Azerbaijan Journal of High Performance Computing, Vol 2, Issue 1, 2019, pp.39-63
https://doi.org/10.32010/26166127.2019.2.1.39.63

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

OBRE: Offer-Borrow-Reform-Evaluate Initiatives for Green DBMSs

Amine Roukh[1], Ladjel Bellatreche[2], Nikos Tziritas[3]

[1] *University of Mostaganem, Algeria, roukh.amine@univ-mosta.dz*
[2] *LIAS/ISAE-ENSMA Poitiers, France, bellatreche@ensma.fr*
[3] *Chinese Academy of Sciences Shenzhen, China, nikolaos@siat.ac.cn*

Abstract

In the last few years, we have been seeing a significant increase in research about the energy efficiency of hardware and software components by both academic and industry. Today, energy efficiency is one of the most challenging issues in the area of information technologies and communication. In data-centric applications, database management systems are one of the major energy consumers, in which, a large amount of data is queried by complex queries running daily. Designing and implementing of an energy-aware DBMS that enables significant energy conservation while processing queries become a necessary need. Traditionally, existing DBMSs focus to high-performance during query optimization phase, while totally ignoring the energy consumption of the queries. In this paper, we propose a methodology, supported by a tool called EcoProD, focusing on query optimizers. To show its effectiveness, we implement it in PostgreSQL DBMS aiming reducing energy consumption without degrading query response time. A mathematical cost model is used to estimate the energy consumption. Its parameters are identified by a machine learning technique. We conduct intensive experiments using our cost models and a measurement tool dedicated to compute energy using dataset of TPC-H benchmark. Based on the obtained results, a probabilistic proof to demonstrate the confidence bounds of our model and results is given.
CCS Concepts: Information systems - Relational database model; DBMS engine architectures; Database query processing; Relational database query languages;

Keywords: Database Design; Query Proc essing; Energy Efficiency

1. Introduction

The COP21 event shows the willingness of countries (Over 145 foreign Heads of State and Government attended the conference at Le Bourget, Paris), companies, individuals, government and non government associations, etc. to save the planet. According to the 2009 Climate Action Plan, electricity is one of the two largest sources of greenhouse gas (GHG) emissions for the campus and Information Technology (IT) is currently estimated to be responsible for approximately 10 percent of that electricity usage. IT has become a critical resource for the mission of the campus and usage of computing equipment continues to increase.

The continued expansion of the industry means that the energy use by data centers, and the associated emissions of greenhouse gases and other air pollutants, will continue to grow. Industry experts, such as the SMARTer 2020, reports that global data center emissions will grow 7 percent year- on-year through 2020 [5]. In a typical data center, DBMS is one of the most important consumers of computational resources among other software deployed, which turn DBMS to be a considerable energy consumer [15]. Traditionally, the design process of a database considers one non-functional requirement, which represents the query response time. This requirement is quite comprehensive since, the end user and decision makers of database applications are looking for the efficiency of the queries. Note that in the Beckman report on databases published in last February, energy constrained processing and scientific data management are considered as challenging issues [1].

Face to the strong requirement of saving energy, database community did not stand idly, but from last decade, it continuously proposes initiatives around OBRE actions (Offer, Borrow, Reform, Evaluate) to deal with energy.

- Offer: the database technology was made available for energy professional for the analysis usage to enable smarter scheduling of energy consumption of entities such as smart cities (in MIRABEL project [21]) and electrical vehicles in EDF (Elecricity De France) project [20].

Borrow: the database technology employs green hardware and platforms to deploy the target database applications.

- Reform: the database community did several efforts in reforming their software to integrate energy. These efforts concern mainly the development of cost models to estimate energy and then use them to generate query plans [24, 10, 11] and select optimization structures such as materialized views [19].
- Evaluate: to test their efficiency and effectiveness, energy initiatives have to be evaluated either using real datasets or benchmarks.

These initiatives have shown their performance in reducing energy consumption. Offering strategic guidelines in terms of competency, knowledge, reuse, etc. may contribute in boosting researchers and industrials to intensively integrate energy

during the process of building their applications and DBMS.

Note that the landscape of DBMS is very large since it includes several components: query optimizer, storage manager, etc. In this paper, we focus on query optimizers, which represents one of the main components of DBMS. There has been extensive work in query optimization since the early'70 in traditional databases. Several algorithms and systems have been proposed, such as System-R project, where its findings have been largely incorporated in many commercial optimizers. Advanced query optimizers perform two main tasks: (i) enumeration of execution plans of a given query and (ii) the selection of the best plan. This selection uses cost-based optimization (CBO). The CBO is a mathematical processor, where it uses formulas to calculate the cost of each execution plan. This cost may be the number of inputs outputs required for executing a given execution plan. A CBO approach is suitable when statistics on tables, indexes, selectivity factors of join and selection predicates, etc. are available. The existing studies on energy-aware query optimizes consider mainly the second task, by reforming the cost models by integrating energy.

In this paper, we focus on the query optimization component of the PostgreSQL DBMS. We propose a design methodology, supported by a tool called EcoProD, that tries to integrate energy in the query generation phase. This is done by revisiting all the query optimizer steps and studying their effect on energy consumption. The new query optimizer will have to deal with two objectives functions, namely: improving performance and minimizing energy. In our design, the end users can specify preferences in their profiles by setting weights on different objectives, representing relative importance. The role of the EcoProD is to minimize the weighted sum over different cost metrics.

The main technical contributions of this paper are:

- A deep classification of existing solutions for minimizing energy by the means of two approach: (i) the hardware approaches, (ii) and the software approaches;
- a multi-objective formalization of the query optimization problem including the query performance and the energy consumption;

a demonstration tool, EcoProD, developed in PostgreSQL that gives to end users execution plans and energy consumption by their queries using comprehensive GUI;

- intensive experiments using real tools to study the effectiveness of our approaches.
- a probabilistic proof is given to demonstrate the confidence bounds of our model data and results, using high-end configuration experimentation data.

The rest of this paper is organized as follows. We summarize the most important studies based on OBRE principle in Section 2. In Section 3, we give more details on undertaken energy initiatives from software and hardware perspectives. Section 4.6 describes our green query optimizers, by detailing all its components. Section 5 presents and interprets our experimental results. Section 6 gives a probabilistic complexity study to demonstrate the confidence bounds of our finding, while our

conclusions are given in Section 7.

2. Related work

The most existing studies will be discussed according our OBRE principle.

Offer. As we said in the Introduction, the database technology is used in the past and now to store energetic data from vehicles, smart cities, etc. The *MIRABEL Project* [21] is an example of this direction. It consists in developing an approach on a conceptual and an infrastructural level that allows energy distribution companies balancing the available supply of renewable energy sources and the current demand in ad-hoc fashion. It uses a DBMS to store forecasting in order to answer time series queries, which is an important functionality in energy data management [3, 14].

Borrow. As other technology, databases never stop borrowing green hardware and platforms for their applications and DBMS [9].

Reform. This aspect consists in reforming existing software components to minimize their energy use. They concern mainly two aspects: (1) the definition of cost models to predict energy and (2) the proposition of optimization techniques to reduce energy.

