Towards Green Query Processing - Auditing Power Before Deploying

Simon Pierre Dembele
LIAS/ISAE-ENSMA
France

Ladjel Bellatreche
LIAS/ISAE-ENSMA
France

Carlos Ordonez
University of Houston
USA

***Abstract*—Nowadays, energy reduction has 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. Their current versions minimize inputs-outputs operations and try to exploit RAM as much as possible, by ignoring energy. A couple of studies proposed the integration of energy into query optimizers that can be classified into hardware and software solutions. Several researchers have the idea that the operating systems and firmware manage energy and put software solutions in the second plan. This does not distinguish between tasks of operating systems and DBMSs. In this paper, we claim that building from scratch a green query processors and revisiting existing ones pass 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 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.**

***Index Terms*—Query processing, DBMS audit, Machine Learning, Green computing, large databases.**

I. Introduction

In today's world, our life depends too much on computers. Therefore, we are forced to look at every way to save energy, including DBMSs. Providers of data storages and processing solutions are at the heart of the new world order. They *have to satisfy at the same time* two important, crucial and conflictual Non-Functional Requirements (NFR): **(1)** a rapid processing of the deluge of data issued by enterprise sources, social networks, Internet of Things, etc. and **(2)** an optimal energy consumption for these Data Storage Systems (DSS) to contribute as the ecological objective fixed to save our planet. Historically, the first NFR has dominated the second one, because decision-makers demand fast access to data, no matter how complex the queries are.

Recently, this tendency has slightly changed, since numerous governments, organizations, associations, scientists, industrials, ordinary and famous people around the world have raised the

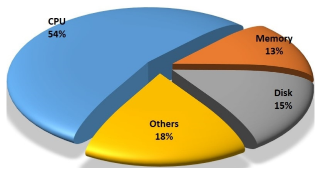


Fig. 1: Energy distribution among different components[31].

flag regarding climate change. They agree that the first step towards energy saving is to rethink our life and work styles by the means of political and economical actions including certainly a review of the process of data deluge. Consequently, the second NFR has emerged and naturally collapsed with the first one, since any IT company is seeking for data to increase its added-value. As mentioned in The Economist *"the world's most valuable resource is no longer oil, but data"*[10]. Like any oil, data pollutes as mentioned in the latest Blog entry of the Martin Tisné published on July 24, 2019: *"Data isn't the new oil, it's the new CO2"*[35]. This pollution is caused by storing and processing this data.

As database researchers, we are then obliged to sensible ourselves, academia, industry, IT companies, funding agencies, and students by promoting research, products, actions related to energy savings of the DSS by considering *small* and *big* initiatives. In addition to the DSS, the infrastructure of a company includes hardware, software, and facility service components that support the delivery of business systems and IT-enabled processes. According to the statistics published by the InfoTech group, IT equipment consumes approximately 50% of the total energy. Figure 1 illustrates the power consumption distribution of major components that consume energy in IT infrastructure. DSSs regardless of their types (DBMSs, Data Centers, and parallel database machines, etc.) have been identified as one of the major energy-consuming components. The processor consumes a major portion of energy followed by the storage device [31]. The first and major efforts in managing the energy of DSS have particularly touched Data Centers [7] since they have been pointed out by several organizations reports as one of the biggest energy consumers. Based on U.S. Environmental Protection Agency, in 2014, U.S. Data Centers consumed approximately 70 billion kilowatt-hours, totaling about 1.8% of domestic electricity consumption[34].

The principle of first tackling the biggest energy consumers is not enough to save the global warming of our planet as recommended by recent Conference Of the Parties (COP) meetings. Small initiatives may have a big impact on saving energy[8]. This principle is true in the context of DSS. It should be noticed that actually several small and medium-sized enterprises intensively own DBMSs. The integration of energy into DBMS *is beyond doubt a crucial and urgent issue*. Claremont's report on database research emphasized the importance of "*designing power-aware DBMSs that limit energy costs without sacrificing scalability*". This is also echoed in the more recent *Beckman report* on database systems, which considers "*energy constrained processing as a challenging issue in Big Data*" [1]. The integration of energy into DBMS has to concern all its components and its actors including hardware and software providers, database designers and administrators, end-users, etc.

In this paper, we focus on building a green query processor – considered as one of the most important energy consumers of the DBMS [27] and touches all actors. Traditionally, the main objective of query optimizers is the satisfaction of the first NFR. This is performed by reducing the number of inputs-outputs (IOs) operations by exploiting the RAM, by scarifying the second NFR representing the energy consumption. The 2010s were a decade that marked the beginning of the integration of energy in designing query optimizers [19], [38], where hardware and software solutions have been discussed, evaluated, analyzed and implemented [8], [14], [40]. Hardware research efforts got more attention than software ones. This is because several studies consider that the operating systems and firmware (hardware programs) manage energy and consequently save energy of query processors. This finding is *questionable* since they use techniques covering software (e.g., finding the best query plan satisfying the first NFR) and hardware (e.g., executing the appropriate algorithms associated with the selected plan on the target platform hosting the DBMS).

We claim that building green query processors passes through a deep audit that allows designers to identify relevant energy-sensitive parameters that are the entries of the mathematical cost models estimating energy when executing a query. The development of such models necessitates a deep understanding of the functioning of the target DBMS hosting the database application (e.g., query execution mode). Note that the value of some energy sensitive parameters cannot be obtained from the statistic module of the DBMS, therefore they have to be computed using a machine learning technique. To show the effectiveness of this procedure, we consider two open-source DBMSs with different functioning policies: PostgreSQL and MonetDB.

The remainder of this paper is organized as follows: In section II, we first present the fundamental notions of energy and a high-level description of PostgreSQL and MonetDB. Section III presents our mathematical cost models estimating energy and a *non-linear regression technique* to set the value of energy-sensitive parameters. A comparison in terms of two NFR of our DBMSs without our contributions is described in Section IV. Experiments of comparing these two DBMSs by considering our energy modeling is commented in Section V. Related work is presented in Section VI. Section VII concludes our paper.

II. Preliminaries

In this section, we give the background to propose green query processors.

A. Energy Efficiency as a Non Functional Requirements

Definition 1: ***Energy*** (**E**) is a measurement (in Joules) of the ability of something to do work. It comes in many forms (magnetic energy, electrical energy, chemical energy, and nuclear energy). It can be transformed from one type to another. In our study, we consider electrical energy. Energy is a physical quantity dependent on time.

