Vers le traitement et optimisation écologique des requêtes

Simon Pierre Dembele, Ladjel Bellatreche, Ordonez Carlos

Vers le traitement et optimisation écologique des requêtes

DEMBELE Simon Pierre*, Bellatreche Ladjel*, Ordonez Carlos+

Université: * LIAS/ISAE-ENSMA, + University of Houston
Ville: * Poitiers, + Houston
Pays: * France, + US
simon-pierre.dembele@ensma.fr, bellatreche@ensma.fr, carlos@Central.UH.EDU

RÉSUMÉ. De nos jours, la réduction d'énergie est devenue un problème critique et urgent pour la communauté des bases de données. De nombreuses initiatives ont été lancées sur l'efficacité énergétique pour le traitement des données massives couvrant les composants électroniques, les logiciels et les applications. Ce traitement est principalement assuré par les optimiseurs de requêtes intégrés dans ces systèmes. Dans cet article, nous affirmons que la construction d'un optimiseur de requête vert ou la révision des optimiseurs existants passent par la procédure en 4 étapes: (1) établissement d'un audit approfondi qui permet de comprendre le fonctionnement de l'optimiseur de requête, (2) identification des paramètres pertinents sensibles à l'énergie appartenant aux composants matériels et logiciels, (3) élaboration des modèles de coût mathématiques estimant l'énergie consommée lors de l'exécution d'une requête sur un SGBD et (4) estimation des valeurs des paramètres sensibles à l'énergie à l'aide d'une des techniques de l'apprentissage automatique. Pour montrer l'efficacité de cette procédure, nous l'appliquons sur deux SGBD libres avec des modes de fonctionnement différents: PostgreSQL et MonetDB et nous les comparons à l'aide du jeu de données et des requêtes du benchmark TPC-H.

ABSTRACT. Nowadays, energy reduction has now become a critical and urgent issue for the database community. A lot of initiatives have been launched on energy-efficiency for intensive-workload computation covering individual hardware components, system software, to applications. This computation is mainly ensured by query optimizers. In this paper, we claim that building from scratch a green query processors and revisiting existing ones passes through 4-steps procedure:(1) establishment of a deep audit that allows understanding the query processor functioning, (2) identification of relevant energy-sensitive parameters belonging to hardware and software components, (3) elaboration of mathematical cost models estimating consumed energy when executing a single query on a target DBMS and (4) setting of values of the energy-sensitive parameters using a *nonlinear regression technique*. To show the effectiveness of this procedure, we apply it on two open-source DBMSs with different functioning policies: PostgreSQL and MonetDB and compared them using the dataset and the workload of the TPC-H benchmark.

MOTS-CLÉS : Optimiseurs verts - audit des SGBD - technique de régression non linéaire.

KEYWORDS : Green query processing - DBMS audit - nonlinear regression technique

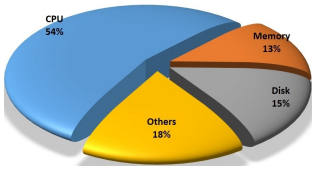


Figure 1 – Distribution de la consommation électrique des principaux composants dans l'infrastructure informatique.[17].

1. INTRODUCTION

Aujourd'hui, les fournisseurs de solutions de stockage et de traitement de données sont au cœur du nouvel ordre mondial. Ils doivent satisfaire à la fois deux contraintes Non Fonctionnelles importantes (NFR), cruciales et conflictuelles : (1) un traitement rapide du déluge de données issues des sources d'entreprise, des réseaux sociaux, de l'internet des Objets, etc. et (2) une consommation d'énergie optimale sur ces systèmes de stockage de données(SSD) pour contribuer aux objectifs écologiques fixés pour sauver notre planète. Historiquement, le premier NFR a dominé le second, car les utilisateurs / décideurs exigent un accès rapide aux données, quelle que soit la complexité des requêtes. Récemment, cette tendance a légèrement changé, depuis que les gouvernements, organisations, associations, scientifiques, industriels, gens ordinaires et célèbres du monde entier ont hissé le drapeau du changement climatique. Ils s'aperçoivent que la première étape vers les économies d'énergie est de repenser nos modes de vie et de travail au moyen d'actions politiques et économiques, y compris certainement une révision du traitement du déluge de données.

En tant que chercheurs dans la communauté des bases de données, nous sommes obligés de sensibiliser autour de nous dans le milieu universitaire, industriel, estudiantin et gouvernemental en promouvant la recherche, les produits, les actions liées à l'efficience énergétique du SSD en considérant les *petites* et *grandes* initiatives. Selon les statistiques publiées par le groupe InfoTech, les équipements informatiques à eux seuls consommeraient environ 50% de l'énergie totale. La figure 1 illustre la distribution de la consommation électrique des principaux composants dans l'infrastructure informatique. Les SSD quel que soit leur type (SGBD, centres de données et machines parallèles gérant de données, etc.) ont été identifiés comme l'un des principaux composants consommateurs d'énergie. Cette consommation est associée à leurs serveurs, périphériques de stockage, réseaux et équipements d'infrastructure tels que les systèmes de refroidissement et de conditionnement d'énergie[19]. Dans cet article, nous nous concentrons sur la construction d'un optimiseur de requêtes vert –considéré comme l'un des plus importants consommateurs d'énergie du SGBD [15]. Pour ce faire, un audit approfondi est nécessaire pour dégager les paramètres sensibles à l'énergie qui seront les entrées des modèles coût mathématiques proposes. Les valeurs de certains paramètres sensibles à l'énergie ne peuvent pas être obtenues à partir du module statistique du SGBD, ils doivent donc être calculés en utilisant les techniques d'apprentissage automatique.Pour montrer l'efficacité de cette procédure, nous considérons deux SGBD libres avec des modes de fonctionnement différents : $PostgreSQL$ et $MonetDB$.