Energy cost models.

Prior works have been concentrated on building power cost models to predict query power consumption. In [24, 23], the authors discussed the opportunities for energy-based query optimization, and a power cost model is developed in the conjunction of PostgreSQL's cost model to predict the query power consumption. A static power profile for each basic database operation in query processing is defined. The power cost of a plan can be calculated from the basic SQL operations, like CPU power cost to access tuple, power cost for reading/writing one page, and so on, via different access methods and join operations using regression techniques. The authors adapt their static model to dynamic workloads using a feedback control mechanism to periodically update model parameters using real-time energy measurements. The authors of [10] propose a technique for modeling the peak power of database operations. A pipeline-based model of query execution plans was developed to identify the sources of the peak power consumption for a query and to recommend plans with low peak power. For each of these pipelines, a mathematical function is applied, which takes as input the rates and sizes of the data flowing through the pipeline operators, and as output an estimation of the peak power consumption. The authors used piece-wise regression technique to build their cost model. In the same direction, the work of [11] proposes a framework for energy-aware database query processing. It augments query plans produced by traditional query optimizer with an energy consumption prediction for some specific database operators like select, project and join using linear regression technique. [16] attempts to model energy and peak power of simple selection queries on single relations using linear regression. In our previous works [18], we proposed cost models to predict the power consumption of single and concurrent queries. Our model is based on pipeline

segmenting of the query and predicting their power based on its Inputs- outputs (IO) and CPU costs, using polynomial regression techniques.

Optimization techniques.

The presence of energy consumption cost models motivate the research community to propose cost-driven techniques. The work in [12] proposed an Improved Query Energy-Efficiency (QED) by Introducing Explicit Delays mechanism, which uses query aggregation to leverage common components of queries in a workload. The work of [11] showed that processing a query as fast as possible does not always turn out to be the most energy-efficient way to operate a DBMS. Based on their proposed framework, they choose query plans that reduce energy consumption. In [10], cost-based driven approach is proposed to generate query plans minimizing the peak power. In [19], genetic algorithm with a fitness function based on a energy consumption cost model, is given to select materialize views reducing energy and optimizing queries. The work by Xu et al. [23] is close in spirit to our proposal in this paper. They integrate their cost model into the DBMS to choose query plans with a low power at the optimization phase. However, they do not study the consumed energy at each phase of query optimizers. Moreover, they use a simple cost model that do not capture the relationship between the model parameters.

Evaluate. Energy evaluation is a sensitive point and as a consequence it requires accurate and transparent evaluation to show its savings. For transparent perspective, we propose an open platform available at the forge of our laboratory, allowing researchers, industrials and students to evaluate it.

3. Initiatives for integrating energy in DBMS

The proposed initiatives covers hardware and software. This categorization is illustrated in Figure 1.

3.1. Hardware solutions

Hardware efforts towards green databases focus on using devices designed with low energy consumption, or controlling the power mode of hardware by doing a transition from high-power state to low-power state when the system is not active (idle mode), is this also known as energy- proportionality [7]. Most modern hardware such as processors, main memory, and disks come with this technology.

Processing device.

The processing device such as CPU is one of the main active power consumer [15]. Since the CPU is power-proportional hardware, significant power can be saved using techniques like Dynamic Voltage and Frequency Scaling (DVFS) with less performance degradation. This has been well studied and verified in the stat-of-arts works [22, 12]. The use of Graphics Processing Units (GPUs) in the database has been shown to incur significant performance benefits. A recent study reported that using a GPU is more energy efficient when the performance improvement is above a

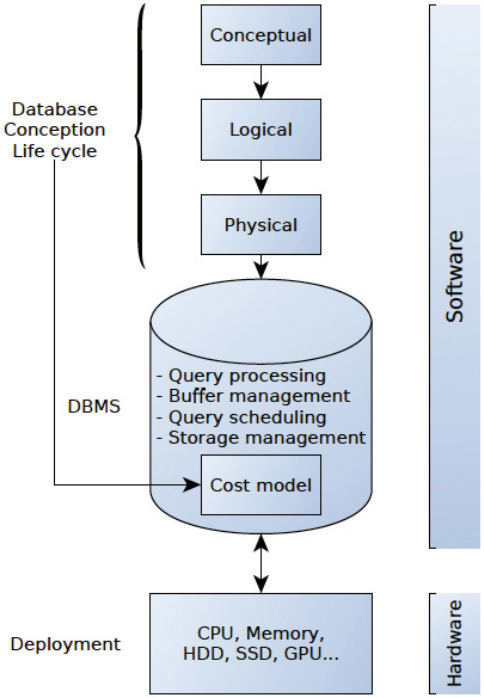

Fig. 1: Energy integration levels in DBMS

certain bound, compared to a CPU-only solution [17].

Storage management.

Making storage management systems green in the context of database start to make it appearance. Switch disks to stand-by mode lead to less energy consumption compared to active mode. Other works use caching and perfecting techniques or consolidating the most frequently accessed data via dynamic power management algorithm to save energy [22]. Using solid state disks (SSDs) as a storage device by databases to improve energy-efficiency is another worthy direction. Studies such as [4] claim that using Smart SSD can be benefit from both the performance and the energy consumption perspectives.

3.2. Software solutions

On the other hand, software-based solutions play an important role in energy optimization. The basic idea is to redesign current algorithms and software applications for better energy use.

Cost model.

Perhaps the most important task in this category is the design and the implementation of an accurate energy cost model. Cost models are used to estimate

the energy consumption of a query plan or an optimization structure. Since they are used by other software techniques to reduce energy, they shall be accurate enough to give better results. The crucial issues have to be discussed: (i) the identification of the relevant parameters that have an impact on energy, such as CPU, I/O and communication costs and (ii) the relationship between these parameters (e.g: linear/non-linear). They are usually calculated as the product of a basic operator such as the number of tuples, disk pages, and network messages.

Query optimization.

The most used non-functional requirement by query optimizers represents the workload performance. Choosing query plans with low-energy consumption without sacrificing the performance was the motivation of several research studies [23, 10, 11]. Their main results show the existence of power-performance trade-off in database query optimization. Up to 22% total power saving has been reported in [23].

Buffer management.

New caching and replacement policies will be needed to reflect energy costs for accessing and storing data in memory. Usually, queries have some common components, such as common subexpressions, we can use multi-query optimization techniques to optimize the workload. This technique can be exploited further to improve the average per-query energy consumption [12].

Physical design.

Several studies have recommended the integration of energy in the physical design of database [6, 8]. Recently, in [19], a energy-driven method for selecting materialized views leads to up to 38% total power saving.

In this work, we consider software approach, specifically the query optimization technique to incorporate the energy dimension.

4. Our green-query optimizer

In order to build energy-aware query optimizers, we first propose an audit of each component to understand whether it is energy-sensitive or not. After our audit, we present in details our methodology to construct our optimizer.