Definition 2: ***Power*** (**P**) is defined to be the rate at which *work* is performed, or the derivative of *work* over time. The unit for power is the *Watts*. In electronics, power is defined as the amount of energy consumed per unit of time by the system. Work (**W**) is related to the amount of energy transferred in or from a system by a force. *Formally, energy and power can be defined as follows:*

$$P(t) = \frac{dE(t)}{dt} \tag{1}$$

$$E(t) = \int_0^{t_0} p(t)dt \tag{2}$$

where P, t, and E represent, respectively, a power, a period, and energy. Since it is hard to guarantee the accuracy of energy measure, in this paper we use the *average power* representing the average power consumed during the execution of the workload.

Definition 3: ***Energy efficiency*** *(EE)* expresses the optimal use of energy to offer the same service. It is expressed by [36]:

$$EE = \frac{Useful\ energy\ output}{Total\ energy\ input} = \frac{Performance}{P} \tag{3}$$

Based on the above equation, we remark that there are two ways to improve EE either by: (i) improving performance or (ii) reducing energy input consumption.

B. Cost Models in Database World

A cost model ($\mathcal{CM}$) is a mechanism to estimate measurable metrics associated with the quantitative QoS attributes (e.g., Inputs-Outputs, CPU and Network Transfer) [26] . It includes parameters belonging to the internal and external components of the database environment. In the context of query processors, a $\mathcal{CM}$ is defined at the query physical operation level (e.g., sort-merge implementation for join operation). Its corresponding metric can be seen as a function with inputs include parameters belonging to the database, the query, the deployed platform, the processing device(s), etc. [21].

A $\mathcal{CM}$ corresponding to a given metric MET_m of an elementary physical query operation $ElemPhyOp$, denoted by $CM^{MET_m}_{ElemPhyOp}$ may be defined as follows [26]:

$$CM^{MET_m}_{ElemPhyOp} : \mathbb{P}^n \longrightarrow \ Value\ \ of\ \ MET_m \in \mathbb{R} \tag{4}$$

where: $\mathbb{P}$: represents the set of the parameters. The parameters's values are either obtained from database statistics or using the Processor Counter Monitor (PCM)[25] or estimated by machine learning techniques. n is the cardinality of the parameters set.

C. A Deep Understanding of Functioning of our Studied DBMS

In order to demonstrate the effectiveness of our procedure to build a green query processor, we consider two DBMS: PostgreSQL and MonetDB. Both are open-source software, developed in C language, offering parallel mode for query processing. But they differ in their storage layouts (Row-Store vs. Colum-Store) and compression rate.

1) ***PostgreSQL:***

It is a row-store DBMS supporting object-relational databases. It uses the server/client model and supports the standard database languages. It offers many advanced functionalities such as user-defined types, table inheritance, sophisticated locking mechanism. The support of a parallel query involves multiple background worker processes. There is a back-end process that handles all queries issued by the connected client. This back-end consists of the following subsystems: **Parser:** it checks the query syntax expressed in a high-level query language like SQL to determine whether it is well formulated according to the grammar rules of the query language. **Analyzer:** The query must be validated by verifying that all attributes and relationship names are valid and semantically significant in the schema of the database. **Rewriter:** Using transformation rules, an internal representation of the query is then created (query tree). **Planner:** It generates the cheapest plan tree that can be executed from the query tree. **Executor:** A query has many possible execution strategies, and the selection of the best plan is usually conducted by cost model-driven strategies [11]. Figure 2 (a) summarizes the different phases of the PostgreSQL query processor.

2) ***MonetDB:***

It was designed primarily for data warehouse applications. Internally, the design, the architecture and the implementation of MonetDB reconsider all aspects and components of classical database architecture and technology by exploiting effectively the potentials of modern hardware. **Storage model:** It is a significant deviation of traditional database systems. It uses the decomposed storage model (DSM) which represents relational tables using vertical fragmentation, by storing each column in a separate $\#surrogate, value\#$ table, called binary association table (BAT). The left column (the surrogate or object-identifier (oid)) is called the head, whereas, the right column is the tail. During the query evaluation, all intermediate results are in a column format. Only just before sending the final result to the client, $N-ary$ tuples are constructed. **Query execution model:** The MonetDB kernel is an abstract machine, programmed in the MonetDB Assembly Language (MAL). The core of MAL is formed by a closed low-level two-column relational algebra on BATs. N-ary relational algebra plans are translated into two-column BAT algebra and compiled to MAL programs. These

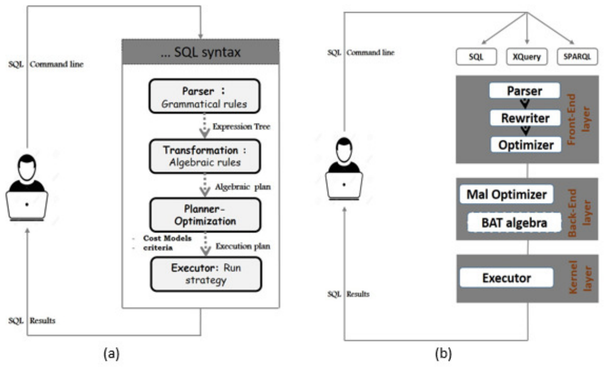


Fig. 2: The main steps of query processor in PostgreSQL and MonetDB architecture.

MAL programs are then evaluated in an operator-at-a-time manner. Figure 2 (b) shows the internal design of MonetDB. MonetDB's query processing scheme is centered around three software layers: *The top layer or front-end* provides the user-level data model and query language. The query language is first parsed into an internal representation (e.g., SQL into relational algebra), which is then optimized using domain-specific rules. *The middle layer or back-end* consists of the MAL optimizers framework and the MAL interpreter. *The bottom layer or kernel* provides BATs as MonetDB's bread-and-butter data structure, as well as the library of highly optimized implementations of the binary relational algebra operators.

For query parallel execution in MonetDB, a sequential execution plan is generated firstly and parallelization is then added in the second optimization phase. The individual MAL operators are marked as either "blocking" or "parallelizable". The optimizers will alter the plan by splitting up the columns of the largest table into separate chunks, then executing the "parallelizable" operators once on each of the chunks, and finally merging the results of these operators together into a single column before executing the "blocking" operators[28].

