The Impact of Multicore CPUs on Eco-Friendly Query Processors in Big Data Warehouses

Ayoub Bouhatous
LIAS/ISAE-ENSMA
Poitiers, France
LISI/Cadi Ayyad University
Marrakech, Morocco
ayoub.bouhatous@ensma.fr

Ladjel Bellatreche
LIAS/ISAE-ENSMA
Poitiers, France
bellatreche@ensma.fr

El Hassan Abdelwahed
LISI/Cadi Ayyad University
Marrakech, Morocco
abdelwahed@uca.ac.ma

Carlos Ordonez
Department of Computer Science
University of Houston
USA

Abstract*—Given the large and growing volume of big data and frequent use of complex analytical queries, understanding energy efficiency of query processing has become a critical research issue, as highlighted by database systems papers in the last few years. Common software solutions mainly consider IO cost models to estimate energy consumption when executing queries. On the other hand, current hardware solutions benefit from advances in the development of green components and their associated tuning techniques, especially dynamic voltage and frequency scaling (DVFS), which can balance the performance and power consumption of multicore CPUs. Unfortunately, to the best of our knowledge, there is an absence of solutions mixing both (hardware and software). Heeding this gap, we propose a novel predictive model to measure and predict energy consumption of analytical queries when using multi-core processors and different frequency configurations. We first experimentally illustrate the surprising impact of CPU frequency and the number of processor cores on execution time, power and energy consumption. Second, we introduce an extended predictive model that enriches a well-known machine learning cost model with our new angle, the frequency scaling in multi-core environment. Specifically, by using Support Vector Regression and Random Forest Regression, we compute the energy coefficients of an accurate regression model for energy prediction. Experiments with benchmark data sets TPC-H and TPC-DS evaluate our proposed framework in terms of energy consumption reduction, showing promising results.

***Index Terms*—Query Optimization, Energy Efficiency, Big Data Warehouses, Green Computing, Multi-core CPU, Frequency Scaling.**

I. Introduction

One of the ways to combat climate change and reduce emissions of carbon dioxide CO_2 is to study the energy efficiency (EE) of devices used in our daily life. Recently, EE has become an important aspect that attracts the attention of researchers in different disciplines [1]–[5]. The data industry is one of the sectors with high energy consumption [6]. Therefore, several recommendations and directives have been launched by organizations and governments to regulate servers and data storage products to improve their energy efficiency [7].

Power and cooling costs are some of the highest costs in data storage and processing products today, which makes improvement in EE crucial [8], [9]. For ten years, the academia and industry communities propose eco-design solutions and services for storing and processing massive amounts of data to meet the challenges of the ecological and energy transition. These solutions are dedicated to traditional Database Management Systems (DBMSs) and data centers [5], [10]–[12]. The major solutions and services of energy efficiency in database systems are compiled in this recent survey [6].

Note that the most natural place to add EE awareness to a data storage and processing product is the query optimizer [8]. By examining the literature, the existing studies on EE query processing are classified into two main classes including [6], [11] (i) *hardware approaches* and (ii) *software approaches*. At the hardware level, many efforts have been undertaken. They are declined into two main directions: (a) the development of new energy-efficient hardware and (b) the proposal of techniques to calibrate the usage of energy of significant hardware such as central processing unit (CPU). The latter consumes more than 50 percent of the power required to run a system such as a DBMS [13], [14]. Dynamic Voltage and Frequency Scaling (DVFS) [15] is one of the most popular techniques since it aims at adjusting processor frequencies [8]. DVFS plays a crucial role in adjusting the performance and power consumption of different kinds of systems (mobile systems, desktops,and servers). It is supported by many types of processors (e.g., AMD PowerNow and Intel's SpeedStep). DVFS allows a voltage and frequency, and hence energy consumption of processors, to be adjusted at run time using operating system modules called frequency governors [16], [17].

In software approaches, several initiatives and solutions have been proposed and implemented in DBMSs. Analytical cost models represent the common thread of these approaches. Their role is to predict the energy consumption of queries based on the main resources consumed during query processing. The three main resources are considered by these existing cost models: CPU, IO (inputs-outputs), and RAM. By analyzing these cost models, we identify that there is a consensus of the existing analytical cost models in their definition to capture energy consumption when executing a given analytical query in Q. This consensus is defined as follows:

$$Power(Q) : \alpha \cdot IO + \beta \cdot CPU + \gamma \cdot RAM \qquad (1)$$

where α, β, γ are power factors that are determined using machine learning (ML) algorithms, whereas IO, CPU, and RAM metrics are provided by the query optimizer of the DBMS hosting the target database.

While hardware solutions dominate the literature of EE of data processing systems, they are expensive and hence are only cost-effective when widely deployed. Software solutions may be an interesting alternative to reduce energy consumption by combining them with existing hardware techniques such as dynamic frequency scaling (DFS). By deeply analyzing these existing research efforts, we realize the absence of solutions that mix hardware and software aspects, except the work proposed in [8] that proposed dynamic fine-grained scheduling of analytical queries in-memory databases but without providing adequate analytical cost models.

In this work, we propose a solution hybridizing the two most popular techniques from hardware (DFS) and software (cost models) in order to improve the prediction accuracy of the existing energy cost models and augment their genericity to cover different processor configurations. To the best of our knowledge, this work is the first that performs this hybridization.

Our main contributions are:

- We provide an extensive study to evaluate the effect of varying the processor frequency and the number of its cores on energy consumption when executing a workload of queries;
- We propose an accurate and portable cost model to predict the energy consumption of queries when using multi-core processors and different frequency configurations. The energy coefficients of our model are determined using two machine learning techniques (Support Vector Regression and Random Forest Regression);
- We conduct intensive experiments to validate our proposal using the TPC-H and TPC-DS benchmarks and PostgreSQL DBMS and a comparison of our findings with two major existing software solutions.

Our paper is organized as follows: Section II describes our motivating example. In section III, we present the fundamental concepts to elaborate our solution. Section IV covers our related works. Section V describes our pipeline-based energy model in detail. Section VI presents our experimental evaluation and a comparison with the state-of-the-art solutions. Finally, section VII concludes our work and gives some perspectives.

II. Motivating Example

In this section, we illustrate through an example, the impact of processor configurations on three main costs: (i) query processing, (ii) power, and (iii) energy consumption of the entire server during the query execution.