4.1. An Audit of Query Optimizers

Recall that a query optimizer is the responsible to execute queries respecting one or several non functional requirements such as response time. The process of executing a given query passes through four main steps: (i) parsing, (ii) rewriting, (iii) planning and optimizing and (iv) executing (cf. Figure 2). To illustrate these steps, we consider PostgreSQL DBMS as a case study.

4.2. Parse

The parser has to check the query string for valid syntax using a set of grammar rules. If the syntax is correct a *parse tree* is built up and handed back. After the parser completes, the transformation process takes the parse tree as input and does the semantic interpretation needed to understand which tables, functions, and operators are referenced by the query. The data structure that is built to represent this information is called the *query tree.* The cost of this phase is generally ignored by the DBMS since its finish very quickly. We follow the same logic and suppose that the energy consumption is negligent.

4.3. Rewrite

The query rewrite processes the tree handed back by the parser stage and it rewrites the tree to an alternate using a set of rules. The rules are system or user defined. This rules-based phase is also used in materialization views query rewriting. As for the previous step, the cost is ignored due to the fast completion.

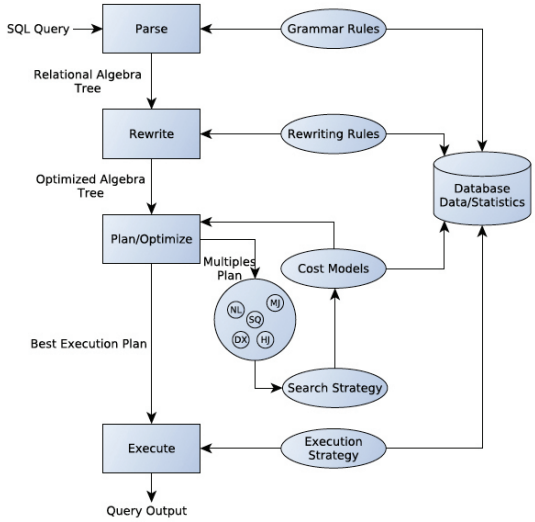

Fig. 2: Query optimizer steps

4.4. Plan/Optimize

The task of the planner/optimizer is to create an optimal execution plan. A given SQL query can be actually executed in different ways, each of which will produce the same set of results. The optimizer's task is to estimate the cost of executing each plan using a cost-based approach and find out which one is expected to run the fastest.

4.4.1. Plan.

The planner starts by generating plans for scanning each individual relation

(table) used in the query. The possible plans are determined by the available *indexes* on each relation. There is always the possibility of performing a *sequential scan* on a relation, so a sequential scan plan is always created. If the query requires joining two or more relations, plans for joining relations are considered after all feasible plans have been found for scanning single relations. The available join strategies are: *nested loop join*, *merge join*, *hash join.* When the query involves more than two relations, the final result must be built up by a tree of join steps, each with two inputs. The planner examines different possible join sequences to find the cheapest one. If the query uses less than a certain defined threshold, a near-exhaustive search is conducted to find the best join sequences, otherwise, a heuristics based genetic algorithm is used.

To study the effects of such searching strategies, let us consider the query *Q8* of the TPC-H benchmark. This a complex query which involves the join of 7 tables. We modify the planner of PostgreSQL in three manners: (i) searching for plan with actual DBMS strategy (ii) using the genetic algorithm, and (iii) manually by forcing the planner to choose a certain plan. For each strategy, we calculate its execution time, and the total energy consumption during query execution against 10GB datasets. Results are presented in Table 1.

Table 1: Planning step for TPC-H Q8 with different searching strategies.

Search Algo	Planning Time (s)	Energy (j)
Default	0.110006	5200.362
GA	0.977013	5387.648
Manual	0.092054	5160.036

From the table, we can see that setting the query plan manually gives the betters results, in both time and energy. While the default searching algorithms (semi-exhaustive) leads to a slightly more execution time and energy consumption. The genetic algorithm gives the worst results in this example, perhaps due to the small number of tables in the query, since this strategy is used by the DBMS where there is more than 12 tables. Considering this small number of tables, if we go in real operational databases where there is a hundred of tables, the searching strategy used by the planner can leads a noticeable energy consumption. Setting the query plan of queries manually by the database administrator is recommended in large databases to gain in energy efficiency.

4.4.2. Optimize.

To evaluate the response time for each execution plan, cost functions are defined for each basic SQL operator. The general formula to estimate the cost of operator *op* can be expressed as:

$$Cost_{op} = \alpha \times I/O \oplus \beta \times CPU \oplus \gamma \times Net \quad (1)$$

Where *I/O*, *CPU* and *Net* are the estimated pages numbers, tuples numbers,

communication messages, respectively, required to execute *op.* They are usually calculated using database statistics and selectivity formulas. The coefficients *a/3* and 7 are used to convert estimations to the desired unit (e.g: time, energy). O represents the relationship between the parameters (linear, non-linear). The coefficients parameters and relationship can be obtained using various techniques such as calibration, regression and statistics. Thus, energy cost model must be defined at this stage with the relevant parameters.

The finished plan tree consists of sequential or index scans of the base relations, plus nested-loop, merge or hash join nodes as needed, plus any auxiliary steps, such as sort nodes or aggregate-function calculation nodes.

4.5. Executor

The executor takes the plan created by the planner/optimizer and recursively processes it to extract the required set of rows. This is essentially a demand-pull *pipeline* mechanism. Each time a plan node is called, it must deliver one more row, or report that it is done delivering rows. Complex queries can involve many levels of plan nodes, but the general approach is the same: each node computes and returns its next output row each time it is called. Each node is also responsible for applying any selection or projection expressions that were assigned to it by the planner.

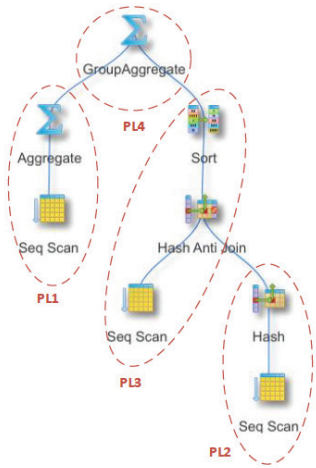

Fig. 3: execution plan of TPC-H benchmark query Q22 with corresponding pipeline annotation.

To study the effect of the execution step on designing green-query optimizer, we consider an example of query *Q22* from the TPC-H benchmark. Figure 3 presents the execution plan returned by the PostgreSQL query optimizer. As we shown in [18], the power consumption is directly influenced by execution model of the DBMS. Therefore, the execution plan can divided into a set of segments, we refer to these segments as *pipelines*, the pipelines are the concurrent execution of a contiguous

sequence of operators. The pipeline segmentation of the optimizer plan for query *Q22* is shown in Figure 3, there are 4 pipelines, and a partial order of the execution of these pipelines is enforced by their terminal *blocking* operators (e.g., PL3 cannot begin until PL2 is complete).

In our previous study, we showed that *when a query switches from one pipeline to another, its power consumption also changes.* During the execution of a pipeline, the power consumption usually tends to be approximately constant [18]. Therefore, the pipelining execution is very important and has a direct impact on power consumption during query execution. The design of power cost model should take into consideration the execution strategy, which is unfortunately ignored by Xu *et al.* [23].