III. Energy Mathematical Cost Models

The previous section allows us understanding the functioning of our studied DBMS. Inspired from [7], we establish the main steps of building green query processors are: (1) identification of relevant energy-sensitive parameters belonging to hardware and software components, (2) elaboration of mathematical cost models estimating consumed energy when executing a query on a target DBMS and (3) setting of values of the energy-sensitive parameters using machine learning techniques.

a) ***Step 1:*** *(Energy-sensitive parameters identification)*

Regarding the extraction of energy-sensitive parameters, we have identified a set of parameters that cover the main components of a DSS, that we have classified them into four categories. These categories are database parameters, query parameters, hardware parameters and parameters related to the deployment architecture.

b) ***Step 2:*** *(Energy-cost model construction)*

To construct our model, we investigate the executor tasks to understand how to profile the power consumption on an individual query. To do that, we execute a set of queries (simple and complex) in parallel mode, where the degree of parallelism is set to 2 with different scale of factors of the TPC-H benchmark. When evaluating these queries, we realize the need for having a data structure capturing the different flows during the parallel query execution mode. To do so, we propose a *Data Precedence and Localization* (DPL) structure. It is like a query tree. To illustrate this structure, let us consider the following example:

```
Q: SELECT * FROM A, B, C
WHERE A.x = B.x and B.y = C.y;
```

Figure 6 shows the DLP graph structure of the above query. The data blocks provenance (DP) that we call localization is modeled by the symbol $\triangle$ and annotated. For each query operation, we use an annotation to specify the parallelism degree (e.g.[.., 2]) when the operation is parallelizable. Data blocks are read from disk in parallel mode (Table A, B, and C) to the intermediate memories. The D stands for the Data dependencies between operators represented by connecting edges. The L denotes the Localization of data blocks. Precedence dependencies stating that an operator must be terminated before another operator can start. Precedence dependencies are represented graphically by a double direct edge. The double direct edge between the probe operator and merging is an example.

In PostgreSQL, we consider that queries are executing in pipelined fashion [31], whereas MonetDB processes the data in an *operator at a time* manner. In this processing model, the operator processes the entire column at once before moving on to the next operator because the intermediate result for each operator must to be materialized into memory so this result can be used by the next operator. Thus, for MonetDB, we adopt an operations-based modeling approach.

For a given query plan of Q_i executing in PostgreSQL, denoted by $PlanPost_i$ consisting by k pipelines noted $\{PL^i_1, PL^i_2, PL^i_3, \dots PL^i_k\}$, its average power cost is estimated as follows:

$$Power(Q_i) = \frac{\sum_{j=1}^{k} Power(PL^i_j) * Time(PL^i_j)}{Time(Q_i)} \qquad (5)$$

where $Time(Q_i)$, $Time(PL^i_j)$ represent respectively, the execution time of the query Q_i and the execution time of PL^i_j. The power dissipated when processing the query is the combination of main identified resources (*CPU, Main memory and Disk*) energy consumption. We work on a single node (centralized DBMS), the communication cost is ignored(no network cost). The formula is given by the following equation:

$$Power(PL^i_j) = W_{cpu} * \sum_{u=1}^{n} C_{cpu_u} + W_{mio} * \sum_{u=1}^{n} C_{mio_u} + W_{dio} * \sum_{u=1}^{n} C_{dio_u} \qquad (6)$$

where W_{cpu}, W_{mio} and W_{dio} are the model parameters. W_{cpu}, W_{mio} and W_{dio} are unit-power cost for instructions, read/write operations on memory and read/write operations on disk respectively. The C_{cpu_u} is the number of instructions executed by CPU. C_{mio_u} is the number of read or write operations accessed on memory. C_{dio_u} is the number of read or write operations accessed on disk. The n is the number of operators in the pipeline. u is the summation index.

For MonetDB, the energy cost for a given query plan (Q_i) denoted by $PlanMonetDB_i$ consisting by k operations noted $\{OP^i_1, OP^i_2, OP^i_3, \dots OP^i_k\}$ is estimated as follows:

$$Power(Q_i) = \frac{\sum_{j=1}^{k} Power(OP^i_j) * Time(OP^i_j)}{Time(Q_i)} \qquad (7)$$

where $Time(Q_i)$, $Time(OP^i_j)$ represent respectively, the execution time of the query Q_i and the execution time of OP^i_j. The power dissipated when processing is the combination of the energy consumption of the main resources identified. The formula is given by the following equation:

$$Power(OP^i_j) = W_{cpu} * C_{cpu_j} + W_{mio} * C_{mio_j} + W_{dio} * C_{dio_j} \qquad (8)$$

c) ***Step 3:*** *(Machine learning)*

To identify the different values of the power unit cost in the equations 6 and 8, we propose a polynomial regression technique. It consists in analyzing a relationship between two quantitative variables and using it to estimate the unknown value of one using the known value of the other. They can be of a simple linear model, multiple linear, nonlinear, etc. [29].

In our work, we use a non-linear regression technique involving the identified energy-sensitive variables. To do that, we set to 2 the polynomial degree for PostgreSQL and to 3 for MonetDB. We find the values of parameters of our equations: W_{cpu}, W_{mio} and W_{dio} by applying this setting in R language. The polynomial regression applied to our equations 6 and 8, we transform them as follows:

$$\begin{aligned} P(PL^i_j) &= \beta_1 * C_{cpu} + \beta_2 * C_{mio} + \beta_3 * C_{dio} + \beta_4 * C_{cpu} * C_{mio} \\ &+ \beta_5 * C_{mio} * C_{dio} + \beta_6 * C_{cpu} * C_{dio} + \beta_7 * C^2_{cpu} \\ &+ \beta_8 * C^2_{mio} + \beta_9 * C^2_{dio} + \beta_0 + \epsilon \end{aligned} \qquad (9)$$

$$\begin{aligned} P(OP^i_j) &= \beta_1 * C_{cpu} + \beta_2 * C_{mio} + \beta_3 * C_{dio} + \beta_4 * C_{cpu} * C_{mio} \\ &+ \beta_5 * C_{mio} * C_{dio} + \beta_6 * C_{cpu} * C_{dio} + \beta_8 * C^2_{cpu} \\ &+ \beta_7 * C_{cpu} * C_{mio} * C_{dio} + \beta_9 * C^2_{mio} + \beta_{10} * C^2_{dio} \\ &+ \beta_{11} * C^2_{cpu} * C_{mio} + \beta_{12} * C_{cpu} * C^2_{mio} + \beta_{18} * C^3_{mio} \\ &+ \beta_{13} * C^2_{cpu} * C_{dio} + \beta_{14} * C^2_{mio} * C_{dio} + \beta_{19} * C^3_{dio} \\ &+ \beta_{15} * C_{cpu} * C^2_{dio} + \beta_{16} * C_{mio} * C^2_{dio} + \beta_{17} * C^3_{cpu} \\ &+ \beta_0 + \epsilon \end{aligned} \qquad (10)$$

where C_{cpu}, C_{mio}, C_{dio} are parameters whose estimated values are provided by the DBMS statistics module and Processor Counter Monitor (PCM), ϵ represents measurement errors and $(\beta_1, \therefore, \beta_{19})$ are regression coefficients that will be estimated.