To do so, we consider six analytical queries $\{Q_1, Q_3, Q_4, Q_5, Q_6, Q_7\}$ issued from the TPC-H[1] benchmark running on a data warehouse with a scale factor of 50 GB[2]. PostgreSQL DBMS is used to host this warehouse. The above queries contain respectively: 1, 2 joins operations that are CPU and IO intensive. These queries are then executed by varying the number of processors and their frequencies. The frequency value of the processor of our server ranges from 800 MHz to 3.4 GHz. From this interval, we select three different values to generate three frequency classes: (i) *Small Frequency* (SF) of 800 MHz, (ii) *Medium Frequency* (MF) of 2.3 GHz, and *Large Frequency* (LF) of 3.4 GHz. For each value, we vary the number of processor cores (1, 2, and 4).

As a consequence, nine different configurations are considered: [SF(1 CPU), SF(2 CPUs), SF(4 CPUs)], [MF(1 CPU), MF(2 CPUs), MF(4 CPUs)], and so on. We consider a hypothesis that the same frequency is assigned to all processor cores. The results of our experiments are summarized as follows:

- **Impact of processor configuration on query execution cost:** In this experiment, we compute the execution cost of our queries under nine different configurations. The obtained results are described in Fig. 1 that show a naive finding stipulating that when the number of processors and their frequency values are high then query execution saving is also high. Consequently, the configuration LF(4 CPUs) is the best one for this requirement.
- **Impact of processor configuration on power consumption cost:** The power consumption cost is measured for each query executed in different configurations. The results in Fig. 2 are the opposite of the above ones (Fig. 1), in the sense that when the frequency and/or the number of processors are low, the reduction of power consumption is guaranteed (the case of the configuration SF(1 CPU)). We note that the best scenario for the query execution time criteria contradicts the one dedicated to the power consumption requirement.
- **Impact of processor configuration on energy execution cost:** Similarly, Fig. 3 gives the energy consumption cost of our queries under nine configurations. We observe that there is any configuration that is better for almost cases. Note that the energy consumption cost of a given query is equal to its execution time cost multiplied by its power cost. Therefore, to ensure energy consumption reduction, the best trade-off between execution time cost and power consumption cost has to be found.

To summarize, there is no better scenario for most configurations. This is due to the requirements of queries in terms of CPU, IO, and memory. Another important point that we got from these experiments concerns the crucial role of the number of processors and their frequencies on query energy consumption costs. Therefore, their integration into analytical energy cost models is necessary.

[1]http://www.tpc.org/tpch/
[2]Giga Byte

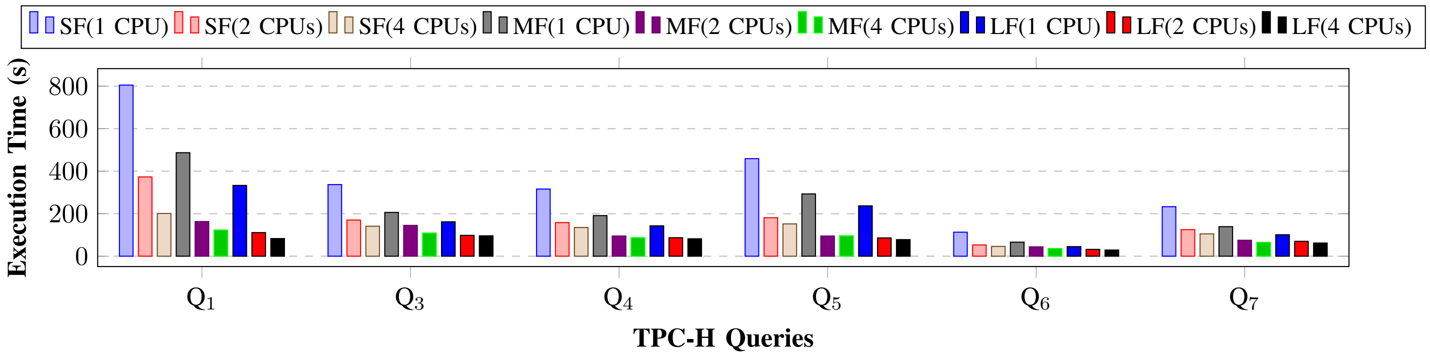


Fig. 1. Execution time of six TPC-H queries under nine different processor configurations.

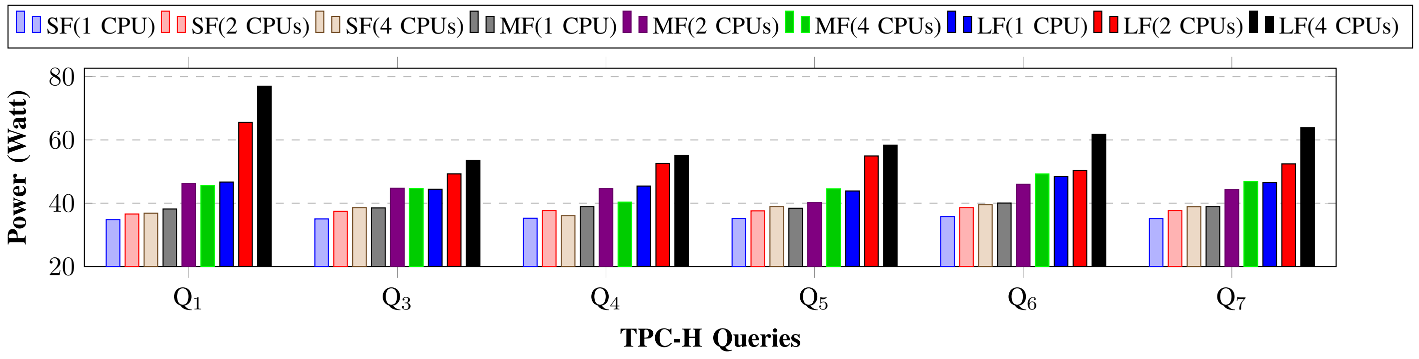


Fig. 2. Power Consumption of six TPC-H queries under nine different processor configurations.

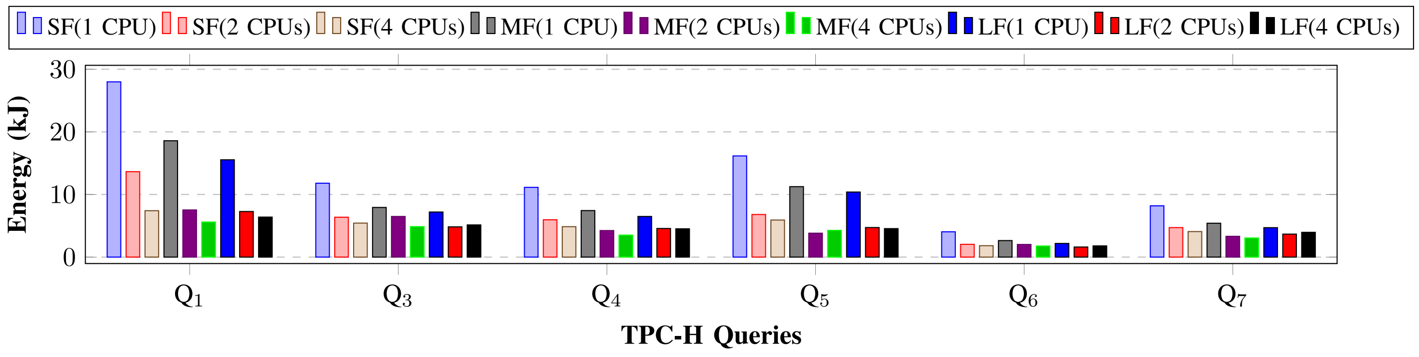


Fig. 3. Energy consumption of six TPC-H queries under nine different processor configurations.