4.6. Our Methodology

In this section, we describe the design and the implementation of our proposal into PostgreSQL database. As we mentioned above, the planner/optimizer and the executor stages have an impact on energy consumption and should considered in designing any green-query optimizer. We extended the cost model, the query optimizer and the communication interface of PostgreSQL to include the energy dimension.

Inspired by the observation made in the previous section, we designed our cost-based power model. The basic idea of this model is to decompose an execution plan into a set of power independent pipelines delimited by blocking/semi- blocking operators. Then for each pipeline, we estimate its power consumption based on its CPU and I/O cost.

The work-flow of our methodology is described in Figure

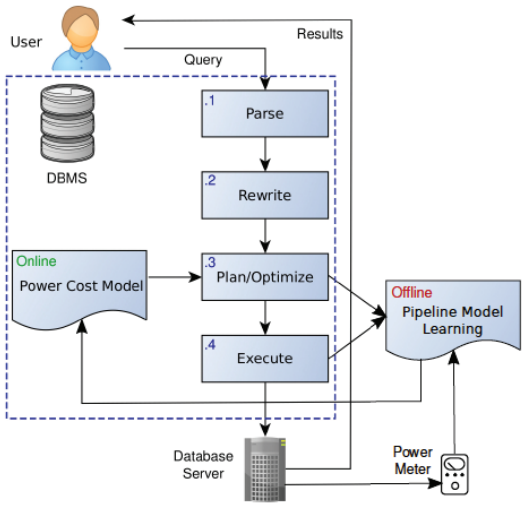

Fig. 4: The Design Methodology

4.7. Power Cost Model

In this section, we present our methodology for estimating energy consumption. The characteristics of our model include: (i) the segmentation of an execution plan into a set of pipelines, (ii) the utilization of the pipeline cost to build the regression model, and (iii) the estimation of the power of future pipeline based on pipeline cost and the regression equation.

4.7.1. Pipeline Segmentation.

When a query is submitted to the DBMS, the query optimizer chooses an execution plan (cf. Figure 3). A physical operator can be either *blocking* or *nonblocking.* An operator is blocking if it cannot produce any output tuple without reading as least one of its inputs (e.g., sort operator). Based on the notion of blocking/nonblocking operators, we decompose a plan in a set of pipelines delimited by blocking operators. Thus, a pipeline consists of a set of concurrently running operators [2]. As in previous work [2], the pipelines are created in an inductive manner, starting from the leaf operators of the plan. Whenever we encounter a blocking operator, the current pipeline ends, and a new pipeline starts. As a result, the original execution plan can be viewed as a tree of pipelines, as showed in Figure 3.

4.7.2. Model Parameters.

Given a certain query, the query optimizer is responsible for estimating CPU and I/O costs. Our strategy for pipeline modeling is to extend the cost models that are built into the PostgreSQL database systems for query optimization. To process a query, each operator in a pipeline needs to perform CPU and/or I/O tasks. The cost of these tasks represents the "cost of the pipeline", which is the active power to be consumed in order to finish the takes. In this paper, we focus on a single server setup and leave the study of distributed databases as future work. Thus, the communication cost can be ignored. More formally, for a given query Q composed of p pipelines $\{PL_1, PL_2, \ldots, PL_p\}$. The power cost $Power(Q)$ of the query Q is given by the following equation:

$$Power(Q) = \frac{\sum_{i=1}^{P} Power(PL_i) \cdot Time(PL_i)}{Time(Q)} \quad (2)$$

The *time* variable represents the pipelines and the query estimated time to finish the execution. Unlike Xu *et al.* study which ignore the execution time [24], in our model, the time is an important factor in determining the CPU or I/O dominated pipeline in a query. The DBMS statistics module provide us with this information. Let a pipeline PLi composed of n algebraic operations $\{OP_1, OP_2, \ldots, OP_n\}$. The power cost $Power\ (PLi)$ of the pipeline PLi is the sum of CPU and I/O costs of all its operators:

$$Power(PL_i) = \beta_{cpu} \sum_{j=1}^{n_i} CPU\_COST + \beta_{io} \sum_{j=1}^{n_i} IO\_COST_j \quad (3)$$

Where β_{cpu} and β_{io} are the model parameter (i.e., unit power costs) for the

pipelines. For a given query, the optimizer uses the query plan, cardinality estimates, and cost equations for the operators in the plan to generate counts for various types of I/O and CPU operations. It then converts these counts to time by using system-specific parameters such as CPU speed and I/O transfer speed. Therefore, in our model, we take I/O and CPU estimations already available in PostgreSQL before converting it to time. The *IO-COST* is the predicted number of *I/O* it will require for DBMS to run the specified operator. The *CPU-COST* is the predicted number of *CPU Tuples* it will require for DBMS to run the specified operator.

4.7.3. Parameters Calibration.

The key challenge in equation (3) is to find model parameters β_{cpu} and β_{io}. Simple linear regression technique, as used in [24, 10, 11], did not work well in our experiments, especially when data size change, this is because the relationships between data size and power are not linear. In other words, processing large files does not *always* translate in high power consumption. It depends more on the type of queries (I/O or CPU intensive) and their execution time. Therefore, we employed

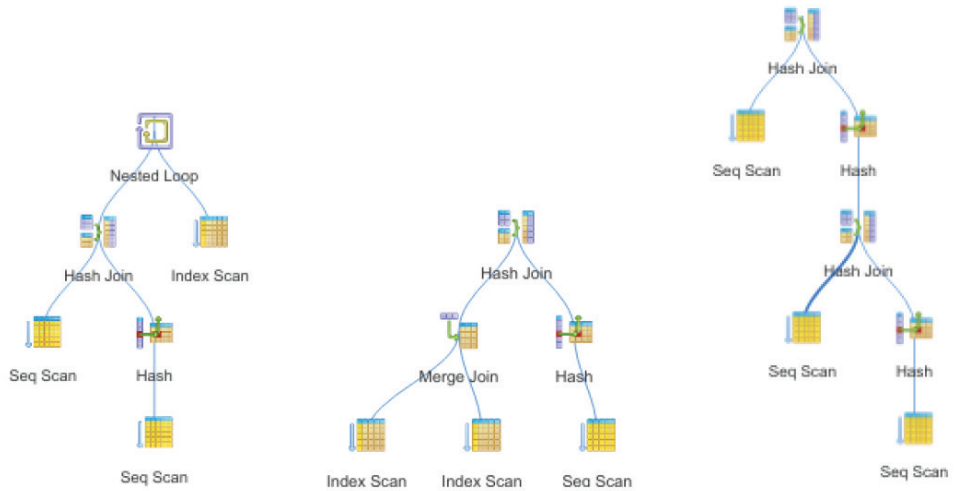

Figure 5: The optimal plan for TPC-H query Q3 when changing user preferences.