To estimate the values of the parameters in our equations, we use the TPC-H benchmark. We generate data at different scale factors: 5 GB and 30 GB. For each scale factor, we study and collect the characteristics of forty-four (44) Select-Project-Join (SPJ) queries and measure the energy consumed by each of

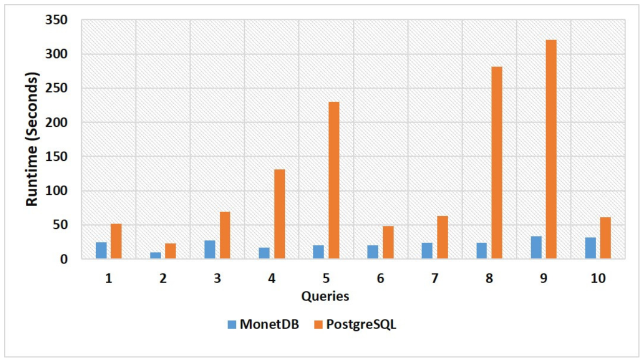


Fig. 3: Execution time comparison using SPJ queries in parallel mode on TPC-H SF5.

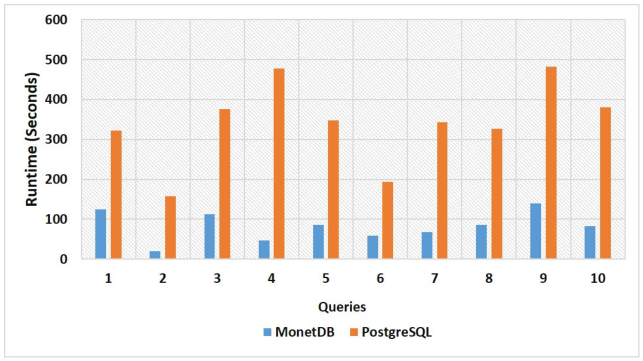


Fig. 4: Execution time comparison using SPJ queries in parallel mode on TPC-H SF30.

them using the power meter. These queries are constituted from simple to complex (Join and Group by) involving CPU intensive operations.

IV. Evaluation of Our DBMSs without Our Cost models

This section presents an experimental study comparing in terms of two NFR (rapid query performance and reduction of power consumption) our DBMS without integrating our findings. These experiments are conducted using the firt 10 SPJ queries of the TPC-H. In all our experiments, we consider SPJ queries.

A. System Setting

Our experiments are conducted in the following setting: a DELL Latitude E6430 with Dell 0CPWYR motherboard, Intel Core i5 3340M CPU@ 2.70GHz (1 CPU - 2 Core - 4 Threads), 8GB SODIMM DDR3@ Synchronous 1600 MHz main memory from Samsung, ATA Disk Toshiba MQ1ABF0 500 GB. To measure the power of the server we use power meters called Watt UP PRO at a 1Hz frequency placed between the electrical power source and the database server. It is linked via USB connections to the monitor to collect result data. The environment topology of our experimentation is shown in Figure 5. The server uses Ubuntu 18.04 bionic (kernel 5.0.0-27-generic) as the operating system with PostgreSQL release 10.10 and MonetDB release 11.33.11. In the experiments, we set the degree of parallelism (DoP) to 2. The following commands are used to set this degree on both DBMSs: $max\_parallel\_workers\_per\_gather = \#number\#$ (PostgreSQL) and $gdk\_nr\_threads = \#number\#$ (MonetDB). We generate data at different scale factors (SF): $5GB, 10GB, 30GB$ to populate our databases.

B. Query Performance Analysis

In this experiment, we evaluate the first NFR of both DBMS using two database configurations $5GB$ and $30GB$. Lastly, elapsed time for each SPJ query on the same database has been compared. Figures 3 and 4 summarize the execution time results obtained using the firt 10 SPJ queries.

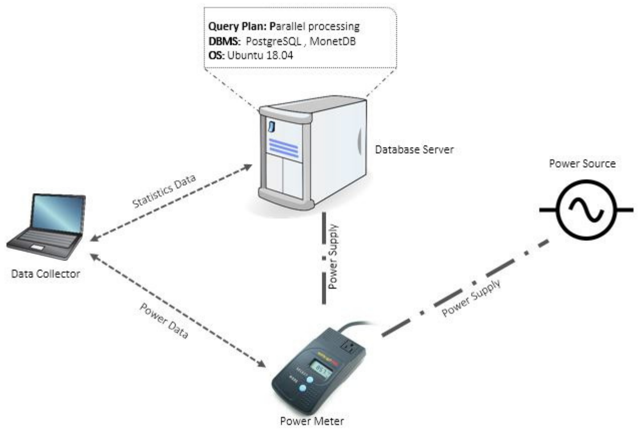


Fig. 5: Deployment of our experimental testbed.

These results show that the performance of MonetDB outperforms those obtained by PostgreSQL for all considered queries and datasets. This performance gaps is mainly due to the fact that PostgreSQL cumulates an important number of page faults when processing queries. The row-store systems have to scan and use the entire n-tuples rather than only the needed columns values. Therefore, the entire rows plus the built-in index tree cannot reside long enough in the main memory or in the cache memories. They must be swapped on the disk this leads to many disks IOs. For column-oriented systems like MonetDB, just the values of the columns required to answer the queries are loaded.

C. Power Consumption Analysis

In Section IV-B, we find that the performance of MonetDB is better than that of PostgreSQL. This is particularly interesting in the context of energy consumption as energy is a quantity that depends on time flowing. To compare the energy consumption on both systems when executing queries, we measure the power dissipated per second using an amperage apparatus named wattmeter. At the end of these measurements, we calculate the average energy consumed by each query. Figures 8 and 9 show average energy consumption for SF5 and SF30 datasets respectively. Starting from the results presented in the Figures

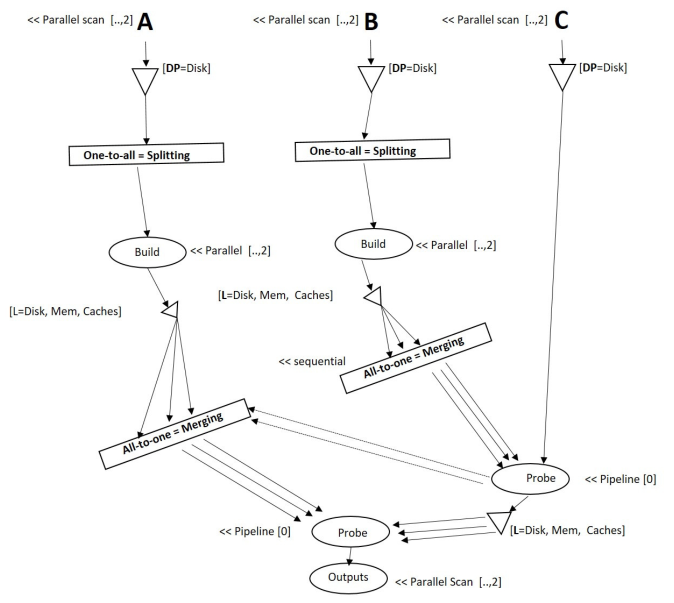


Fig. 6: The DLP-Graph of Q query.