III. Background

This section introduces the key concepts related to energy and query processing in DBMSs.

Definition 1: **Power** represents the rate of doing work or the energy amount per unit of time. It is measured in watts (W). Formally, the power can be defined as follows:

$$P = \frac{J}{T} \tag{2}$$

where P, T, and J represent respectively, a power, a period of time, and the total work performed in that period of time.

Definition 2: **Energy** is defined as a measure of the ability to do work. The joule is its measurement unit. Formally, energy is defined as follows:

$$E = P \cdot T \tag{3}$$

where P, T, and E represent respectively, a power, a period of time, and the energy.

Usually, in information technology context, the energy is the electrical energy consumed by the computing system over time, whereas the power is the rate of electrical energy consumed per a second [18]. The consumption of the electrical power of a given system can be decomposed into two parts: (1) **Baseline Power** (P_{static}) that represents the power required to operate the fans, processor, memory, IO devices, and other components of the motherboard in an *idle state*. (2) **Active**

Power (P$_{dynamic}$) is the power consumed when executing a workload. The total electrical power (P_{total}) consumed by a given computing system is the summation of the static power and the dynamic power.

$$P_{total} = P_{static} + P_{dynamic} \tag{4}$$

In this paper, we consider the average power consumed during the query execution.

Definition 3: **Energy efficiency (EE)** is the use of the same power to provide better performance or the same service with less energy [14]. It is defined as follows:

$$\begin{aligned} EE = \frac{WorkDone}{Energy} &= \frac{WorkDone}{Power \cdot Time} \\ &= \frac{Performance}{Power} \end{aligned} \tag{5}$$

A. Query Processing in a DBMS

Processing a given SQL like query passes through four main steps: (i) query code parsing, (ii) transformation, (iii) planning and optimization, and (iv) execution. **Query code parsing:** generates a parse tree from a plain text SQL statement after checking its syntax and performing a semantic analysis. **Transformation:** transforms the parse tree to a logical plan (algebraic plan) using algebraic rules. **Planning and optimization:** generates several query execution plans that can be executed from the logical query plan. Next, the planner uses a cost model to estimate the cost of each query plan and selects the most efficient one. **Execution:** executes the query by accessing the tables and access methods in the order created by the selected execution plan.

B. Pipeline Segmentation

The execution of a generated query plan is performed using a pipeline strategy that segments the plan into a set of pipelines based on blocking and non-blocking relational operators [19], [20]. An operator is blocking if it cannot produce any output tuple without reading all its inputs (e.g., sort operator) [19]. Thus, a pipeline consists of a set of concurrently running operators. An example of segmentation of a query execution plan into pipelines is illustrated in Fig. 4, where we can see that the query plan is segmented into 4 pipelines $\{PL1, PL2, PL3, PL4\}$. These pipelines are executed in sequential order: $PL1 \longrightarrow PL2 \longrightarrow PL3 \longrightarrow PL4$.

The average power consumption of a given query Q whose execution plan is composed of n pipelines $\{PL_1, PL_2, \ldots, PL_n\}$, is defined as follows [12]:

$$Power(Q) = \frac{\sum_{i=1}^{n} Power(PL_i) \cdot Time(PL_i)}{Time(Q)} \tag{6}$$

where *Power(PL$_i$)*, *Time(PL$_i$)*, and *Time(Q)* represent respectively, the average power of the pipeline PL$_i$, the execution time of PL$_i$, the time required to execute Q.

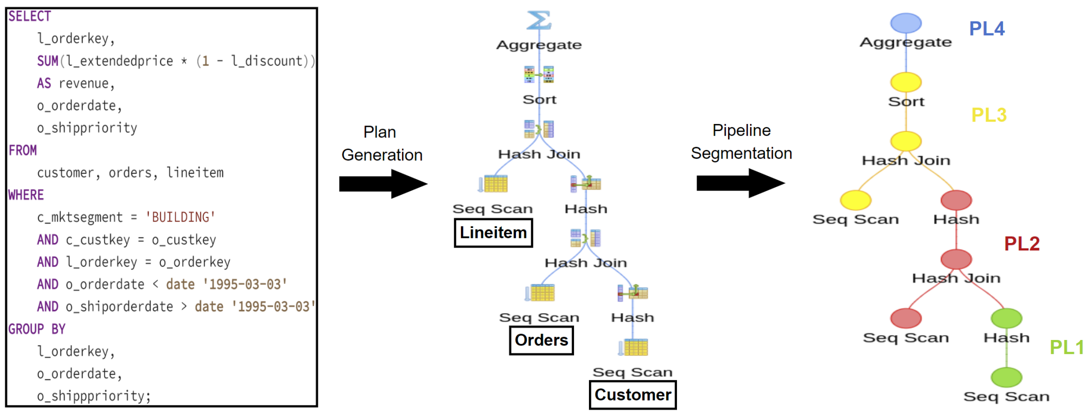
IV. Related Work

This section discusses the main studies devoted to the integration of energy consumption in query processors. These studies may be divided into two main categories: (i) hardware-oriented approaches and (ii) software-oriented approaches.