(a) Performance oriented plan *(b) Power oriented plan* *(c) Performance/power traed-o oriented plan*

multiple polynomial regression techniques. This method is suitable when there is a *nonlinear* relationship between the independents variables and the corresponding dependent variable. Based on our experiments, the order *m=4* gives us the best results (the residual sum of squares is the smallest). The power cost *Power (PLi)* of the pipeline *PLi* is computed as:

$$Power(PL_i) = \beta_0 + \beta_1(IO\_COST) + \beta_2(CPU\_COST) + \beta_3(IO\_COST^2) + \beta_4(CPU\_COST^2) + \beta_5(IO\_COST \cdot CPU\_COST) + \cdots + \beta_{13}({IO_{COST}}^4) + +\beta_{14}(CPU\_COST^4) + \varepsilon \quad (4)$$

Where *IO-COST*, *CPU-COST* denote the pipeline I/O and CPU costs respectively, these costs are calculated using the DBMS cost model functions, and $\mathbf{e}$ is a noise term that can account for measurement error. The β parameters are regression coefficients that will be estimated while learning the model from training data. Thus, the regression models are solved by estimating the model parameters β, and this is typically done by finding the least-squares solution [13].

4.8. Plans Evaluation

The query optimizer evaluates each possible execution path and takes the fastest. Adding energy criterion, we must adjust the comparison functions to reflect the tradeoffs between energy cost and processing time. In order to give the database administrator a solution with the desired trade-off, we propose to use the weighted sum of the cost functions method. In this scalarization method, we calculate the weighted sum of the cost functions so as to aggregate criterion's and have an equivalent single criterion to be minimized. This method is defined as follows:

$$y = f(x) = \sum_{i=1}^{k} w_i f_i(\vec{x}) \tag{5}$$

$$\sum_{i=1}^{k} w_i = 1$$

Where $\mathbf{w_i}$ are the weighting coefficients representing the relative importance of the k cost functions. $f(x)$ represents power cost function and performance cost function respectively. We implemented these two coefficients as an external parameter in the DBMS, so the database administrator or users can change them in the fly.

Figure 5 shows the optimal query plan returned by the modified query planner/optimizer for TPC-H query *Q3* and how it changes when user preferences vary. Initially, we used a performance only optimization goal, the total estimated cost is 371080 and the estimated total power is 153. Changing the goal to be only power, the estimated cost increased to 626035 but the power cut down to 120. In the trade-off configuration, the estimated cost is 377426 and the power is 134. In Figure 5a the nested loop operator draws the high amount of power in the query (33 watts) but the plan is chosen by the optimizer because it is very fast. In Figure 5b, we realise that the merge join operator is the slowest in query, its estimated cost is 539200 but the power is minimal. The two hash join operators used in Figure 5c give a good trade-off, for a 1.7% of performance degradation, we get 12.4% of power saving.

4.9. EcoProD GUI

In this section, we describe the graphical user interface part of EcoProD. The GUI helps manipulating EcoProD, changing parameters and showing in real time their impact on the power consumption.

The EcoProD GUI interface is used to facilitates users manipulating the framework settings and seeing their effect on the system. The interface is implemented using C+-b programing language and Qt library. Figure 6 gives an overview of the main

GUI, which comprises several component modules:

4.9.1. Configuration.

This module is responsible for the connexion establishment with the DBMS server. Users can also specify the path for the power meter driver in order to capture realtime power consumption. The most important part here is the power/performance settings, which decide the optimization goals to be performance or power oriented.

4.9.2. SQL Query.

In this module, users can give their SQL query to be executed. Queries supported varies from simple transactional operations to very complex reporting operations involving many tables with large data size. The execution is done in a separate thread and the results are displayed in a tree table widget.

4.9.3. Power Time-line.

When the user execute a query, EcoProD dynamically displays via the power meter the real time power consumption. After the query finished executing, the total energy that has been consumed during query execution time is computed and showed. This can gives users a real observation of the energy that has been saved using the desired trade-off parameters. Also, users can compare between the estimated and the real values to check model accuracy or further refine it.

4.9.4. Execution Plan.

When the user submit a query, the query optimizer will select their best execution plan in respect to the pre-defined trade-off. The execution plan is displayed with various informations, such as estimated cost, power consumption, I/O and CPU costs for every physical operator through mouse-hovering events. Also, the pipeline notation is demonstrated, as shown in Figure 6 (4), the pipeline trees are grouped with the same color. The GUI shows how the trade-off parameters affect the generated plan. Thus, we can help users better understand and interpret runtime optimization informations and pipeline notation.

5.Experiments and results

To evaluate the effectiveness of our proposal, we conduct several experiments. Next we present our experimental machine to compute the energy and the used datasets and simulator.

5.1. Experiment Setup

We use a similar setup used in the state-of-arts [11, 24, 10]. Our machine is equipped with a “Watts UP? Pro ES” power meter with one second as a maximum resolution. The device is directly placed between the power supply and the database workstation under test to measure the workstation’s overall power consumption. The

Fig. 6: EcoProD main GUI and its component module panels.

power values are logged and processed in a separate monitor machine.

We used a Dell PowerEdge R310 workstation having a Xeon X3430 2.40GHz processor and 32GB of DDR3 memory. To validate our model using di_erent low-end hardware con_guration, we created another setup with a Dell Precision T1500 workstation equipped with an Intel Core i5 2.27GHz processor and 4GB of memory. Our workstation machine is installed with our modi_ed version of PostgreSQL 9.4.5 DBMS under Ubuntu 14.04 LTS with kernel 3.13. We use TPC-H datasets and queries with 10GB and 100GB scale factor. The TPC-H benchmark illustrates decision support systems that examine large volumes of data, execute di_erent types of queries with a high degree of complexity. The queries are executed in an isolated way. In our experiments, we consider three types of PostgreSQL con_guration: (1)

Power-PG, which is the con_guration that gives the minimal power cost, (2) Time-PG, is the con_guration with minimal time cost, (3) Tradeo_-PG, using weighted sum method with $\omega_1 = 0.5, \omega_2 = 0.5$

5.2. Power Model Building

As mentioned above, the _ parameters are estimated while learning the model from training data. We then perform series of observations in which queries are well-chosen, and the power values consumed by the system are collected using measuring equipment while running these queries. In the same time, for each training instance, we calculate their costs. To generate training instances, we create our

custom query workload based on TPC-H datasets. The workload containing queries divided into two main categories: (i) queries with operations that exhaust the system processor (CPU intensive queries) and (ii) queries with exhaustive storage subsystem resource operations (I/O intensive queries). Note that the considered queries include: queries with a single table scan, queries with multiple joins with different predicates. They also contain sorting/grouping conditions and simple and advanced aggregation functions as in [10]. After collecting power consumption training queries, we apply the regression equation (4) using the R language software[9] to find our model parameters. Once we get them, an estimation of new queries is obtained without the use of our measurement equipment.

5.3. Results

In this section, we present the results of our various experiments.

