. Patel (2011). Rethinking query processing for energy efficiency : Slowing down to win the race. *IEEE Data Eng. Bull. 34*(1), 12–23.

Lang, W. et J. Patel (2009). Towards eco-friendly database management systems. *arXiv preprint arXiv :0909.1767*.

Liebert, E. (2007). Five strategies for cutting data center energy costs through enhanced cooling efficiency. White paper.

Ouared, A., Y. Ouhammou, et L. Bellatreche (2016). Costdl : A cost models description language for performance metrics in database. In *ICECCS*, pp. 187–190. IEEE Computer Society.

Poess, M. et R. O. Nambiar (2008). Energy cost, the key challenge of today's data centers : a power consumption analysis of tpc-c results. *PVLDB 1*(2), 1229–1240.

Rofouei, M., T. Stathopoulos, S. Ryffel, W. Kaiser, et M. Sarrafzadeh (2008). Energy-aware high performance computing with graphic processing units. In *Workshop on power aware computing and system*.

Roukh, A., L. Bellatreche, A. Boukorca, et S. Bouarar (2017). Eco-physic : Eco-physical design initiative for very large databases. *Information Systems*.

Roukh, A., L. Bellatreche, et C. Ordonez (2016). Enerquery : Energy-aware query processing. In *ACM CIKM*, pp. 2465–2468. ACM.

Schall, D. et T. Härder (2014). Wattdb-a journey towards energy efficiency. *Datenbank-Spektrum 14*(3), 183–198.

Schall, D., V. Hudlet, et T. Härder (2010). Enhancing energy efficiency of database applications using ssds. In *Proceedings of the Third C* Conference on Computer Science and Software Engineering*, pp. 1–9. ACM.

Tsirogiannis, D., S. Harizopoulos, et M. A. Shah (2010). Analyzing the energy efficiency of a database server. In *sigmod*, pp. 231–242.

Tu, Y.-C., X. Wang, B. Zeng, et Z. Xu (2014). A system for energy-efficient data management. *ACM SIGMOD Record 43*(1), 21–26.

Venkatachalam, V. et M. Franz (2005). Power reduction techniques for microprocessor systems. *ACM Computing Surveys (CSUR) 37*(3), 195–237.

Woods, L., Z. István, et G. Alonso (2014). Ibex : an intelligent storage engine with support for advanced sql offloading. *Proceedings of the VLDB Endowment 7*(11), 963–974.

Xu, Z., Y.-C. Tu, et X. Wang (2010). Exploring power-performance tradeoffs in database systems. In *ICDE*, pp. 485–496.

Xu, Z., Y.-C. Tu, et X. Wang (2013a). Dynamic energy estimation of query plans in database systems. In *ICDCS*, pp. 83–92. IEEE.

Xu, Z., X. Wang, et Y.-C. Tu (2013b). Power-aware throughput control for database management systems. In *ICAC*, pp. 315–324.

Summary

The recent Conferences of Parties (COP) contributed to launch initiatives to reduce the global warming bound to greenhouse gases. In the IT world, *data centers* are one of the major energy consumers. This consumption is due to the various components of the dbms (database management system). query optimizer are the most energy consumers, because they are traditionally designed to improve the performance of requests. The existing works integrate energy into the design of the queries optimizers by considering the sequential mode to process operations. Recently, dbms centralized propose the parallel mode involving several processors (or core) to optimize the response time of requests, but neglect the energy dimension. In this article, we propose an analytical model cost that estimates the energy consumption of queries executed in parallel mode. This model is integrated into the queries processing module of PostgreSQL to predict the energy consumption.