A. Hardware-oriented Approaches

Several research efforts have been devoted to reducing the energy consumption of the central processing unit (CPU) of storage systems hosting data. Hardware manufacturers are developing EE components and multi-core technologies that enable the processing of higher loads. Modern CPUs have energy-management features defined by the Advanced Configuration and Power Interface (ACPI) [21] that reduces energy consumption by dynamically adjusting the performance state (P-state) based on utilization. A widely adopted technique for reducing the energy consumption of server processors is DVFS (dynamic voltage and frequency scaling). DVFS is typically controlled by frequency governors in the operating system [16], [17]. Based on the DVFS technique, the authors in [22] proposed the Predictive Energy Saving Online Scheduling algorithm in the context of Web search engines. To process a query, PESOS selects the most appropriate CPU frequency that reduces the CPU energy consumption of a query-processing node (i.e., a server dedicated to processing user queries) while respecting tail latency requirements imposed on queries. The work of [23] uses the frequency and/or voltage mechanism (PVC, DVFS) to calibrate energy consumption to satisfy the query performance requirement. To save energy, they execute queries at lower voltage and frequency based on the ability of modern processors, the throughput target, and the workload characteristics (e.g., CPU or I/O intensive queries). In [24], the authors proposed an application that dynamically regulates the clock speed based on response time in order to reduce the energy consumption of OLTP (online transaction processing) in a multi-core environment. This application is able to save 7.6% of total energy consumption when executing TPC-C benchmark queries[3]. In [25], an experimental study was conducted to evaluate the impact of reducing the CPU clock frequency on the energy consumption of common DBMS operators (scans, simple aggregations, and joins). They found that reducing processor frequency significantly improves the EE of main-memory database systems. The work of [17] proposed a frequency selection algorithm called POLARIS (POwer and Latency Aware Request Scheduling) that aims to reduce power consumption while satisfying the required latency of DBMS transactions. This technique controls database transaction scheduling and directly manages the processor frequency based on the execution time of all transactions, including waiting and ongoing transactions, predicted by a transaction execution time model.

[3]https://www.tpc.org/tpcc/



```sql
SELECT
    l_orderkey,
    SUM(l_extendedprice * (1 - l_discount))
    AS revenue,
    o_orderdate,
    o_shippriority
FROM
    customer, orders, lineitem
WHERE
    c_mktsegment = 'BUILDING'
    AND c_custkey = o_custkey
    AND l_orderkey = o_orderkey
    AND o_orderdate < date '1995-03-03'
    AND o_shiporderdate > date '1995-03-03'
GROUP BY
    l_orderkey,
    o_orderdate,
    o_shipppriority;
```

Fig. 4. An example of segmenting a query plan into pipelines.

B. *Software-Oriented Approaches*

Several studies focusing on software aspects have been proposed to integrate the energy dimension in query processors. These efforts mainly concern the development of analytical cost models to characterize the energy consumption of a query or a workload. The main particularity of these models is that they are built on the top of DBMS query optimizers, in the sense that they pick the values of the different costs computed by query optimizers such as CPU, IO, and Memory. In [26], [27], the authors proposed a cost model to predict energy consumption using the response time of a query delivered by PostgreSQL DBMS. This model is dedicated to executing a single query executed without parallelism (inter-query). A model was defined for each basic DBMS operation that estimates its power according to the number of tuples processed by the processor and the number of pages read/written from the disk. The parameter values of this model have been obtained thanks to a simple linear regression technique. The authors in [28] proposed a pipeline-based model to estimate the peak power consumption of an isolated query using a step-wise regression technique. This model is then used to select the query plan with the lowest peak power. It is based on a mathematical function that takes as input the rates and sizes of the data flowing through the pipeline operators and gives an estimate of the peak power consumption. Using the linear regression technique, the authors in [29] proposed an operator-based model to predict the energy consumption of some DBMS operators (select, project, and join). The proposed model is used to choose energy-aware query execution plans while maintaining traditional performance constraints. In [30], the authors proposed a pipeline-based cost model to predict the energy consumption of an isolated query using the polynomial regression technique (non-linear regression). The model is constructed based on the CPU cost and I/O cost of each pipeline. The work in [5] studied the impact of the main memory size and the three main cache structures (Database Buffer Cache, Dictionary Cache, and Library Cache) on various query execution costs (response time, power, and energy). Then, we build a linear cost model that takes the memory cost into account in addition to CPU cost and I/O cost, to predict energy consumption for queries in a sequential execution mode and at operator-level modeling. The values of different parameters of this model are obtained by a simple linear regression technique. In [11], [31], the authors studied the effect of the parallel execution mode on the energy consumption of queries. As an extension of the sequential model in [30], they proposed an energy-aware cost model that uses polynomial regression technique and artificial neural networks to set the values of different parameters. The obtained model showed an improvement of the energy estimation in a multi-core environment.

To summarize the above studies, we figure out the *existence of a gap between hardware and software-oriented approaches*, although they can be easily combined. To bridge the gap, we propose in this paper, an example of the hybridization of hardware and software solutions that enriches the most popular software solution concerning the analytical cost model by hardware DFS technique. One of the main characteristics of this hybridization is that it does not challenge existing solutions, but it assists designers and developers in considering the DFS component when building energy consumption analytical cost models.

V. Energy Modeling Process

In this section, we describe our modeling approach based on machine learning to predict the energy consumption of query execution plans.

A. *Overview of Modeling Process*

Our framework DFSOFT to capture energy consumption when executing a query is based on two phases: training and prediction as shown in Fig. 5 and 6.

In the *training phase*, we build an energy cost model that estimates the energy consumption of queries under different processor configurations. This model is built on top of a DBMS query optimizer. We consider a set of analytical queries for this phase executed by the target DBMS in a sequential way. The final execution plan of each query is produced by the query optimizer in raw format. To get different pipelines

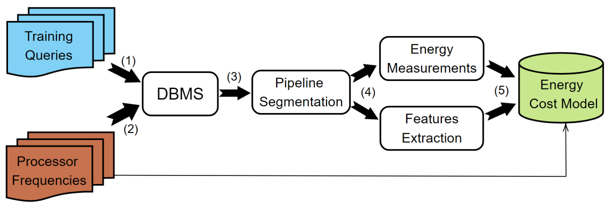


Fig. 5. Energy modeling process: training phase.

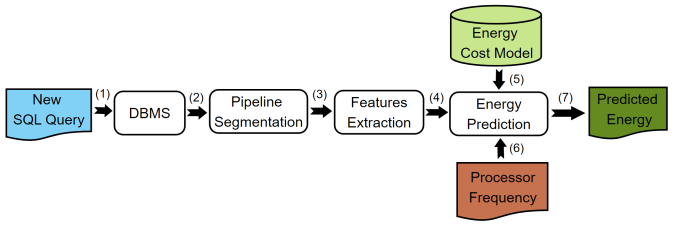


Fig. 6. Energy modeling process: prediction phase.

of this query, the plan is analyzed based on blocking and non-blocking operators. This analysis also allows us to get various data exploitable by our final model: (a) energy measurement of different pipelines, (b) involved parameters, and (c) processor frequency. This data is then stored in a repository. Once the training phase is done, a preparation phase has to be performed on the gathered data. This phase consists of cleaning and normalizing our data by removing duplicate rows, outliers, and rows that contain missing or incorrect values.

The prediction phase cares about the new arrival analytical queries. When a query arrives, the following steps are executed: (i) analyze the query execution plan to obtain its different pipelines, (ii) extract the necessary parameters to perform the estimation, and (iii) predict the energy consumption using our final trained model.