2. Notions Fondamentales

Dans cette section, nous donnons quelques définitions qui faciliteront la lecture et la compréhension de notre proposition.

2.1. Énergie comme NFR

– ***Energie*(E)** et ***Puissance*(P)** : est une mesure (en Joules) de la capacité à faire un travail. La puissance définit la quantité d'énergie consommée par unité de temps par le système. L'unité de puissance est le Watt. *Formellement, l'énergie et la puissance peuvent être définies comme suit :* $P(t) = d/dtE(t), \quad E(t) = \int_0^{t_0} p(t)dt$, où P, t et E définissent respectivement la puissance, le temps et l'énergie. La précision de la mesure de l'énergie est une question difficile c'est pourquoi dans cette étude nous ferons référence à la *puissance moyenne* représentant la puissance moyenne consommée pendant l'exécution des requêtes.

– ***Efficacité énergétique(*EE*)*** : exprime l'utilisation optimale de l'énergie pour offrir le même service. Elle s'exprime par : $EE = \frac{Energie\ Utile}{Energie\ Absorbee} = \frac{Performance}{P}$ Sur la base de l'équation ci-dessus, nous remarquons qu'il existe deux manières d'améliorer EE soit :(i) en améliorant les performances ou (ii) en réduisant la consommation d'énergie.

2.2. Mode de fonctionnement des SGBD étudiés

Afin de démontrer l'efficacité de notre procédure pour construire un optimiseur vert de traitement des requêtes, nous considérons deux SGBD : PostgreSQL et MonetDB. Les deux sont des logiciels libres, développés en langage C, offrant un mode de fonctionnement parallèle pour le traitement des requêtes mais ils diffèrent dans leurs principes de stockage (Stockage-en-ligne vs Stockage-en-colonne) et leur taux de compression.

– **PostgreSQL :** est un SGBD de stockage en ligne prenant en charge les bases de données relationnelles objet. Il utilise le modèle serveur/client et le langage de requête structuré(SQL) pour interroger ou piloter les bases de données. Il offre de nombreuses fonctionnalités avancées telles que les types définis par l'utilisateur, l'héritage de table, un mécanisme de verrouillage sophistiqué, etc. Le traitement des requêtes en mode parallèle implique plusieurs processus qui travaillent en arrière-plan. Il existe un processus principal ou maître qui gère toutes les requêtes émises par les clients connectés. Ce dernier est responsable de quatre fonctions principales : Analyse, Réécriture, Planification, Exécution des requêtes[1].

– **MonetDB :** Il a été conçu principalement pour les applications d'entrepôt de données. Comparé aux SGBD traditionnels, Il atteint des performances significatives grâce à des innovations apportées à toutes les couches du SGBD, par exemple, un modèle de stockage basé sur la fragmentation verticale (stockage en colonnes), une architecture d'exécution de requête optimisée pour les processeurs modernes, l'indexation adaptative, l'optimisation des requêtes au moment de l'exécution[11]. L'architecture de MonetDB est composée de trois couches, chacune ayant son propre système d'optimiseurs. La couche supérieure sert à fournir une interface SQL, la couche intermédiaire contient les optimiseurs pour le langage d'assemblage MonetDB (MAL), et la couche inférieure est le noyau de la base de données qui permet d'accéder aux tables d'association binaires.

Le traitement des requêtes en mode parallèle dans Monetdb génére dans une première étape un plan séquentiel puis la parallélisation est ensuite rajoutée dans la deuxième étape

1. https://www.postgresql.org/docs/manuals/

de l'optimisation. Les opérateurs MAL individuels sont marqués comme «bloquants» ou «parallélisables». Les optimiseurs modifieront le plan en divisant les colonnes des grandes tables en blocs séparés, puis les opérateurs "parallélisables" sont exécutés sur chacun des blocs, les valeurs résultantes sont fusionnées en une seule colonne avant l'exécution des opérateurs "bloquants» [16].

3. Modèle de coût énergétique

Un modèle de coût $\mathcal{CM}$ est défini au niveau physique lors du traitement des requêtes (par exemple, l'implémentation de tri fusion par l'opération de jointure). Sa métrique correspondante peut être vue comme une fonction dont les entrées incluent des paramètres appartenant à la base de données, à la requête, à la plate-forme de déploiement, aux périphériques de traitement, etc. Inspiré de [5], les principales étapes pour la mise en œuvre des optimiseurs de requêtes optimisant l'énergie sont : (1) identification des paramètres pertinents et sensibles à l'énergie appartenant aux composants matériels et logiciels, (2) élaboration des modèles coûts mathématiques estimant l'énergie consommée lors de l'exécution d'une requête sur un SGBD et (3) l'estimation des valeurs des paramètres sensibles à l'énergie en utilisant les techniques de l'apprentissage automatique.