		DBMS	
	Variables	**MonetDB**	**PostgreSQL**
Outputs	Multiple R-squared	0.58	0.83
	Adjusted R-squared	0.45	0.78
	Residual	$[-2, 2]$	$[-3, 3]$

TABLE I: Regression statistics and residual outputs.

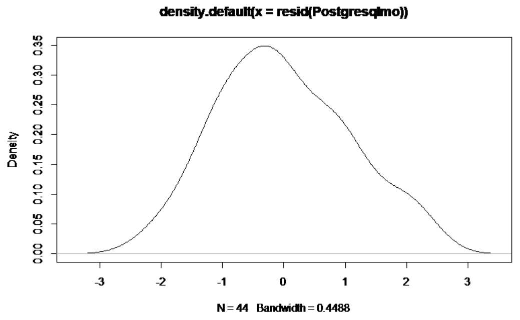
8 and 9, the average energy consumed by *MonetDB* during the execution of the queries is reasonable than for *PostgreSQL*. The high consumption of the PostgreSQL query processor can be explained by the characteristics of this type of row-store DBMSs. MonetDB uses compression techniques to reduce the cost of data scans. This directly impacts query performance due to fewer I/O requests and page faults. *To summarize, all experiments that we conducted to execute the* 10 *TPC-H queries in both DBMS unequivocally place first MonetBD, far ahead of PostgreSQL for two NFR.*

V. Evaluation of Our Green DBMSs

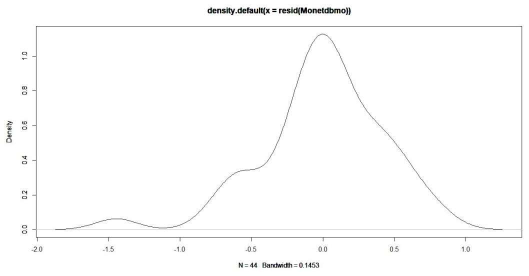
In this section, we evaluate our green DBMSs obtained by using our mathematical cost models in the same environment as for the previous experiments(cf. subsection IV-A).

A. Parameters Values

We use polynomial regression using R version 3.5.2 language to determine parameters values for our models described in the equations 9 and 10. The correlation coefficient denoted by R measures the strength and direction of a linear relationship between two variables. After the analysis of the collected data, Table in I give more information about regression statistics and residual output obtained from language R. The statistical variable R define the coefficient of correlation or determination of the model.



(a) For data collected from PostgreSQL.



(b) For data collected from MonetDB.

Fig. 7: Residual density of training data.

The following equations 12 and 11 describe our equations models in 10 and 9 respectively with these parameters values.

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

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

Figures 7a and 7b respectively, illustrate the residual density for the PostgreSQL and MonetDB systems obtained from Language R after training data analysis.

B. Query Plan Evaluation

Typically, a database server receives each query from a user, compiles and executes it. The optimizer selects one plan from a set of plans that have acceptable performance. The actual evaluation model used in traditional database systems uses cost-driven approaches that select the final query plan

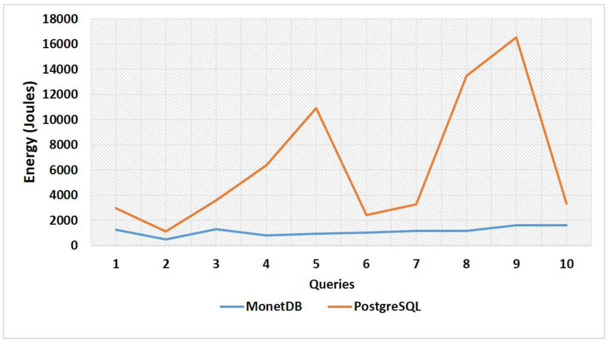


Fig. 8: Average energy consumption for MonetDB comparing to PostgreSQL on TPC-H SF5.

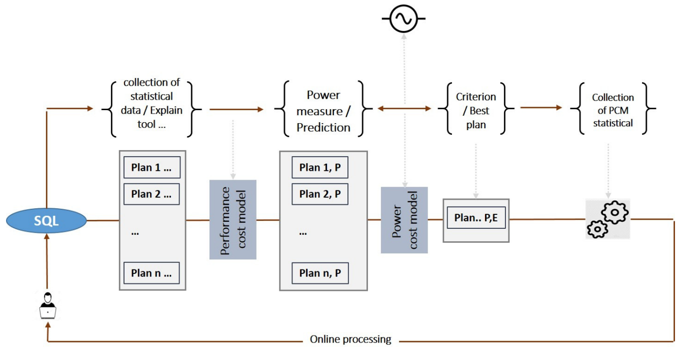


Fig. 10: For PostgreSQL DBMS.

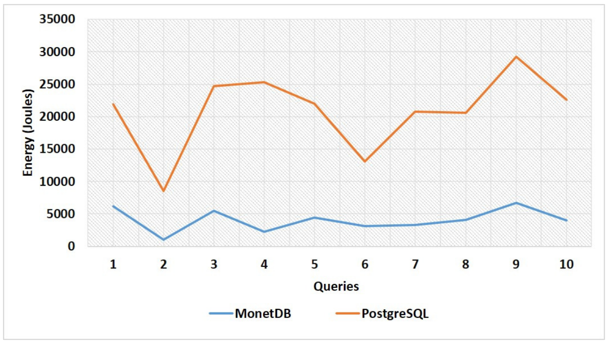


Fig. 9: Average energy consumption for MonetDB comparing to PostgreSQL on TPC-H SF30.