B. *Identification of Energy Sensitive Parameters*

This identification of relevant parameters is made using intensive experiments. They are conducted by executing a workload of simple and complex queries in an isolated way on serial and parallel modes by varying degrees of parallelism, the scale factor of the TPC-H benchmark, and processor frequency configurations. Based on the pipeline mechanism, the execution query plans created by the query optimizer are executed recursively by the executor component to retrieve the set of required tuples. During the execution of pipeline operators four different parameters are collected: (i) the set of instructions executed by the processor. The cost associated to this set represents the CPU cost ($COST_{CPU}$), (ii) the number of data blocks that are loaded from the disk to the main memory. The cost related to this loading is denoted by $COST_{IO}$, (iii) the manipulated main memory blocks. They are associated to a main-memory cost ($COST_{RAM}$), and (iv) the number of CPU cores used to execute the CPU instructions. This number corresponds to the degree of parallelism (N_{CPU}). In this work, we focus on a single node with a multi-core environment.

In addition to the above parameters, the frequency configuration (f_{CPU}) is also considered. It represents the processor frequency used during the execution of the pipeline operators. The frequency values lie in the interval [800 MHz, 3.4 GHz].

Thus, for a given query Q, whose execution plan is composed of n pipelines noted $\{PL_1, PL_2, \ldots, PL_n\}$, let m be the number of relational operators $\{OP_1, OP_2, \ldots, OP_m\}$ composing a pipeline PL_i.

Formally, the power cost associated with the execution of a pipeline PL_i using a frequency setting f_{CPU} is represented by the following equation:

$$\begin{aligned} Power(PL_i) = \beta_{CPU} \cdot \sum_{k=1}^{m} COST_{CPU_k} \\ \oplus \beta_{IO} \cdot \sum_{k=1}^{m} COST_{IO_k} \oplus \beta_{RAM} \cdot \sum_{k=1}^{m} COST_{RAM_k} \\ \oplus \beta_f \cdot f_{CPU} \oplus \beta_N \cdot N_{CPU} \end{aligned} \tag{7}$$

where β_{CPU}, β_{IO}, β_{RAM}, β_f and β_N are the model coefficients (i.e., unit power costs) for the pipelines. $\oplus$ represents the relationship (linear, non-linear) between the parameters of our model. The $COST_{CPU_k}$ is the number of instructions executed by the operator k per second, $COST_{IO_k}$ is the number of data blocks loaded from disk by the operator k per second, $COST_{RAM_k}$ is the number of data blocks retrieved from the main memory by the operator k per second, and N_{CPU} is the number of CPU cores used to execute the pipeline PL_i.

The energy consumption *Energy(Q)* of the query Q is the sum of the energy consumed by the pipelines composing this query. Its represented by the following equation:

$$Energy(Q) = \sum_{i=1}^{n} Energy(PL_i) \tag{8}$$

Similarly to energy, the execution time *Time(Q)* of the query Q is the sum of the execution time of the pipelines composing this query. Its represented by the following equation:

$$Time(Q) = \sum_{i=1}^{n} Time(PL_i) \tag{9}$$

We can distinguish between two types of query execution modes: (i) serial mode and (ii) parallel mode.

- **Serial mode:** where the degree of parallelism is 1 ($N_{CPU} = 1$) which means that pipeline operators are executed using just one processor.
- **Parallel mode:** the pipeline operators can be executed using many processors (N_{CPU} greater than or equal to 2).

In PostgreSQL DBMS (since release 10), parallelism is enabled by default and is controlled by using a configuration parameter called max_parallel_workers_per_gather[4]. This parameter is set to 2 by default.

[4]https://www.postgresql.org/docs/14/how-parallel-query-works.html

Our model parameters ($COST_{CPU}$, $COST_{IO}$, $COST_{RAM}$, N_{CPU}) are extracted from the execution query plans obtained using explain[5] tool of PostgreSQL DBMS. And for getting and changing the processor frequency configuration, we used an Intel DVFS implementation called EIST[6] (Enhanced Intel SpeedStep Technology).

To estimate the values of our model coefficients in (7), we used two machine learning techniques: The support Vector Regression technique and the Random Forest Regression technique.

Support Vector Regression. It is a supervised learning algorithm used in regression problems [32]. It supports both linear and non-linear regressions. Among the possible training parameters for Support Vector Regression, we find the kernel (the kernel type to be used in the algorithm), gamma (the kernel coefficient), C (the regularization parameter), and epsilon value (that determines the width of the tube (hyperplane) within which no penalty is associated in the training loss function with points predicted within a distance epsilon from the actual value).

Random Forest Regression. It is an ensemble learning method for regression [32]. This method operates by constructing a multitude of decision trees (forest) at training time to improve accurate prediction. It is fast to train and it produces good predictions.

C. Training and Testing Datasets

To train and validate our energy cost model, we used the TPC-H benchmark widely used for evaluating analytical complex queries. It consists of eight tables (Part, Partsupp, Region, Supplier, Customer, Lineitem, Nation, and Orders).

For the training set, we generated data at the following scale factors: 10 GB, 30 GB, and 50 GB. Next, we created a workload composed of seventy TPC-H queries divided into two main categories: (i) CPU-intensive queries with operations that exhaust the system processor and (ii) I/O-intensive queries with operations that exhaust the hard disk. Our workload contains queries with a single table scan and queries with multiple joins with different predicates. It also includes sorting/grouping conditions and simple and advanced aggregation functions.

For each scale factor, we collected the query execution plans of our queries and measured the energy consumed by each of them using the power meter. For each query, we repeated the execution processing by varying the number of CPU cores (degree of parallelism) from 1 (serial mode) to 4 (parallel mode), and for each CPU frequency configuration (3.4 GHz, 3 GHz, 2.7 GHz, 2.3 GHz, 1.9 GHz, 1.5 GHz, 1.2 GHz, 0.8 GHz). Note that before starting our experiments, we disable unnecessary background tasks. We also empty the operating system and PostgreSQL buffers before the execution of each query. After collecting our training dataset, we use two machine learning algorithms (SVR and RFR) to obtain the values of our cost model parameters. After that, we consider a dataset with 50 GB and the 22 TPC-H queries and 13 TPC-DS[7] queries (generated using QGEN[8] tool) to validate and measure the accuracy of our cost model. Once our model is validated, energy prediction of new queries is obtained without using the power meter.

[5]https://www.postgresql.org/docs/14/using-explain.html

[6]https://en.wikipedia.org/wiki/SpeedStep

VI. Experimental Evaluation

In this section, we present and discuss the experimental environment of our study and the several experiments conducted in order to evaluate the effectiveness of our energy cost model.