3.1. Étape 1 : (Identification des paramètres sensibles à l'énergie)

l'identification des paramètres sensibles à l'énergie couvrent les principaux composants d'un DSS, que nous avons classés en quatre catégories. Ces catégories sont les paramètres de base de données, les paramètres de la requête, les paramètres matériels et les paramètres liés à l'architecture de déploiement.

3.2. Étape 2 : (Construction des modèles de coût)

Pour construire nos modèles, nous avons étudié les tâches de l'exécuteur pour comprendre et dégager les paramètres pertinents influençant la consommation de l'énergie lors du traitement d'une requête isolée. Pour ce faire, nous exécutons un ensemble de requêtes (simples et complexes) en mode parallèle, en fixant le degré de parallélisme à 2 sur différents facteurs d'échelles du benchmark TPC-H. Pour ce faire, nous proposons une structure nommée *précédence des données et de localisation* (DPL). Pour illustrer cette structure, considérons l'exemple suivant : `Q:SELECT * FROM A, B, C WHERE A.x=B.x and B.y=C.y` La figure 2 illustre la structure du graphe DLP de la requête ci-dessus. La provenance des blocs de données (DP) que nous appelons ici localisation est modélisée par le symbole $\triangle$ et est annotée. Pour chaque opération de la requête, nous utilisons une annotation pour spécifier le degré de parallélisme (par exemple $[.., 2]$) lorsque l'opération est parallélisable. Les blocs de données sont lus à partir du disque en mode parallèle (tables A, B et C) vers les mémoires intermédiaires. D représente les dépendances de données entre les opérateurs représentés par des arêtes de connexion. Le L désigne la localisation des blocs de données. La dépendance de précédence indique qu'un opérateur doit être terminé avant qu'un autre opérateur puisse démarrer. Les dépendances de précédence sont représentées graphiquement par une double flèche. La double flèche entre l'opérateur probe et la fusion en est un exemple.

Dans PostgreSQL, les requêtes s'exécutent en mode pipeline [18], alors que MonetDB traite les données dans l'approche *un opérateur à la fois*. Dans ce modèle de traitement,

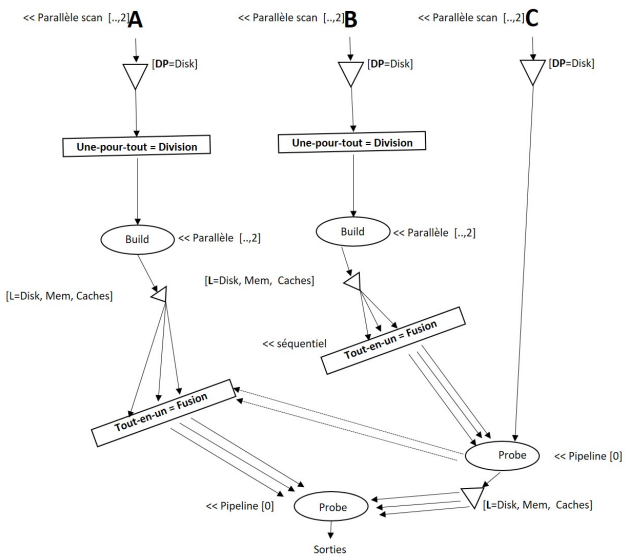


Figure 2 – Le DLP-Graphe de la requête Q

l'opérateur traite une colonne entière à la fois avant de passer à l'opérateur suivant car les résultats intermédiaires de chaque opérateur doivent être matérialisé en mémoire afin que le résultat puisse être utilisé par l'opérateur suivant. Ainsi, pour Monetdb, nous adoptons une approche de modélisation basée sur les opérations.

Pour un plan donné de la requête Q_i exécuté dans PostgreSQL, noté $PlanPost_i$ composé de k pipelines notés $\{PL_1^i, PL_2^i, \dots PL_k^i\}$, la puissance moyenne du plan peut être estimé comme suit : $Puissance(Q_i) = \frac{\sum_{j=1}^{k} Puissance(PL_j^i) * Temps(PL_j^i)}{Temps(Q_i)}$, où $Temps(Q_i)$, $Temps(PL_j^i)$ représentent respectivement le temps d'exécution de la requête Q_i et le temps d'exécution de PL_j^i.

La puissance dissipée lors du traitement de la requête est la combinaison de la consommation énergétique des principales ressources identifiées. La formule est donnée par l'équation suivante :

$$Puissance(PL_j^i) = W_{cpu} * \sum_{u=1}^{n} C_{cpu_u} + W_{mio} * \sum_{u=1}^{n} C_{mio_u} + W_{dio} * \sum_{u=1}^{n} C_{dio_u} \quad (1)$$

où W_{cpu}, W_{mio} et W_{dio} sont les paramètres du modèle. W_{cpu}, W_{mio} et W_{dio} sont les coûts unitaires correspondant à l'énergie pour une instruction, une opération de lecture/écriture sur la mémoire et une opération de lecture/écriture sur le disque respectivement. C_{cpu_u} est le nombre d'instructions exécutées par le CPU. C_{mio_u} est le nombre d'opérations de lecture ou d'écriture accessibles en mémoire. C_{dio_u} est le nombre d'opérations de lecture ou d'écriture accessibles sur le disque. Le n est le nombre d'opérateurs dans le pipeline. u est l'indice de sommation.