5.3.1. Cost Model Quality.

The results of the training phase in our two setup con_gurations, against the _tted values from polynomial models (cf. Equation 4) are plotted in Figure 7. As we can see, the predicted and actual power consumption approximate the diagonal lines closely using our cost model in both congurations. Otherwise, in the server conguration we can see some variance between the predicted and the observed power for some training queries. Much of this can be attributed to the errors made by the DBMS query optimizer in estimating IO and CPU costs for these queries when there is a large working memory. This problem has been faced by query optimizers for a long time, and all the performance models proposed so far suffer from this problem, which is inherited from the cardinality estimation errors. In fact, the estimation errors in the low levels pipeline are propagated to the upper level and may significantly degrade the prediction accuracy.

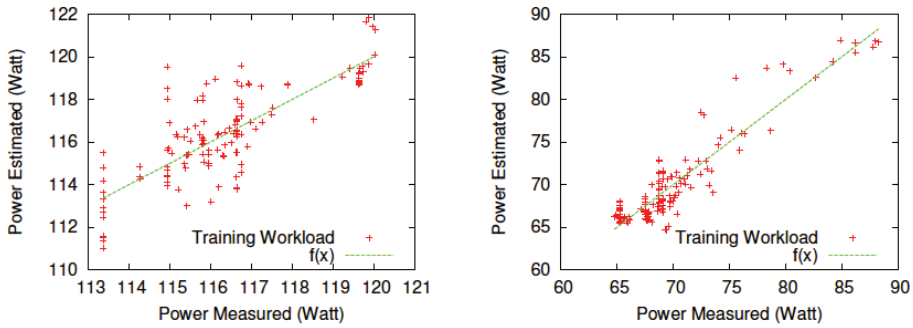

Fig. 7: Training workload power consumption and regressions t.
(a) Regression Model t using high-end conguration
(b) Regression Model t using low-end conguration

Table 2: Estimation errors in TPC-H benchmark queries with different database sizes.

Query	10GB	100GB	Query	10GB	100GB
Q1	0.01	0.002	Q11	0.04	-
Q2	-	-	Q12	0.009	0.00029
Q3	0.01	0.01	Q13	0.04	0.04
Q 4	0.006	0.005	Q14	0.02	0.02
Q5	0.01	0.03	Q15	0.004	0.02
Q6	0.04	0.02	Q16	0.05	0.0003
Q7	0.004	0.01	Q18	0.004	-
QS	0.0007	0.01	Q19	0.01	0.009
Q10	0.006	0.003	Q22	0.01	0.004

5.3.2. Cost Model Estimation Error.

In this type of experiment, given the estimated power cost predicted by our model (E)., *we* compare it with the actually observed system active power consumption (M). To quantify the model accuracy, we used the following error ratio metric:

$$error = \frac{|M - E|}{M}$$

To test our model with large datasets, we run all 22 queries of the TPC-H benchmark against two database scale factor: 10GB and 100GB. Most of the queries contain more than 4 pipelines. The results are shown in Table 2. We not that some queries were aborted since they exceeded 72 hours of execution in our current test environment.

As we can see from the table, the average error is typically small (0.1% in both 100GB and 10GB datasets), and the maximum error is usually below 5%. The experiment shows the accuracy of our prediction model, indicating that is sufficiently accurate for the intended applications.

5.3.3. Query Characterization.

To study the characterization of the TPC-H 22 query, we conduct a series of tests using the modified PostgreSQL. In such tests and for every configurations (Time-PG, Power- PG, Tradeoff-PG) we run all the TPC-H queries and collect the estimated performance cost and power cost returned by the query optimizer. From Figure 8 (values are plotted on logarithmic scale) we can see that 16 of 22 queries have the potential for power saving in the Power-PG configuration. Normally, the benefit of power saving for these queries have a negative impact on the processing time cost as shown in the same figure. However choosing the trade-off configuration can lead to a good power saving values with less performance degradation. These queries are characterized by an important number of SQL operators and various I/O and CPU operations, which gives the query optimizer a variety of plans to choose from.

Therefore, we can achieve a good power saving queries from those plans. On the other hand, the rest of queries that doesn't show opportunities for power saving, are simple queries with a few tables and SQL operators. This leads the query optimizer to choose the same plan in every PostgreSQL configuration, duo to the small search space of the plans.

5.3.4. Power Saving.

The purpose of this set of experiments is to investigate the benefit of our approach in term of energy efficiency. We configured the DBMS to evaluate the performance and power consumption cost models for the three configurations (Time- PG, Power-PG, Tradeoff-PG). We repeat the same experiments under two different database sizes: 10GB, and 100GB using TPC-H benchmark.

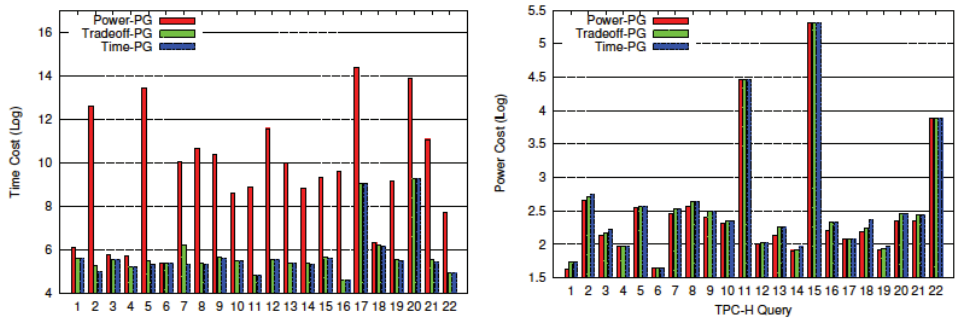

Fig. 8: Performance and power for TPC-H queries using dierent PostgreSQL congurations.

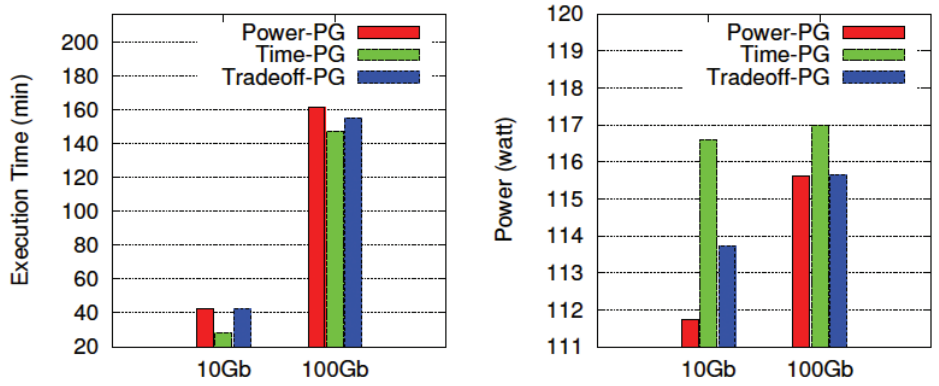

Fig. 9: Performance and power saving with dierent PostgreSQL congurations using TPC-H benchmark.