A. Experimental Setup

The main components of the system that we use as a server to run our experiments are as follows: Dell Precision Tower 3620 with Dell 09WH54 motherboard, Intel Core i7-6700 CPU @ 3.4GHz (1 CPU - 4 Cores - 8 Threads), 16 GB Dual Channel DDR4 @ 2133MHz main memory, SSD Disk SM951 NVMe SAMSUNG 256GB. Our server has a Thermal Design Power (TDP) of 65W and can adjust dynamically the CPU frequency using the DVFS technique. A description of the static and dynamic power of major components in our server is provided in Table I. Our experimental environment consists of a database server, a power meter, and a client machine (monitor). The energy consumption measurement of the server is done by using a power meter called Yocto-Watt[9] (produced by Yoctopuce) at a 1Hz frequency. The power meter is placed between the electrical power source and the database server and it is linked via a USB cable to the monitor machine for data collection. In our server, the Operating System used is Ubuntu 20.04 (kernel 20.04.4 LTS) and PostgreSQL (release 14.1) as an open-source row-store DBMS. The experimental platform is shown in Fig. 7. We modified the query optimizer in order to collect more details statistics in the query execution plans.

TABLE I
STATIC AND DYNAMIC POWER CONSUMPTION OF THE MAJOR COMPONENTS OF THE SERVER.

Component	**Static Power (Watt)**	**Dynamic Power (Watt)**
CPU: Intel Core i7-6700 CPU @ 3.4GHz	8.97	42.90
Memory: 16 GB DDR4 @ 2133MHz	2.50	4.68
SSD Disk SM951 NVMe SAMSUNG 256 GB	4.00	6.30

B. Values of Energy Cost Model Parameters

After several observations in which the chosen queries to train our model are executed one after the others, we used two Machine learning techniques (Support Vector Regression, Random Forest Regression) using sklearn[10] tool to determine the values of the parameters for our model represented in (7).

[7]https://www.tpc.org/tpcds/

[8]https://github.com/electrum/tpch-dbgen

[9]https://www.yoctopuce.com/FR/products/yocto-watt

[10]https://scikit-learn.org/stable/

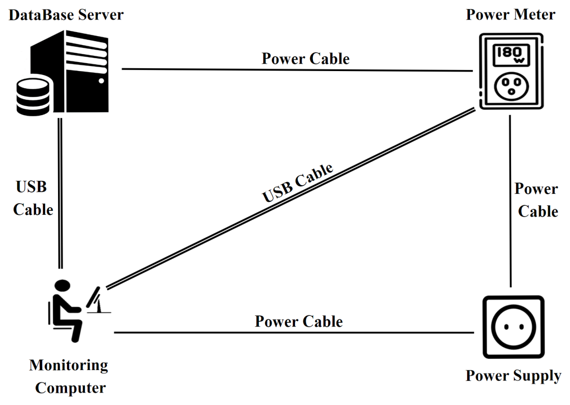


Fig. 7. The deployment environment of our experiments.

The model was trained on all the data collected from the queries. Training parameters were found using the GridSearchCV technique that applied an exhaustive search over the specified parameter values for our model. The best value of each parameter was selected from the following values: *Support Vector Regression (SVR)* model: kernel ('rbf', 'linear'), gamma (1, 0.1, 0.01, 0.001, 1e-4, 1e-7), C (10, 100, 150, 200), epsilon (0.1, 0.2, 0.3, 0.5). *Random Forest Regression (RFR)* model: n_estimators (100, 150, 281, 463, 645, 827, 1009, 1190, 1372, 1554, 1736, 1918, 2100), max_features ('auto', 'sqrt'), max_depth (None, 3, 5, 7, 10, 15, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110), min_samples_split (2, 5, 10, 15, 20), min_samples_leaf (1, 2, 3, 4).

The other parameters are set to default values. Table II shows the best values of training parameters selected after the end of the training phase for the two models.

TABLE II
PARAMETERS VALUES OBTAINED AFTER THE TRAINING PHASE.

ML Model	Training Parameters	Values
RFR	n_estimators	150
	max_features	sqrt
	max_depth	10
	min_samples_split	10
	min_samples_leaf	1
SVR	kernel	rbf
	gamma	0.1
	C	10
	espsilon	0.5

C. Energy Cost Model Validation

In this section, we aim, through several experiments, to validate the accuracy prediction of our energy cost model, described in section V, using the standard queries provided by the TPC-H and TPC-DS benchmarks. We execute the 22 TPC-H queries and 13 TPC-DS queries on a database with a scale factor of 50 GB (SF=50). We measured the actual energy consumed by the entire database server during the query processing and compared it with the energy predicted using our energy cost model. To evaluate the prediction accuracy of our model, we used prediction error rate (PER) as our error metric. The formula defining our error metric is as follows: $PER = \frac{|Actual_Q - Predicted_Q|}{Actual_Q}$, where $Actual_Q$ denotes the real values of power measured by the power meter and $Predicted_Q$ denotes the prediction values of our energy model. We compared our energy cost model (DFSOFT) with two other energy prediction approaches:

(1) **Pipeline-based model (PBM)**: we implemented the pipeline-based cost model proposed in [31]. This model has three parameters that are the number of instructions executed by the processor, the number of data blocks retrieved from disk, and the number of data blocks retrieved from the main memory. The machine learning technique used in this work is Polynomial Regression.

(2) **Operator-based model (OBM):** We also implemented the operator-based cost model proposed in [5]. This model has three parameters that are the number of instructions executed by the processor, the number of multiple read operations retrieved from the hard disk, and the number of data blocks retrieved from the main memory. The machine learning technique used in this work is multiple-linear regression (MLR). Table III presents a comparison between our model and the other two models in terms of parameters used to predict the energy.

TABLE III
PARAMETERS OF OUR MODEL AND OTHER MODELS

Model Parameters	Energy Cost Models		
	PBM	OBM	Our Model
CPU	✓	✓	✓
Disk	✓	✓	✓
RAM	✓	✓	✓
Frequency	x	x	✓
Parallelism	x	x	✓

Next, we present the prediction error rates of power consumption of the TPC-H and TPC-DS queries obtained in a serial mode (parallelism degree fixed to 1) and in a parallel mode, where the parallelism degree is fixed to 2 and 4, using our cost model and two existing ones.

1) Prediction Analysis on TPC-H benchmark

Serial mode. Fig. 8 shows a comparison of the energy estimation errors of the TPC-H benchmark queries obtained using DFSOFT (with SVR and RFR techniques), PBM, and OBM in a serial mode. The difference between the estimated energy using our model and the measured one is very small in almost all cases compared with the other models. For DFSOFT, the RFR model performs better than the SVR model. The average of prediction errors is 2.37%, 1.31%, 14.87%, and 13.93% for DFSOFT-SVR, DFSOFT-RFR, PBM, and OBM respectively.