Pour MonetDB, le coût énergétique d'un plan de la requête (Q_i) noté par $PlanMonetDB_i$ composé de k opérations notées $\{OP_1^i, OP_2^i, \dots OP_k^i\}$ est estimé comme suit :$Puissance(Q_i) = \frac{\sum_{j=1}^{k} Puissance(OP_j^i) * Temps(OP_j^i)}{Temps(Q_i)}$, où $Temps(Q_i)$, $Temps(OP_j^i)$ représentent respecti-

vement : le temps d'exécution de la requête Q_i et le temps d'exécution de OP_j^i.

La puissance dissipée lors du traitement de la requête est la combinaison de la consommation énergétique des principales ressources identifiées. La formule est donnée par l'équation suivante :

$$Puissance(OP_j^i) = W_{cpu} * C_{cpu_j} + W_{mio} * C_{mio_j} + W_{dio} * C_{dio_j} \quad (2)$$

3.3. Étape 3 : (Apprentissage automatique)

Pour déterminer les valeurs des différents coûts unitaires dans les équations 1 et 2, nous utilisons la technique de régression polynomiale. Elle consiste à analyser une relation entre deux variables quantitatives et à l'utiliser la valeur connue de l'une pour estimer la valeur inconnue de l'autre. La régression peut être simple, multiple, polynomiale, etc. Dans notre travail, nous utilisons le degré polynomial défini sur 2 pour PostgreSQL et 3 pour MonetDB pour trouver les valeurs des paramètres de nos équations : W_{cpu}, W_{mio} et W_{dio}. Nous générons des données à différents facteurs d'échelle(SF) du benchmark TPC-H : 5 Go, 10 Go et 30 Go. Pour chaque facteur d'échelle, nous étudions et collectons les caractéristiques de quarante-quatre (44) requêtes de Selection-Projection-Jointure (SPJ) et mesurons l'énergie consommée par chacune d'entre elles à l'aide du wattmètre. Les tâches suivantes sont effectuées en amont des exécutions : **(i)** nettoyage du cache mémoire de la base de données, **(ii)** nettoyage du cache du système d'exploitation, (iii) définition de la valeur du degré de parallélisme (DoP).

4. Évaluation des SGBD sans nos modèles de coûts

Cette section présente une étude expérimentale comparant les deux contraintes Non Fonctionnelles NFR (performance de traitement et réduction de la consommation d'énergie) des SGBD sans utilisations de nos modèles. Ces expérimentations sont menées en utilisant les 22 requêtes SPJ du benchmark TPC-H. Dans toutes nos expériences, nous considérons les requêtes SPJ.

4.1. Configuration matérielle

Nos expérimentations sont menées sur un server ayant les caractéristiques suivantes : DELL Latitude E6430 avec un processeur Intel Core-i5 3340M à 2,70 GHz (1 processeur-2 cœurs-4 threads). Pour mesurer la puissance du serveur, nous utilisons un wattmètres appelé Watt UP PRO à une fréquence de 1 Hz placée entre la source d'alimentation électrique et le serveur de base de données. La topologie de notre environnement de travail est présentée dans la figure 3. Le serveur utilise Ubuntu 18.04 bionic (noyau 5.0.0-27-générique) comme système d'exploitation avec PostgreSQL version 10.10 et MonetDB version 11.33.11. Dans nos expérimentations, nous avons fixé le degré de parallélisme (DoP) à 2. Les commandes suivantes sont utilisées pour définir ce degré sur les deux SGBD : $max\_parallel\_workers\_per\_gather = \#number\#$ (PostgreSQL) et $gdk\_nr\_threads = \#number\#$ (MonetDB).

4.2. Analyse des performances

Dans cette expérimentation, nous évaluons la première contrainte NFR des deux SGBD en utilisant les facteurs d'échelles : $5Go$ et $30Go$. Le temps d'exécution de chaque

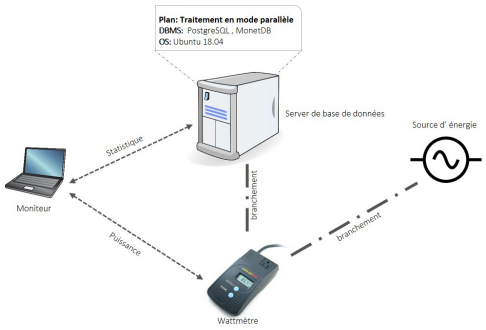
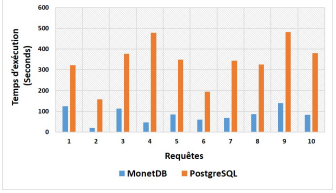


Figure 3 – Deployment of our experimental testbed.

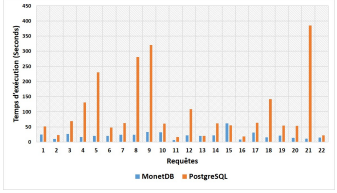
requête SPJ sur la même base de données est mesuré puis comparé. Les figures 4a et 4b résument les résultats des temps d'exécution obtenus à l'aide des requêtes SPJ du benchmark TPCH. Ces résultats révèlent que les performances de Monetdb surpassent celles obtenues par PostgreSQL pour toutes nos requêtes et sur tous les jeux de données considérés. Ces écarts de performances sont principalement dûs au fait que PostgreSQL cumule un nombre important de défauts de page lors du traitement des requêtes. Les systèmes de stockage en ligne doivent analyser et utiliser les n-tuples entiers plutôt qu'uniquement les valeurs des colonnes requises. Par conséquent, les lignes entières plus l'arborescence d'index intégrée ne peuvent pas résider longtemps dans la mémoire principale ou dans les mémoires caches à cause de la capacité réduite de ces mémoires. Ils doivent être renvoyé vers le disque, ce qui conduit à de nombreux disques E/S. Pour les systèmes orientés colonnes comme MonetDB, seules les valeurs des colonnes requises pour répondre aux requêtes sont chargées.