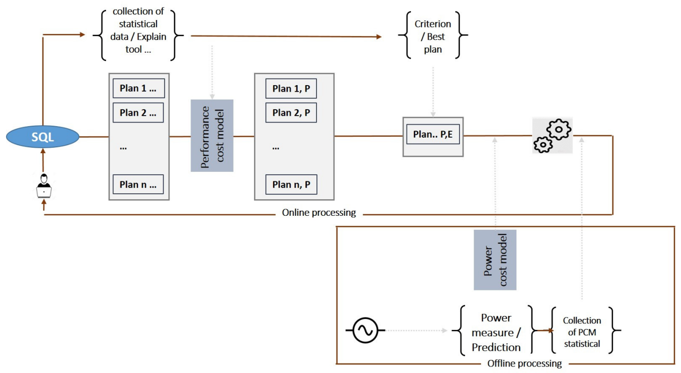


Fig. 11: For MonetDB DBMS.

with minimum cost. For energy-aware optimizer, a cost model based on energy consumption must take place in the core of system. Rather than choosing a plan with optimal performance, the energy cost model can be used to choose a plan that saves energy or to define a certain threshold of the trade-off between energy and performance, for more details see our works in [31]. The adjustable trade-off between performance and energy is made by using a criterion that reflects the choice of users/administrators. The criterion model adopted has the following format: $C = n * T + (1 - n) * E$ with T denote processing time, E denote the power cost, C denote the plan cost and n is constant value in interval $[0, 1]$. In the case of PostgreSQL, it is easy to integrate our energy cost in the optimizer because it is designed following a cost-based approach, the process to integrate it is described in [31]. Figure 10 illustrates our query model evaluation on PostgreSQL.

For MonetDB, it is not easy to integrate energy cost because the query execution plan is not optimal in terms of cost model [12]. For this reason in this study, we run our model cost outside from the threes layers (MonetDB kernel). Figure 11 illustrates our query model evaluation on MonetDB.

C. Results

We used our energy modeling defined in equations 12 and 11 to execute the 22 queries. We compare the energy estimated (EE) by our cost models with the real energy (ER) consumed by the database server during the processing of the query in parallel mode. To validate the energy model as an accurate prediction of cost for database system, we use the metric called Estimation Error (ERR) to quantify the model accuracy. The metric is defined by the following formula:$EER = \frac{|ER - EE|}{ER}$. ER denotes the real values of power measured by the power meter and EE denotes the prediction values of our models (a non-linear regression). Tables II and III give the rate estimation error obtained by using our energy cost model for parallelism degree fixed to 2 on PostgreSQL and MonetDB respectively. Noted that in Table III, the average of the estimation errors is 2.5% and the maximum error is 13%. The difference between the estimated average power and the measured one is very small in almost all cases. Exploiting this model obtained with a non-significant estimation rate by varying the value of n in the criterion model will allow us to highlight the energy-saving gains that we will present in our future work. On Table II the average of the error estimation is 4.1% and the maximum error is 10%. From the analysis of the two tables, we can conclude that our energy cost model fits as well with $MonetDB$ as with $PostgreSQL$ because of non-significant estimation estimate deviations.

VI. Related Work

In this section, we review then main studies and research efforts dedicated to reduce the energy consumption when processing data. Figure 12 lists the major approaches explored to increase the EE. This review covers major elements of the

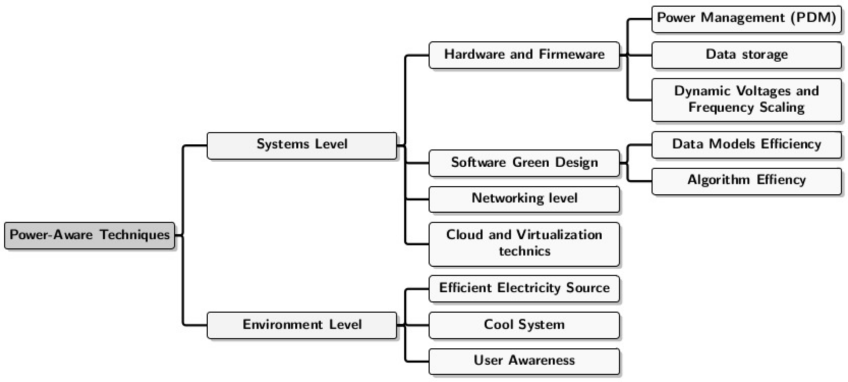


Fig. 12: Overview of techniques to improve the EE.

TABLE II: Prediction Error Rate on the Postgresql DBMS SF5.

Queries	*Measured*	*Prediction*	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%

When collecting system analysis data, the *Explain Analysis* command on queries 18 and 21 took a long time without giving any answers. We did not integrate them in the case of $PostgreSQL$.

TABLE III: Prediction Error Rate on the MonetDB DBMS SF5.

Queries	*Measured*	*Prediction*	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%

DBMS environment.

Physical components (Hardware) and firmware level: Techniques applied at this level commonly called Dynamic Power Management can be broadly divided into two categories: dynamic component deactivation and dynamic performance scaling [3]. Dynamically controlling the clock speed of the CPU has largely inspired the development of dynamic voltage adjustment (DVS) techniques. Study in [20] proposed a PVC (Processor Voltage/ Frequency Control) mechanism to trade energy consumption for performance. They perform certain instructions at a lower processor voltage and frequency by taking advantage of modern processor capabilities. Their experiments showed that PVC can be used to reduce the CPU energy consumption by 20% and 49%, while incurring 6% and 3% response time penalties on MySQL. [6] proposed the Predictive Energy Saving Online Scheduling (PESOS) algorithm in the context of Web search. Using the DVFS technique, PESOS selects the most appropriate CPU frequency to process a query. The goal of PESOS is to reduce the CPU energy consumption of a query-processing node while imposing a required tail latency on the query response times. [17] presents a technique called *POLARIS* for reducing the power consumption of transactional database systems. POLARIS directly manages processor DVFS within the DBMS and controls database transaction scheduling. Its goal is to minimize power consumption while ensuring the transactions are completed within a specified latency target. In [33], the authors explored the performance behavior and the related energy consumption of Solid State Drives(SSDs) under typical access patterns for I/O database intensive applications.

Application-level: Many applications can be executed in different ways to accomplish the same computational task. In [24], an approach to improve the EE of software by optimizing design patterns automatically at compile time is proposed. They transform design for the Observer and Decorator patterns and found energy reductions in the range of 4.32% to 25.47%. In [22], a framework to assist software engineers to develop high-end energy-efficient applications (*SEEDS*) has been proposed.