Parallel mode. Fig. 9 presents a comparison of the energy estimation errors of the TPC-H benchmark queries obtained using DFSOFT (with SVR and RFR techniques), PBM, and OBM for a degree of parallelism fixed to 4. Also, in this mode, we notice that the difference between the estimated energy

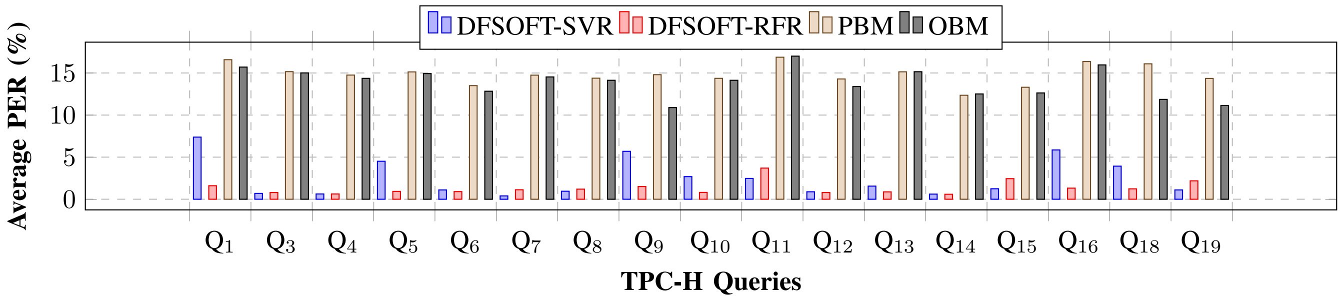


Fig. 8. Energy estimation errors of the TPC-H benchmark queries in a serial mode ($N_{CPU} = 1$) using DFSOFT, PBM and OBM.

using our model and the measured one is very small in almost all cases as well as our model performs better than the other two models. The average prediction errors are this execution mode are 4.65%, 4.44%, 11.4%, and 11.31% for DFSOFT-SVR, DFSOFT-RFR, PBM, and OBM respectively.

As shown in Fig. 8 and 9, our cost model to estimate energy consumption produces a more accurate prediction of power with very small errors in all our experiments and in all execution modes (serial and parallel).
Fig. 10 presents a summary of the average estimation errors obtained by DFSOFT (with SVR and RFR techniques), PBM, and OBM with a degree of parallelism fixed to 1, 2, and 4, and the general case (all modes) that presents the average errors of all the execution modes. This figure shows that DFSOFT outperforms the two existing models PBM and OBM in all execution modes (serial and parallel).

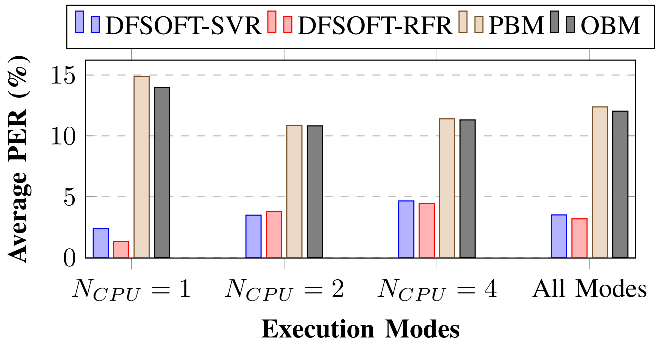


Fig. 10. Energy estimation errors of the TPC-H benchmark queries in serial mode ($N_{CPU} = 1$), parallel mode ($N_{CPU} = 2, 4$) and all modes using DFSOFT, PBM and OBM.

2) *Prediction Analysis on TPC-DS benchmark*

The results of serial mode have been omitted for lack of space. We then concentrate on parallel mode. Our results show that the error increases compared to the one obtained using the TPC-H benchmark. This is due to the complexity of TPC-DS queries.

VII. Conclusion

In this paper, we pointed out the absence of hybrid solutions combining hardware and software aspects in building eco-query processors. To perform this hybridization, we proposed to consider two main popular techniques belonging to hardware and software which are DFS technique and mathematical cost models dedicated to estimating energy consumption when executing workload of queries. To show the impact of DFS, we considered a motivating example with different processor configurations (frequency and number of cores). Based on the encouraging results that we got from this example, we build our accurate and portable energy cost model that aims at predicting the energy consumption of queries in DBMSs in which DFS is a part of the model. This latter takes into account the important features related to the processor ecosystem such as frequency and the number of cores of the processor dedicated to executing queries. In order to identify the coefficient values of our model, we used Support Vector Regression and Random Forest Regression. Finally, we conducted several experiments using the TPC-H and TPC-DS benchmark queries running on PostgreSQL DBMS to validate the proposal and compare it with state-of-the-art energy cost models. Our results show that our model predicts accurately the energy consumption of queries as well as outperforms the other two existing approaches. We believe that these findings can be easily reproducible in the context of machine learning computations.

We are currently studying XGBoost regression and Artificial Neural Networks with much larger training datasets, and deploying our analytical energy consumption model into the PostgreSQL query optimizer to: (i) identify energy-hungry queries and warn users before executing them, and (ii) find the best processor configuration that minimizes the energy consumption.

Acknowledgment

This research is conducted within the framework of the ”Data Connect” project, which is supported by the Région Nouvelle Aquitaine, France.

References

[1] Z. Qiao, S. Liang, N. Damera, S. Fu, H. Chen, and M. Lang, “ACTOR: active cloud storage with energy-efficient on-drive data processing,” in *IEEE BigData*, 2018, pp. 3350–3358.
[2] E. Feller, L. Ramakrishnan, and C. Morin, “On the performance and energy efficiency of hadoop deployment models,” in *IEEE BigData*, 2013, pp. 131–136.
[3] A. J. M. Kell, M. Forshaw, and A. S. McGough, “Exploring market power using deep reinforcement learning for intelligent bidding strategies,” in *IEEE BigData*, 2020, pp. 4402–4411.