4.3. Analyse de la consommation d'énergie

Dans la section 4.2, nous constatons que les performances de Monetdb sont meilleures que celles de PostgreSQL. Ceci est particulièrement intéressant dans le contexte de l'énergie car l'énergie est une quantité qui dépend du temps qui s'écoule. Pour comparer la consommation d'énergie sur les deux systèmes lors de l'exécution des requêtes, nous mesurons la puissance dissipée par seconde à l'aide d'un ampèremètre nommé wattmètre. A la suite de ces mesures, nous calculons l'énergie moyenne consommée par chaque requête. Les figures 5a et 5b montrent la consommation d'énergie moyenne sur les jeux de données SF5 et SF30 respectivement. À partir des résultats présentés dans les figures 5a et 5b, l'énergie moyenne consommée par *MonetDB* pendant l'exécution des requêtes est plus économique que pour *PostgreSQL*. La forte consommation des requêtes dans Post-

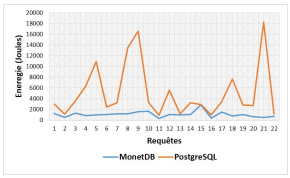


(a) TPCH SF5.

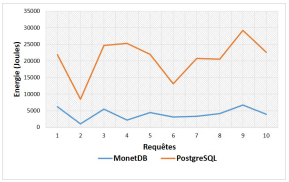


(b) TPCH SF30.

Figure 4 – Comparaison du temps d'exécution des requêtes du benchmark TPCH en mode parallèle.



(a) TPC-H SF5.



(b) TPCH SF30.

Figure 5 – Consommation d'énergie moyenne pour MonetDB par rapport à PostgreSQL.

greSQL peut s'expliquer par les caractéristiques de fonctionnement de ce type de SGBD de stockage en ligne. Il consomme plus d'énergie lors du transfert de blocs de données, car la compression des données n'est pas bien adaptée et font beaucoup d'E/S. MonetDB utilise des techniques de compression pour réduire le coût d'accès aux données et cela pèse considérablement sur les performances des requêtes pour des raisons de demande d'E/S et de défauts de page réduit(e)s. *Pour résumer, dans toutes les expérimentations que nous avons menées pour exécuter les 22 requêtes du benchmark TPCH dans les deux SGBD placent sans équivoque MonetBD plus performant que PostgreSQL pour les deux contraintes fonctionnelles NFR.*

5. Évaluation de nos SGBD verts

Dans cette section, nous évaluons nos SGBD verts obtenus en utilisant nos modèles de coût mathématiques dans le même environnement que pour les expérimentations précédentes (Sous-section 4.1).

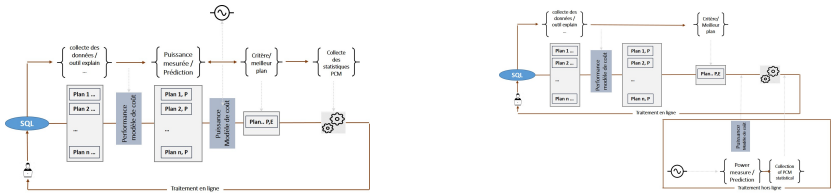
5.1. valeurs des paramètres :

Nous appliquons la régression polynomiale en utilisant le langage R version 3.5.2 pour déterminer les valeurs des paramètres de nos modèles décrits dans les équations 1 et 2. Les équations 4 et 3 décrivent nos d'équations obtenues à partir des équations 1 et 2 avec les valeurs des paramètres respectivement.

$$\begin{aligned} P(OP_j^i) = & -2,44 * 10^{-7} * C_{cpu} + 9,17 * 10^{-5} * C_{mio} + 1,99 * 10^{-6} * C_{dio} + 1,51 * 10^{-13} * C_{cpu} \\ & * C_{mio} - 7,22 * 10^{-11} * C_{mio} * C_{dio} + 1,67 * 10^{-13} * C_{cpu} * C_{dio} - 2,82 * 10^{-18} \\ & * C_{cpu} * C_{mio} * C_{dio} + 3,27 * 10^{-14} * C_{cpu}^2 - 1,46 * 10^{-9} * C_{mio}^2 - 1,56 * 10^{-12} * C_{dio}^2 \\ & - 3,74 * 10^{-19} * C_{cpu}^2 * C_{mio} + 1,07 * 10^{-16} * C_{cpu} * C_{mio}^2 + 9,67 * 10^{-21} * C_{cpu}^2 \\ & * C_{dio} + 2,93 * 10^{-16} * C_{mio}^2 * C_{dio} + 1,83 * 10^{-20} * C_{cpu} * C_{dio}^2 + 2,58 * 10^{-17} * C_{mio} \\ & * C_{dio}^2 + 3,87 * 10^{-21} * C_{cpu}^3 - 2,30 * 10^{-15} * C_{mio}^3 + 2,07 * 10^{-19} * C_{dio}^3 + 47,26 + \epsilon \end{aligned} \tag{3}$$

$$\begin{aligned} P(PL_j^i) = & 4,53 * 10^{-6} * C_{dio} - 1,46 * 10^{-6} * C_{mio} - 1,12 * 10^{-6} * C_{cpu} + 2,30 * 10^{-14} \\ & * C_{cpu} * C^{mio} - 5,88 * 10^{-12} * C_{mio} * C^{dio} + 5,97 * 10^{-15} * C_{cpu} * C^{dio} \\ & + 2,46 * 10^{-17} * C_{cpu}^2 + 8,01 * 10^{-12} * C_{mio}^2 - 9,25 * 10^{-13} * C_{dio}^2 + 48,8 + \epsilon \end{aligned} \tag{4}$$