Telecommunication networking level: Telecommunication networks constitute a major sector of ICT and they undergo a tremendous growth. Energy is mostly consumed in network transmission and switching equipment such as routers. In the literature, we observe that the focus of many research is on control and optimization strategies for computer network equipment enabling energy saving, by adapting network capacities and computing resources to the current traffic load and demands, while ensuring end-to-end quality of service (QoS)[4]. In [16], authors developed a big data analytics subystem called BigOptiBase to provide elastic energy efficient solutions for the base stations using data analytics and machine learning technologies. This tool, which aims to optimize the energy consumption of the base stations, takes decisions based on the energy policy of the base stations taking into account the user's configuration. Based on historical analysis and real-time tuning, the authors in [23] introduce a novel application-layer solution called GreenDataFlow, which aims to achieve high data transfer throughput while keeping the energy consumption at the minimal levels. It is based on mathematical modeling with offline knowledge discovery and adaptive online decision-making. Experimental results show that GreenDataFlow outperforms the closest competing state-of-the art solution. To achieve a power saving in router, authors in [39] proposed to follow two approaches; power efficient designing and power saving designing. The former is an approach to create a high performance router at low power consumption and the latter is an approach to save wasted power. For power efficient designing, they have developed technologies for integrating the ASICs/FPGAs and memories of routers. for power saving design, the authors worked on static performance control, which allows turning off unused ports and modules. Furthermore, they proposed a technology that lowers the frequency of lightly utilized modules to save the wasted power. Running under the low frequency mode, the power efficiency improved by 10-20%.

Environmental management and conventional rules: In response to the over energy consumption in datastores and the need to reduce the related environmental, economic and energy supply security impacts, the governments, and corporate sector impose regulations and acts. The European Commission (EC) created a Code of Conduct for Energy Efficiency[2] in Data Centers in 2008 with the aim of improving the energy efficiency. Like EC, the U.S. Department of Energy has the ENERGY STAR[5] program, which offers energy efficiency guidelines for all types of buildings including Data Centers. In addition to the Code of Conduct for Energy Efficiency, there are several energy-efficiency standards imposed to control the manufacture and use of equipment. Cooling system contribute about one-third of this energy use in data [15]. The environmental temperature recommended by American Society of Heating, Refrigeration and Air Conditioning (ASHRAE) for Class A1 to A2 Data Centers is between 18° Celsius and 27° Celsuis (64° to 81°F).

Software-Based Techniques: Most important studies in this category concern (i) the definition of cost models to predict the energy and (ii) the proposition of cost-driven techniques for reducing energy. In [38], the authors build a static power profile for each basic database operation using a simple linear regression technique. In [18], the authors use pipeline-based modeling to the sources of peak power consumption for a query and to recommend plans with low peak power. The proposed model relies on the pipeline segmenting of the query. Local learning support vector regression (SVR) is used by [13] to perform energy consumption predictions and they compare their results with traditional SVRs and with deep neural networks exploiting an H2O machine learning platform for big data. They found that local SVR outperformed others in terms of prediction accuracy and computation time. In Cloud database systems environment, where users are billed for the use of services, the authors in [37] proposed a method that adaptively optimizes a query execution plan to satisfy both the query response time and monetary cost objectives. Experimental results show that the proposed method can save either the time cost or the monetary cost based on the type of queries. The authors in [14] analyze the effect of the three main cache structures (Database Buffer Cache, Dictionary Cache, and Library Cache). Based on this, they have taken the cost of memory into account in their linear model cost dedicated to sequential query processing mode. In [31][32], the authors proposed cost models to predict the power consumption of single running queries. The proposed model relies on pipeline segmenting of the query. The model is built based on the CPU and I/O cost of each pipeline. It measured energy in an offline task, using a non-linear regression technique (polynomial regression). Also the authors proposed an initiative that integrates the energy into the physical design of database systems when selecting materialized views. They propose a multi-objective formalization of the materialized view selection problem, considering two objectives: query performance and energy consumption while executing a given workload. In [8][9], we confront the sequential and parallel execution modes and study their impact on EE. They extented the sequential cost model proposed in [30], by taking into account the parallel mode in a centralized DBMS to predict the energy cost when executing a given query. *Contrary to our proposal, most of these studies used linear regression techniques.*

VII. Conclusions

In this paper, we attempt to alert academia, industry, IT companies and funding agencies by promoting research, tools and software solutions to save energy in Data Management Systems. We identify query processing as a critical task, which consumes significant energy. Therefore, we propose to re-engineer green query processors. To reach this objective, a comprehensive procedure, consisting of four main steps, was proposed: (1) a deep audit that allows understanding the query processor functioning at a low level, (2) identification and measurement of main energy-sensitive parameters in hardware and software components, (3) proposing mathematical cost models estimating consumed energy when executing queries on a DBMS and (4) finding values of these energy-sensitive parameters based on nonlinear regression, going a step beyond previous linear regression models. This procedure highlighted the importance of auditing DBMSs by understanding their internal subsystems, before proposing a suitable energy modeling. To validate our novel procedure, we analyzed two open-source DBMSs with complementary storage mechanisms: PostgreSQL and MonetDB. After conducting a deep audit of these DBMSs, mathematical cost models with relevant parameters estimating energy consumption were proposed. Extensive experiments were conducted to evaluate the effectiveness of our proposed models. We hope our research will spark interest in new directions to save energy to store, query and analyze databases.

We plan in our future work to use others machine learning techniques and compare them with the nonlinear regression results on much larger datasets to determine their impact on our energy models accuracy. And also, we plan to apply the same approach on HYRISE system and compare it with moneDB and PostgreSQL system.

References

[1] D. Abadi, R. Agrawal, A. Ailamaki, M. Balazinska, P. A. Bernstein, M. J. Carey, S. Chaudhuri, J. Dean, A. Doan, M. J. Franklin *et al.*, "The beckman report on database research," *Communications of the ACM*, vol. 59, no. 2, pp. 92–99, 2016.

[2] M. Avgerinou, P. Bertoldi, and L. Castellazzi, "Trends in data centre energy consumption under the european code of conduct for data centre energy efficiency," *Energies*, vol. 10, no. 10, p. 1470, 2017.

[3] A. Beloglazov and et al., "A taxonomy and survey of energy-efficient data centers and cloud computing systems," *Advances in Computers*, vol. 82, no. 2, pp. 47–111, 2011.

[4] K. Bilal, S. U. R. Malik, O. Khalid, A. Hameed, E. Alvarez, V. Wijaysekara, R. Irfan, S. Shrestha, D. Dwivedy, M. Ali *et al.*, "A taxonomy and survey on green data center networks," *Future Generation Computer Systems*, vol. 36, pp. 189–208, 2014.

[5] R. Brown, C. Webber, and J. G. Koomey, "Status and future directions of the energy star program," *Energy*, vol. 27, no. 5, pp. 505–520, 2002.

[6] M. Catena and N. Tonellotto, "Energy-efficient query processing in web search engines," *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, no. 7, pp. 1412–1425, July 2017.

[7] M. Dayarathna, Y. Wen, and R. Fan, "Data center energy consumption modeling: A survey," *IEEE Communications Surveys Tutorials*, vol. 18, no. 1, pp. 732–794, 2016.