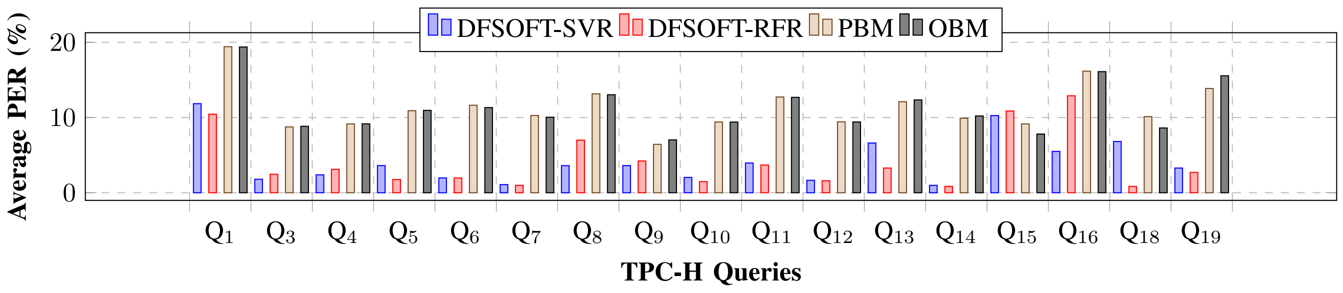


Fig. 9. Energy estimation errors of the TPC-H benchmark queries in a parallel mode ($N_{CPU} = 4$) using DFSOFT, PBM and OBM.

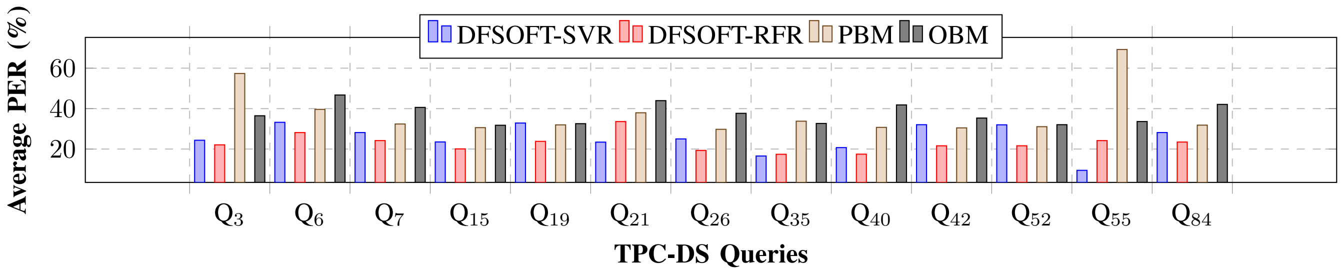


Fig. 11. Energy estimation errors of the TPC-DS benchmark queries in a parallel mode ($N_{CPU} = 4$) using DFSOFT, PBM and OBM.

[4] A. Katal, S. Dahiya, and T. Choudhury, "Energy efficiency in cloud computing data center: a survey on hardware technologies," *Clust. Comput.*, vol. 25, no. 1, pp. 675–705, 2022.

[5] 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.

[6] G. Binglei, Y. Jiong, Y. Dexian, L. Hongyong, and L. Bin, "Energy-efficient database systems: A systematic survey," *ACM Comput. Surv.*, 2022.

[7] E. COMMISSION. (2019) Commission regulation (EU) 2019/424. [Online]. Available: https://eur-lex.europa.eu/legal-content/EN/TXT/?qid=1553786820621&uri=CELEX%3A32019R0424

[8] I. Psaroudakis, T. Kissinger, D. Porobic, T. Ilsche, E. Liarou, P. Tözün, A. Ailamaki, and W. Lehner, "Dynamic fine-grained scheduling for energy-efficient main-memory queries," in *Tenth International Workshop on Data Management on New Hardware (DaMoN)*, 2014, pp. 1:1–1:7.

[9] 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.

[10] I. Ghabri, L. Bellatreche, and S. B. Yahia, "Energy efficiency vs. performance of analytical queries: The case of bitmap join indexes," in *IEEE BigData*, 2021, pp. 3066–3074.

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

[12] 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.

[13] L. Corporation. (2007) Five strategies for cutting data center energy costs through enhanced cooling efficiency. [Online]. Available: http://www.rvip.ru/files/1071/Emerson_EfficiencyWP_low_041707.pdf

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

[15] M. Etinski, J. Corbalán, J. Labarta, and M. Valero, "Understanding the future of energy-performance trade-off via DVFS in HPC environments," *J. Parallel Distrib. Comput.*, vol. 72, no. 4, pp. 579–590, 2012.

[16] V. Pallipadi and A. Starikovskiy, "The ondemand governor: past, present and future," in *Proceedings of Linux Symposium*, 2006, pp. 223–238.

[17] M. Korkmaz, M. Karsten, K. Salem, and S. Salihoglu, "Workload-aware CPU performance scaling for transactional database systems," in *ACM SIGMOD*, 2018, pp. 291–306.

[18] V. Venkatachalam and M. Franz, "Power reduction techniques for microprocessor systems," *ACM Computing Surveys*, vol. 37, no. 3, pp. 195–237, 2005.

[19] S. Chaudhuri, V. Narasayya, and R. Ramamurthy, "Estimating progress of execution for SQL queries," in *ACM SIGMOD*, 2004, pp. 803–814.

[20] J. Li, R. V. Nehme, and J. F. Naughton, "GSLPI: A cost-based query progress indicator," in *IEEE ICDE*, 2012, pp. 678–689.

[21] U. E. Inc. (2016) Advanced Configuration and Power Interface Specification. [Online]. Available: http://www.uefi.org/sites/default/files/resources/ACPI_6_1.pdf

[22] M. Catena and N. Tonellotto, "Energy-efficient query processing in web search engines," *IEEE Trans. Knowl. Data Eng.*, vol. 29, no. 7, pp. 1412–1425, 2017.

[23] W. Lang and J. M. Patel, "Towards eco-friendly database management systems," in *CIDR*, 2009.

[24] Y. Hayamizu, K. Goda, M. Nakano, and M. Kitsuregawa, "Application-aware power saving for online transaction processing using dynamic voltage and frequency scaling in a multicore environment," in *Architecture of Computing Systems*, vol. 6566, 2011, pp. 50–61.

[25] S. Noll, H. Funke, and J. Teubner, "Energy Efficiency in Main-Memory Databases," in *Datenbank-Spektrum*, 2017, pp. 335–344.

[26] Z. Xu, Y. Tu, and X. Wang, "Exploring power-performance tradeoffs in database systems," in *IEEE ICDE*, 2010, pp. 485–496.

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

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

[29] 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.

[30] A. Roukh and L. Bellatreche, "Eco-processing of OLAP complex queries," in *DaWaK*, vol. 9263, 2015, pp. 229–242.

[31] S. P. Dembele, L. Bellatreche, and C. Ordonez, "Towards green query processing - auditing power before deploying," in *IEEE BigData*, 2020, pp. 2492–2501.

[32] D. S. Palmer, N. M. O'Boyle, R. C. Glen, and J. B. O. Mitchell, "Random forest models to predict aqueous solubility," *J. Chem. Inf. Model.*, vol. 47, no. 1, pp. 150–158, 2007.