Le coefficient de corrélation noté R mesure la force et la direction d'une relation entre deux variables. Après l'analyse des données collectées, le coefficient de corrélation donné par R est :***Multiple R-carré : 0,83, R ajusté au carré : 0,78*** pour PostgreSQL et ***Multiple R-carré : 0,58, R ajusté au carré : 0,45*** pour MonetDB. Les valeurs résiduelles des données



(a) PostgreSQL DBMS.

(b) MonetDB DBMS.

Figure 6 – Le modèle d'évaluation des optimiseurs verts.

d'apprentissage sont comprises dans l'intervalle $]-3,3[$ et $]-2,2[$ pour PostgreSQL et MonetDB respectivement.

5.2. Évaluation du plan de la requête

Un serveur de base de données reçoit les requêtes venant des utilisateurs, les compile et les exécute. L'optimiseur a pour rôle de sélectionner un plan en fonction d'un modèle de cout ayant des performances acceptables. Le modèle de coût énergétique peut être utilisé pour choisir un plan qui économise de l'énergie ou pour définir un certain seuil de compromis entre l'énergie et la performance, pour plus de détails, voir nos travaux dans [18] [6]. Dans le cas de PostgreSQL, il est facile d'intégrer notre coût énergétique dans l'optimiseur car il est conçu selon une approche basée sur les modeles coûts. La figure 6a illustre l'évaluation de notre modèle de coût sur PostgreSQL. Pour MonetDB, il n'est pas facile d'intégrer notre modèle de coût énergétique car le plan d'exécution n'est pas optimisé pour fonctionner selon une approche fondée sur les modèles de coût [9]. Pour cette raison, dans cette étude, nous exécutons notre modèle de coût en dehors des trois couches. La figure 6b illustre comment nous évaluons notre modèle sur MonetDB.

5.3. Résultats

Nous utilisons nos modèles de coût énergétique définis dans les équations 4 et 3 pour prédire la consommation énergétique de chacune des 22 requêtes du benchmark TPC-H. Nous comparons l'énergie estimée (EE) par nos modèles de coût avec l'énergie réelle (ER) consommée par le serveur de base de données lors du traitement de la requête en mode parallèle. Pour valider nos modèles énergétiques, nous utilisons la métrique Erreur d'Estimation(ERR) pour quantifier la précision de nos modèles. La métrique est définie par la formule suivante : $EER = \frac{|ER-EE|}{ER}$. ER désigne les valeurs réelles de l'énergie mesurée par le wattmètre et EE désigne l'énergie prédite par nos modèles (régression non linéaire).

Les tableaux 1 et 2 donnent les erreurs d'estimation obtenues en utilisant nos modèles de coût énergétique sur un degré de parallélisme fixé à 2 au sein de PostgreSQL et MonetDB respectivement. Veuillez remarquer qu'à partir du tableau 2, la moyenne des erreurs d'estimation est de 2,5 $\%$ et l'erreur maximale est 13 $\%$. La différence entre la puissance moyenne estimée et la puissance mesurée est très faible dans tous les cas. Dans le tableau 1, la moyenne des erreurs d'estimation est $4,1\%$ et l'erreur maximale est 10%. À partir de l'analyse des deux tableaux, nous pouvons conclure que notre modèle de coût énergétique correspond aussi bien à $MonetDB$ qu'à $PostgreSQL$ à cause des écarts d'estimation non significatifs. L'exploitation de nos modèles avec un taux d'erreur non significatif en faisant varier la valeur du critère de choix n nous permettra de mettre en évidence les économies d'énergie que nous présenterons dans nos futurs travaux.

Tableau 1 – Taux d'erreur de prédiction sur le SGBD Postgresql SF5

Requêtes	*Mesurée*	*Estimée*	EER
1	56.37	56.66	0.5%
2	50.09	48.86	2.5%
3	51.22	51.98	1.5%
4	51.21	48.51	5.6%
5	51.17	47.36	8.0%
6	51.74	50.67	2.1%
7	52.51	51.62	1.7%
8	51.13	47.96	6.6%
9	46.57	51.48	9.5%
10	50.49	54.18	6.8%
11	50.69	52.07	2.6%
12	51.74	50.76	1.9%
13	52.81	58.97	10.4%
14	52.13	51.71	0.8%
15	49.84	53.19	6.3%
16	50.32	53.19	5.4%
17	50.86	53.67	5.2%
19	52.25	52.24	0.0%
20	51.82	50.93	1.7%
22	50.64	49.40	2.5%