[8] S. P. Dembele, L. Bellatreche, C. Ordonez, and A. Roukh, "Think big, start small: a good initiative to design green query optimizers," *Cluster Computing*, vol. 23, no. 3, pp. 2323–2345, 2020.

[9] S. P. Dembele, A. Roukh, and L. Bellatreche, "Vers des optimiseurs verts de requêtes en mode parallèle," in *EDA*, 2018.

[10] T. Economist. The world's most valuable resource is no longer oil, but data. [Online]. Available: https://www.economist.com/leaders/2017/05/06/the-worlds-most-valuable-resource-is-no-longer-oil-but-data

[11] R. Elmasri and S. Navathe, *Fundamentals of Database Systems*, 6th ed. USA: Addison-Wesley Publishing Company, 2010.

[12] R. Goncalves and M. Kersten, "The data cyclotron query processing scheme," *ACM Trans. Database Syst.*, vol. 36, no. 4, pp. 1–35, 2011.

[13] K. Grolinger, M. A. Capretz, and L. Seewald, "Energy consumption prediction with big data: Balancing prediction accuracy and computational resources," in *2016 IEEE International Congress on Big Data (BigData Congress)*. IEEE, 2016, pp. 157–164.

[14] B. Guo, J. Yu, B. Liao, D. Yang, and L. Lu, "A green framework for dbms based on energy-aware query optimization and energy-efficient query processing," *Journal of Network and Computer Applications*, vol. 84, pp. 118–130, 2017.

[15] M. Iyengar, R. Schmidt, and J. Caricari, "Reducing energy usage in data centers through control of room air conditioning units," in *12th IEEE Intersociety Conference on Thermal and Thermomechanical Phenomena in Electronic Systems*, 2010, pp. 1–11.

[16] E. Kassela, N. Provatas, A. Tsiourvas, I. Konstantinou, and N. Koziris, "Bigoptibase: Big data analytics for base station energy consumption optimization," in *2019 IEEE International Conference on Big Data (Big Data)*. IEEE, 2019, pp. 6098–6100.

[17] M. Korkmaz, M. Karsten, K. Salem, and S. Salihoglu, "Workload-aware cpu performance scaling for transactional database systems," in *Proceedings of the 2018 International Conference on Management of Data*. ACM, 2018, pp. 291–306.

[18] M. Kunjir, P. K. Birwa, and J. R. Haritsa, "Peak power plays in database engines," in *EDBT*. ACM, 2012, pp. 444–455.

[19] W. Lang, R. Kandhan, and J. M. Patel, "Rethinking query processing for energy efficiency: Slowing down to win the race." *IEEE Data Eng. Bull.*, vol. 34, no. 1, pp. 12–23, 2011.

[20] W. Lang and J. Patel, "Towards eco-friendly database management systems," *arXiv preprint arXiv:0909.1767*, 2009.

[21] S. Manegold, P. A. Boncz, and M. L. Kersten, "Generic database cost models for hierarchical memory systems," in *VLDB*, 2002, pp. 191–202.

[22] I. Manotas, L. Pollock, and J. Clause, "Seeds: a software engineer's energy-optimization decision support framework," in *ICSE*, 2014, pp. 503–514.

[23] M. S. Z. Nine, L. Di Tacchio, A. Imran, T. Kosar, M. F. Bulut, and J. Hwang, "Greendataflow: Minimizing the energy footprint of global data movement," in *2018 IEEE International Conference on Big Data (Big Data)*. IEEE, 2018, pp. 335–342.

[24] A. Noureddine and A. Rajan, "Optimising energy consumption of design patterns," in *Proceedings of the 37th International Conference on Software Engineering - Volume 2*, ser. ICSE '15. IEEE Press, 2015, pp. 623–626.

[25] OPCM, "Processor counter minitor," https://github.com/opcm/pcm, 2013.

[26] A. Ouared, Y. Ouhammou, and L. Bellatreche, "Qosmos: Qos metrics management tool suite," *Computer Languages, Systems & Structures*, vol. 54, pp. 236–251, 2018.

[27] M. Poess and R. O. Nambiar, "Energy cost, the key challenge of today's data centers: a power consumption analysis of tpc-c results," *PVLDB*, vol. 1, no. 2, pp. 1229–1240, 2008.

[28] M. Raasveldt and H. Mühleisen, "Monetdblite: An embedded analytical database," *CoRR*, vol. abs/1805.08520, 2018.

[29] J. Rawlings, *Applied regression analysis: a research tool*. Wadsworth & Brooks/Cole Advanced Books & Software, 1988.

[30] A. Roukh and L. Bellatreche, "Eco-processing of olap complex queries," in *International Conference on Big Data Analytics and Knowledge Discovery*. Springer, 2015, pp. 229–242.

[31] A. Roukh, L. Bellatreche, S. Bouarar, and A. Boukorca, "Eco-physic: Eco-physical design initiative for very large databases," *Inf. Syst.*, vol. 68, pp. 44–63, 2017.

[32] A. Roukh, L. Bellatreche, and C. Ordonez, "Enerquery: energy-aware query processing," in *ACM CIKM*, 2016, pp. 2465–2468.

[33] D. Schall, V. Hudlet, and T. Härder, "Enhancing energy efficiency of database applications using ssds," in *C3S2E Conference*. ACM, 2010, pp. 1–9.

[34] A. Shehabi, S. J. Smith, D. A. Sartor, R. E. Brown, M. Herrlin, J. G. Koomey, E. R. Masanet, N. Horner, I. L. Azevedo, and W. Lintner, "United states data center energy usage report," Energy Technology Area, Report, June 2016.

[35] M. Tisné. Data isn't the new oil, it's the new co2. [Online]. Available: https://luminategroup.com/posts/blog/data-isnt-the-new-oil-its-the-new-co2

[36] D. Tsirogiannis, S. Harizopoulos, and M. A. Shah, "Analyzing the energy efficiency of a database server," in *sigmod*, 2010, pp. 231–242.

[37] C. Wang, Z. Arani, L. Gruenwald, and L. d'Orazio, "Adaptive time, monetary cost aware query optimization on cloud database systems," in *2018 IEEE International Conference on Big Data (Big Data)*, 2018, pp. 3374–3382.

[38] Z. Xu, Y.-C. Tu, and X. Wang, "Dynamic energy estimation of query plans in database systems," in *ICDCS*, 2013, pp. 83–92.

[39] M. Yamada, T. Yazaki, N. Matsuyama, and T. Hayashi, "Power efficient approach and performance control for routers," in *2009 IEEE International Conference on Communications Workshops*, 2009, pp. 1–5.

[40] C. Yang, Y. Du, Z. Du, and X. Meng, "Micro analysis to enable energy-efficient database systems," in *EDBT*, 2020, pp. 61–72.