In Figure 9 we present the results of the experiments. We can clearly see that workloads consume significantly lower power when we choose a query optimizer configuration that favors low-power plans. When we compare the power-only (Power-

PG) with the performance-only (Time-PG) results, we observe a large margin in power savings, the benefit is remarkably considerable in small database size, perhaps this is due to the large amount of I/O operations and data processing required by queries of big database size which translate in more power consumption regardless of plan chosen by query optimizer. As expected, the savings of the Tradeoff- PG configuration are smaller than those obtained by the power-only experiment, but it still acceptable, especially, in 100GB datasets they are approximate. On the other hand, the power-only configuration takes more time to finish executing all the queries, which translate in a noticeable performance degradation, this not surprising, if we gain in power we automatically lose in performance. In the Tradeoff-PG configuration, the performance degradation is actually acceptable if we consider the power gain achieved.

Note that all results of our experiments only considered direct power savings in a single database server. This number could be even higher if we consider large-scale data centers with thousands of servers and cooling systems.

6. Confidence bounds

In this section we will prove the confidence in our model data and results, using high-end configuration experimentation data. To find the lower and upper bounds of population, we use Chebyshev's inequality (6). The inequality is based on population mean (denoted by μ) and population standard deviation (denoted by σ). Unfortunately, μ and σ are unknown parameters. Therefore, we must found their upper and lower bounds with some degree of confidence to calculate Chebyshev's inequality. To do the above, we (a) test whether the samples comes from a population that follows the normal distribution; (b) find the lower and upper bounds of the population mean, with the degree of confidence being 99%; (c) find the lower and upper bounds of the population standard deviation, with the degree of confidence being 99%.

6.1. Testing whether the population follows the normal distribution

The sample mean equals 116.4554, which is denoted by $\bar{x}$; while the sample standard deviation equals 2.1822, which is denoted by s. The number of samples equals 131 and is denoted by n. We perform hypothesis testing to identify whether the samples comes from a normally distributed population or not. The hypothesis testing is conducted by applying the chi-squared test for normal distribution. The null hypothesis (H_0) is defined as "The population probability distribution is normal". On the other extreme, the alternative hypothesis (H_a) is defined as "The population probability distribution is not normal".

We first divide the standard normal distribution $N(0,1)$ into a set $A_1 t$ containing eight proportionally equal parts, with each part being equal to 1/8; and then we find a set B_1 containing eight parts in a one-to-one correspondence with the ones belonging to set B_1 such that each sample is assigned onto one of the parts

belonging to B_1. We find the right-most split point of standard normal distribution, which equals 1.15, by subtracting 1/8=0.125 from 1 and then looking into the normal distribution table. By following the aforementioned procedure we result in the following set (named A_2) of split points -1.15, -0.675, -0.32, 0, 0.32, 0.675, 1.15 representing Z values. According to the aforementioned split points, we find seven new split points in one- to-one correspondence with the previous ones. Specifically, we apply Equation 6 for each point of A_2, resulting in the following set (named B_2) of split points 113.946, 114.9825, 115.7572, 116.4554, 117.1537, 117.9284, 118.9649.

By taking into consideration the above, we can reformulate the null hypothesis and the alternative hypothesis as follows:

- H_0: All the parts of B_1 are proportionally equal.
- H_a: At least one of the parts of B_1 is not proportionally equal with the rest ones.

$$\mathrm{Z} = \frac{\mathrm{x}-\bar{\mathrm{x}}}{\mathrm{s}} \Leftrightarrow \mathrm{x} = \bar{\mathrm{x}} + \mathrm{Z} * \mathrm{s} \tag{6}$$

$$\widetilde{\mathrm{X}}^2 = \sum_{\mathrm{i}=1}^{8} \frac{(\mathrm{O_i}-\mathrm{E_i})^2}{\mathrm{E_i}} \tag{7}$$

Next we find how many (estimated) points belong to each of the parts of A_1. For the first part of A_1, we find that it contains $np_1 = 131 * \frac{1}{8} = 16.375$ (estimated) points, with p_1 representing the probability of first part. Because all the parts have the same probability, we conclude that each part contains 16.375 (estimated) points. The observed points belonging to the parts of B_1 are found as follows. Each point that is less than or equal to the left-most point of B_2 belongs to the first part of B_1. The points that are greater than the left-most point of B_2 and less than or equal to the second left-most point of B_2 belong to the second part of B_1. The assignment process proceeds in a similar way for the rest parts. Table 3 contains the aforementioned information as well as information for calculating Equation 7 which asymptotically approaches chi-squared distribution X^2.

The critical region of the null hypothesis represents the region that the null hypothesis is rejected. To calculate that region we first need to choose (a) the significance level a, which is normally between 5% and 10%; and (b) the degrees of freedom $df = k - m - 1$, with k and m denoting the number of groups and the number of model parameters, respectively. In our case, we choose $a = 0.5$; the number of groups equals eight; while the number of model parameters equals two (mean and standard deviation). Therefore, $df = 8 - 2 - 1 = 5$. The critical region is the region beyond $X^2_{0.5} = 11.07$, which is denoted as critical value and calculated from the chi-squared distribution table. According to Equation 7 and Table 3, we observe that $X^2 = 8.42$, which is less than the critical value 11.07. As a result the null hypothesis is not rejected and we can safely assume that the population follows the normal distribution. Note that the bigger X^2, the stronger the evidence to reject the null hypothesis.

6.2. Finding the lower and upper bounds of the population mean with 99% confidence

The population mean is denoted by μ, while the population standard deviation is defined as σ. We find the lower and upper bounds (μ_1 and μ_u) of the population mean under $P\%$ probability or equivalently $\mathrm{P}\%$ degree of confidence. We first need first to specify the significance level $a = (100 - P)/100$. Note that the greater the degree of confidence, the greater the interval between the lower and upper bounds. For our problem, we will calculate the lower and upper bounds of population mean with 99% degree of confidence, which is a common value. As a result, the significance level is $a = \frac{100-99}{100} = 0.01$. Because the population follows the normal distribution and the deviation σ is not known, we use Equation 8 to calculate the bounds of population mean. Note that $t_{a/2}$ represents the power t distribution, with a denoting the significance level. The degrees of freedom df equal $n - 1$, we calculate $t_{0.005} = 2.58$. Therefore, according to Equation 8, we can observe that with 99% degree of confidence that $\mu_1 = 115.96$,and $\mu_2 = 116.94$.

$$\tilde{\mathrm{x}} \pm \mathrm{t}_{a/2} * \frac{8}{\sqrt{\mathrm{n}}} \tag{8}$$

6.3. Finding the lower and upper bounds of the population standard deviation with 99% confidence

Since we demand 99% confidence, the significance level is $a = 0.01$. The lower and upper bounds of population standard deviation are expressed by Equation 9 and Equation 10, respectively. By looking into the chi-squared distribution table we observe that $X^2_{0.005} = 175.3$ and $X^2_{0.995} = 92.2$. Therefore, $\sigma_1 = 1.88$ and $\sigma_u = 2.59$.