Lors de la collecte des données d'analyse du système, la commande *Explain Analysis* sur les requêtes 18 et 21 a pris beaucoup de temps sans donner de réponse. Nous ne les avons pas intégré dans le cas de $PostgreSQL$

Tableau 2 – Taux d'erreur de prédiction sur le SGBD MonetDB SF5

Requêtes	*Mesurée*	*Estimée*	EER
1	48.85	49.82	2.0%
2	48.10	48.30	0.4%
3	48.72	47.83	1.9%
4	48.40	48.15	0.5%
5	47.83	47.34	1.0%
6	57.71	51.50	13%
7	48.24	47.90	0.5%
8	47.98	47.96	0.01%
9	47.41	48.27	1.8%
10	50.13	50.28	0.3%
11	47.27	48.99	3.5%
12	48.62	47.16	3.1%
13	48.07	48.01	0.1%
14	48.03	46.86	2.5%
15	47.51	45.55	4.3%
16	48.80	47.71	2.3%
17	48.92	47.57	2.8%
18	48.56	47.82	1.6%
19	44.08	48.19	8.5%
20	48.40	47.3	2.3%
21	48.70	48.27	0.9%
22	48.21	47.36	1.8%

6. État de l'art

Dans cette section, nous parcourons les principales études et efforts de recherche dédiés à réduire la consommation de l'énergie lors du traitement des données. Cette étude couvre les principaux éléments ayant trait avec l'environnement des SGBD. La figure 7 répertorie les principales approches explorées pour augmenter l'EE.

– **Niveau matériel :** Les techniques appliquées dans cette approche, communément appelées gestion dynamique de l'énergie peuvent être divisées en deux catégories : désactivation dynamique des composants électroniques et la modification dynamique des performances [2]. Les études de [1] [8] exploitent l'approche fondée sur l'ajustement dynamique de la tension et de fréquence (DVFS en anglais) pour optimiser la consommation d'énergie des circuits intégrés tels que le processeur et le bus en réduisant la fréquence à laquelle ils fonctionnent. L'utilisation d'architectures non-conventionnelles telles que les processeurs graphiques (GPU en anglais), les circuits intégrés composés d'un réseau de cellules programmables (FPGA en anglais) et le traitement numérique du signal offre aujourd'hui une grande promesse pour améliorer les performances des applications et EE [4]. Les systèmes de stockage contribuent également à accroitre la consommation d'énergie, les études de [13] décrit les problèmes, défis et solutions lié(e)s à la fourniture de solutions de stockage éco-énergétiques.

– **Niveau de réseautage :** Les réseaux de communication sont responsables d'environ 2% des émissions mondiales de carbone [2]. Pour réduire cette consommation, plusieurs études se sont concentrées sur les stratégies de contrôle et d'optimisation des équipements de réseaux en adaptant les capacités du réseau et les ressources informatiques à la charge et aux demandes du trafic, tout en garantissant une qualité de service (QoS)[3].

– **Niveau logiciel :** Dans les approches logicielles, les études les plus importantes concernent (i) la définition de modèles de coûts pour prédire l'énergie et (ii) la proposition de techniques de réduction d'énergie basées sur ces coûts. Les études de [21] proposent une série de modèles de coût pour les opérateurs relationnels individuels dans un plan de requête basé sur le temps et l'énergie. Leur modèle prend le coût du processeur et le disque

2. https ://www.theclimategroup.org/sites/default/files/archive/files/Smart2020Report.pdf

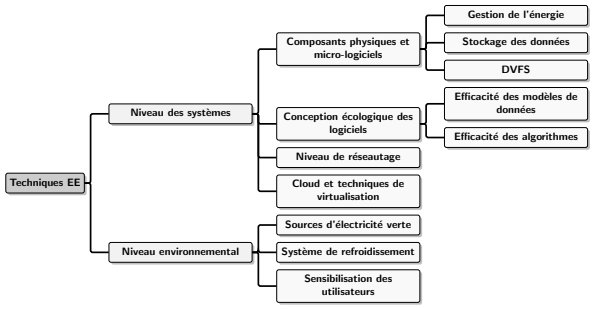


Figure 7 – Aperçu des techniques explorées pour améliorer l'EE

comme principaux composants énergivores d'un SGBD. Dans [12], une modélisation de la puissance 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 pic sur une requête isolée et de recommander des plans avec une faible puissance de pic. Dans [10], les auteurs font une étude approfondie de l'effet de trois structures de cache principales (cache de tampon, cache de dictionnaire et cache de bibliothèque) sur les coûts de traitement des requêtes. Sur cette base, ils ont proposé un modèle de coût linéaire en tenant compte de l'énergie de la mémoire centrale, du coût du processeur et du disque. Dans [6][7], les auteurs ont proposé un modèle de coût énergétique en utilisant un facteur énergétique défini en fonction du degré de parallélisme et qui exprime la différence énergétique entre la puissance consommée pendant le traitement en mode parallèle par rapport au traitement séquentiel au niveau du processeur sur les architectures multi coeurs. Un modèle de plans d'exécution sur la base de pipeline a été développé, ce plan parallèle est segmenté en deux groupes de pipelines : pipeline parallélisé et pipeline séquentiel.

7. Conclusion

Dans cet article, nous tentons de sensibiliser le monde universitaire, industriel, les sociétés informatiques, les agences de financement et les étudiants en promouvant la recherche, les produits, les actions liées aux économies d'énergie des systèmes de stockage de données en considérant des *petites* et *grandes* initiatives. Après un audit approfondi de ces SGBD, des modèles de coûts mathématiques avec des paramètres pertinents estimant la consommation d'énergie ont été proposés. La valeur de ces paramètres est obtenue par la technique de la régression non linéaire. Des expérimentations intensives ont été menées pour évaluer l'efficacité de nos résultats. Nous espérons que les lecteurs de cet article se rendront compte que la construction des optimiseurs vert de requêtes est une tâche difficile, mais réalisable pour contribuer à sauver notre planète. Une chose importante que nous devons souligner est que les éditeurs de SGBD ne sont pas obligés de reprogrammer leurs optimiseurs de requêtes, mais peuvent opter pour des solution comme les nôtres pour identifier les étapes qui contribuent à économiser l'énergie.

Références

[1] Andrea Alimonda, Andrea Acquaviva, Salvatore Carta, and Alessandro Pisano. A control theoretic approach to run-time energy optimization of pipelined processing in mpsocs. pages 876–877, 2006.

[2] Anton Beloglazov, Rajkumar Buyya, Young Choon Lee, Albert Zomaya, et al. A taxonomy and survey of energy-efficient data centers and cloud computing systems. *Advances in Computers*, 82(2) :47–111, 2011.

[3] Kashif Bilal, Saif Ur Rehman Malik, Osman Khalid, Abdul Hameed, Enrique Alvarez, Vidura Wijaysekara, Rizwana Irfan, Sarjan Shrestha, Debjyoti Dwivedy, Mazhar Ali, et al. A taxonomy and survey on green data center networks. *Future Generation Computer Systems*, 36 :189–208, 2014.

[4] Xuntao Cheng, Bingsheng He, and Chiew Tong Lau. Energy-efficient query processing on embedded CPU-GPU architectures. In *DaMoN Workshop*, pages 10 :1–10 :7, 2015.

[5] M. Dayarathna, Y. Wen, and R. Fan. Data center energy consumption modeling : A survey. *IEEE Communications Surveys Tutorials*, 18(1) :732–794, Firstquarter 2016.

[6] Simon Pierre Dembele, Ladjel Bellatreche, Carlos Ordonez, and Amine Roukh. Think big, start small : a good initiative to design green query optimizers. *Cluster Computing*, Oct 2019.

[7] Simon Pierre Dembele, Amine Roukh, and Ladjel Bellatreche. Vers des optimiseurs verts de requêtes en mode parallèle. In *EDA*, 2018.

[8] Maja Etinski, Julita Corbalán, Jesús Labarta, and Mateo Valero. Understanding the future of energy-performance trade-off via DVFS in HPC environments. *J. Parallel Distrib. Comput.*, 72(4) :579–590, 2012.

[9] Romulo Goncalves and Martin Kersten. The data cyclotron query processing scheme. *ACM Trans. Database Syst.*, 36(4) :1–35, 2011.

[10] Binglei Guo, Jiong Yu, Bin Liao, Dexian Yang, and Liang Lu. A green framework for dbms based on energy-aware query optimization and energy-efficient query processing. *Journal of Network and Computer Applications*, 84 :118–130, 2017.

[11] Stratos Idreos, Fabian Groffen, Niels Nes, Stefan Manegold, K. Sjoerd Mullender, and Martin L. Kersten. Monetdb : Two decades of research in column-oriented database architectures. *IEEE Data Eng. Bull.*, 35(1) :40–45, 2012.

[12] Mayuresh Kunjir, Puneet K Birwa, and Jayant R Haritsa. Peak power plays in database engines. pages 444–455. ACM, 2012.

[13] Pablo Llopis, Javier Garcia Blas, Florin Isaila, and Jesus Carretero. Survey of energy-efficient and power-proportional storage systems. *The Computer Journal*, 57(7) :1017–1032, 2014.

[14] Lauri Minas and Brad Ellison. *Energy efficiency for information technology : How to reduce power consumption in servers and data centers*. Intel Press, 2009.

[15] Meikel Poess and Raghunath Othayoth Nambiar. Energy cost, the key challenge of today's data centers : a power consumption analysis of tpc-c results. *PVLDB*, 1(2) :1229–1240, 2008.

[16] Mark Raasveldt and Hannes Mühleisen. Monetdblite : An embedded analytical database. *CoRR*, abs/1805.08520, 2018.

[17] Amine Roukh, Ladjel Bellatreche, and Carlos Ordonez. Enerquery : Energy-aware query processing. In *Proceedings of the 25th ACM International Conference on Information and Knowledge Management*, 10 2016.

[18] Amine Roukh, Ladjel Bellatreche, and Carlos Ordonez. Enerquery : energy-aware query processing. In *ACM CIKM*, 2016.

[19] Dimitris Tsirogiannis, Stavros Harizopoulos, and Mehul A Shah. Analyzing the energy efficiency of a database server. pages 231–242, 2010.

[20] Dimitris Tsirogiannis, Stavros Harizopoulos, and Mehul A Shah. Analyzing the energy efficiency of a database server. In *sigmod*, pages 231–242, 2010.

[21] Zichen Xu, Yi-Cheng Tu, and Xiaorui Wang. Dynamic energy estimation of query plans in database systems. In *2013 IEEE 33rd International Conference on Distributed Computing Systems*, pages 83–92. IEEE, 2013.