$$\sigma_l = s\sqrt{\frac{n-1}{X^2_{a/2}}} \tag{9}$$

$$\sigma_u = s\sqrt{\frac{n-1}{X^2_{1-a/2}}} \tag{10}$$

6.4. Finding probabilistically the lower and upper bounds of the population

From Chebyshev's inequality (Equation 11) we can find probabilistically bounds for the population. Specifically, to find the lower bound of the population, we assume that $X - \mu \leq 0. \mu = \mu_l$ and $\sigma = \sigma_u$.According to the above, inequality 11 becomes Inequality 12. For $k = 3$ we have that $Pr\,(X \leq 108.19) \leq 0.11$. To find the upper bound of the population, we assume that $X - \mu \geq 0, \mu = \mu_u$ and $\sigma = \sigma_u$. According to the above, inequality 11 becomes inequality 13. For $k = 3$ we have that $Pr\,(X \geq 124.71) \leq 0.11$. To find the confidence level for the lower bound of population, we have to multiply the probability of A to be greater than 108.19 (i.e., 1 — 0.11 = 0.89) by (a) the confidence level for the lower bound of population mean, and (b) by the confidence level for the upper bound of population standard deviation. As a result, the lower bound of population is equal to 108.9 with 87% (0.99 * 0.99 * 0.89) degree of confidence. By working in a similar way, the upper bound of the population is

equal to 124.71 with 87% degree of confidence. Note that we can increase the degree of confidence (by increasing *k*) at the cost of decreasing/increasing the lower/upper bound of population.

$$P_r(|X - \mu| \geq k\sigma) \leq 1 / k^2 \quad (11)$$

$$P_r(X \leq \mu_l - k\sigma_u) \leq 1 / k^2 \quad (12)$$

$$P_r(X \geq \mu_u + k\sigma_u) \leq 1 / k^2 \quad (13)$$

7. Conclusion

In this paper, we first summary the initiatives that the database community did for building energy applications and DBMS. These initiatives cover software and hardware aspects. Due to the complexity of the DBMS, we propose a green-query optimizer build on the top of PostgreSQL. Before building it, an audit has been performed to identify energy-sensitive components of the query optimizers. Based on this audit, a methodology of building such a query optimizer is given and proposes to modify the query processor. Our methodology is supported by an open source tool available at the forge of our laboratory to allow researchers, industrials and students to get benefit from it. Intensive experiments were conducted to demonstrate the efficiency and usage of our proposal. The obtained results are encouraging. Based on these results a probabilistic proof is given to evaluate the confidence bounds of our model and results. We can conclude that our proposal is a complete since it cover a large state of art discussion, a comprehensive methodology supported by a open source tool and solid mathematical proof.

Table 3: Information of the calculations.

Interval	Observed points (O)	Estimated points (E)	(O-E)	$(O-E)^2/E$
< 113.946	15	16.37	-1.37	0.114655
(113.946, 114.9825]	18	16.37	1.63	0.162303
(114.9825, 115.7572]	18	16.37	1.63	0.162303
(115.7572, 116.4554]	19	16.37	2.63	0.422535
(116.4554, 117.1537]	18	16.37	1.63	0.162303
(117.1537, 117.9284]	6	16.37	-10.37	6.569145
(117.9284, 118.9649]	20	16.37	3.63	0.804942
> 118.9649	17	16.37	0.63	0.024246

Currently, we are integrating physical design aspects in our tool and pushing its development to become like an energy advisor.

References

[1] Abadi, D., Agrawal, R., Ailamaki, A., Balazinska, M., Bernstein, P. A., Carey, M. J., ... & Gehrke, J. (2016). The Beckman report on database research. *Communications of the ACM, 59*(2), 92-99.

[2] Chaudhuri, S., Narasayya, V., & Ramamurthy, R. (2004, June). Estimating progress of execution for SQL queries. In *Proceedings of the 2004 ACM SIGMOD international conference on Management of data* (pp. 803-814). ACM.

[3] Dannecker, L., Schulze, R., Böhm, M., Lehner, W., & Hackenbroich, G. (2011, July). Context-aware parameter estimation for forecast models in the energy domain. In *International Conference on Scientific and Statistical Database Management* (pp. 491-508). Springer, Berlin, Heidelberg.

[4] Do, J., Kee, Y. S., Patel, J. M., Park, C., Park, K., & DeWitt, D. J. (2013, June). Query processing on smart SSDs: opportunities and challenges. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data* (pp. 1221-1230). ACM.

[5] e Sustainability Initiative. (2012). G., the Boston Consulting Group, I: Gesi smarter 2020: The role of ict in driving a sustainable future. *Press Release, December.*

[6] Graefe, G. (2008, March). Database servers tailored to improve energy efficiency. In *Proceedings of the 2008 EDBT workshop on Software engineering for tailor-made data management* (pp. 24-28). ACM.

[7] Härder, T., Hudlet, V., Ou, Y., & Schall, D. (2011, April). Energy efficiency is not enough, energy proportionality is needed!. In *International Conference on Database Systems for Advanced Applications* (pp. 226-239). Springer, Berlin, Heidelberg.

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

[9] Intel and Oracle (2011). Oracle exadata on intel R xeonR processors: Extreme performance for enterprise computing. White paper.

[10] Kunjir, M., Birwa, P. K., & Haritsa, J. R. (2012, March). Peak power plays in database engines. In *Proceedings of the 15th International Conference on Extending Database Technology* (pp. 444-455). ACM.

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

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

[13] McCullough, J. C., Agarwal, Y., Chandrashekar, J., Kuppuswamy, S., Snoeren, A. C., & Gupta, R. K. (2011, June). Evaluating the effectiveness of model-based power characterization. In *USENIX Annual Technical Conf* (Vol. 20).

[14] Otoo, E., Rotem, D., & Tsao, S. C. (2009, June). Energy smart management of scientific data. In *International Conference on Scientific and Statistical Database Management* (pp. 92-109). Springer, Berlin, Heidelberg.

[15] Poess, M., & Nambiar, R. O. (2008). Energy cost, the key challenge of today's data centers: a power consumption analysis of TPC-C results. *Proceedings of the VLDB Endowment, 1*(2), 1229-1240.

[16] Rodriguez-Martinez, M., Valdivia, H., Seguel, J., & Greer, M. (2011). Estimating power/energy consumption in database servers. *Procedia Computer Science, 6,* 112-117.

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

[18] Roukh, A., & Bellatreche, L. (2015, September). Eco-processing of OLAP complex queries. In *International Conference on Big Data Analytics and Knowledge Discovery* (pp. 229-242). Springer, Cham.

[19] Roukh, A., Bellatreche, L., Boukorca, A., & Bouarar, S. (2015, October). Eco-dmw: Eco-design methodology for data warehouses. In *Proceedings of the ACM Eighteenth International Workshop on Data Warehousing and OLAP* (pp. 1-10). ACM.

[20] Royer, K., Bellatreche, L., & Jean, S. (2014, October). One semantic data warehouse fits both electrical vehicle data and their business processes. In *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)* (pp. 635-640). IEEE.

[21] Šikšnys, L., Thomsen, C., & Pedersen, T. B. (2015). MIRABEL DW: Managing Complex Energy Data in a Smart Grid. In *Transactions on Large-Scale Data-and Knowledge-Centered Systems XXI* (pp. 48-72). Springer, Berlin, Heidelberg.

Submitted 20.03.2019
Accepted 31.05.2019