ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)

Vol 11, Issue 2, February 2024

GreenPipeline: Data-Driven in the Light of Energy Constraints

[1] Simon Pierre DEMBELE

[1] Institute of Computer Science, University of Tartu
Corresponding Author Email: [1] simon.pierre.dembele@ut.ee

Abstract*— The high energy consumption has now become a critical and urgent issue for database community. Query optimizer are identified as one of the most energy hungriest component in Data Storage Systems (DSSs). This is mainly due to the users' needs to ingest, store and process data very frequently. With the torch of environmental sustainability being waved and the exorbitant cost of energy, the development and application of energy reduction techniques within these systems is more urgent than ever. In this paper, we propose a tool baptized "GreenPipeline" that enhances the query execution engine of the PostgreSQL system by integrating energy constraints during Analytical Queries processing. This initiative aligns with the development of environmentally friendly databases and has two primary objectives: to evaluate the energy efficiency benefits achieved by incorporating an energy model into query processing and to provide users a means to estimate the system's energy consumption without requiring physical equipment. More specifically, our primary goal is to integrate our energy cost model and an evaluation plan methodology into the core of the PostgreSQL optimizer. We offer an evaluation plan approach that enables the optimization of either energy efficiency, performance, or a compromise between the two. Preliminary results highlight the effectiveness of our tool in reducing energy consumption for various query sets from different benchmarks.

Keywords: Data storage, optimization, energy model, energy-aware query processing, benchmarking.

I. INTRODUCTION

The pursuit of eco-efficient solutions to mitigate the economic cost and environmental degradation resulting from the substantial energy consumption of computer systems has captured the attention of researchers, both in academic and professional environments. Some strategies explored to address these issues are categorized into two approaches: hardware-based and software-based approaches. In addition to Dynamic Energy Management (DEM) and the utilization of virtualization techniques in the software approach, it has been observed that it is feasible to regulate energy efficiency at a more specific level within the internal architecture of Database Management Systems (DBMS) [6][7].

Indeed, Data storage systems architectures encompass a set of interconnected components, each playing a specific role in routine operations such as data retrieval, modification, construction, organization, transformation, backup, or data restoration. Among these components, the query processing engine stands out as one of the most complex. On one hand, it is tasked with efficient access and manipulation of massive data, and on the other hand, it interacts with other phases related to the selection and maintenance of optimization structures, such as indexing methods [8].

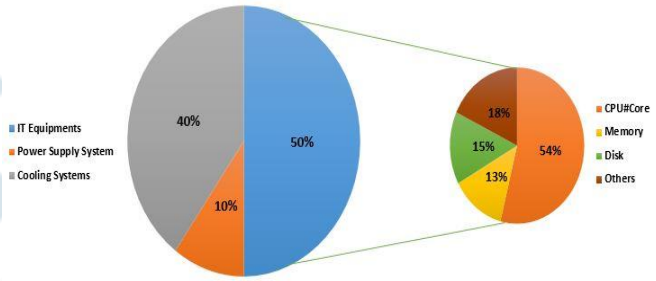

Fig. 1: Energy distribution among different components.

Data storage systems have been recognized as one of the primary contributors to energy consumption in Data center. This energy consumption is attributed to various elements, including servers, storage devices, networks, and supporting infrastructure components [9][11][1][2]. **Fig. 1** provides a representation of the power distribution within a Data center.

The integration of energy constraints begins with the identification of points within the DSS where integration and exploitation are feasible. One of the key integration points is the query processing system. In comparison to traditional database systems, where the choice of query execution plan is primarily based on a performance-focused cost model, *Green databases* must process data in an *energy-efficient manner*. One approach is to select the final plan based on an energy model with the aim of minimizing the overall data storage system's energy consumption while preserving performance and user-friendliness.

This approach necessitates the development of precise energy cost models that consider various parameters related to the database components, such as the database schema (table sizes, tuple lengths, etc.), query workload (query type,

ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)

Vol 11, Issue 2, February 2024

join selectivity factors, selection predicates, intermediate result sizes, etc.), and hardware characteristics (number of CPU cores, buffer size, disk page size, etc.).

In this paper, we introduce a benchmark for energy-aware query optimization, referred to as "GreenPipeline". This benchmark harnesses the PostgreSQL query optimizer and incorporates the energy dimension into the optimization process. Our approach is inspired by the research presented in [10], where the authors present a model for sequential query processing. We enhance their work by capitalizing on the intra-parallelism capabilities introduced in PostgreSQL since version 9.6 on multi-core architectures, enabling intra-query parallelism processing. To achieve this, we have developed and integrated mathematical cost models that estimate energy consumption, considering the parallel processing mode within the query optimizer. Following an extensive series of experiments to evaluate the energy savings achieved through our approach. We also present a user-friendly graphical interface, the interface serves as a diagnostic tool for end-users, developers, and database administrators (DBAs), empowering them to enhance their awareness of energy consumption.

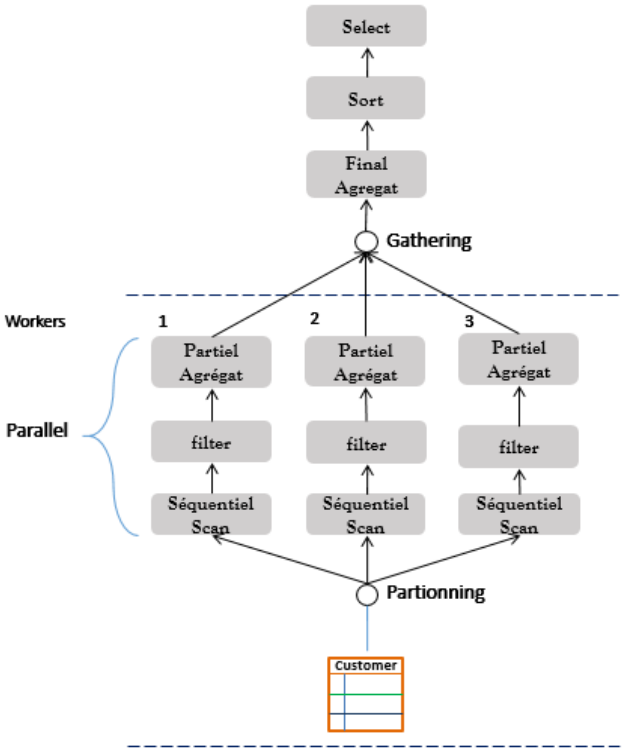

Fig. 2: Parallel Plan Illustration for PostgreSQL.

II. MOTIVATION

The main motivations of this work are, on one hand, financial and well-being benefits: energy efficiency contributes to the reduction of *CO_2 emissions*, which has a positive impact on *socio-economic and health aspects*, especially in terms of the sustainability and the health of individuals. On the other hand, the goal is to examine the feasibility of this approach and assess potential energy savings.

Traditional optimizers rely on a cost model to evaluate the performance of each plan and select the most optimal one. The primary objective of this approach is not energy optimization but rather runtime. PostgreSQL's optimizer executes a query by segmenting it into a set of pipelines, delimited by blocking operators. The introduction of the intra-parallel execution mode enables the optimizer to generate intra-parallel execution plans based on the chosen number of workers. To force the optimizer to generate a plan using three workers, it is necessary to initialize the "***max_parallel_workers_per _gather***" variable to 3 in the query planner. **Fig. 2** provides an illustration of the execution plan for the following query:

SELECT sum(sales) **FROM** *customers* **WHERE** *customers.city*='London'**;**

For each query, the optimizer generates a certain physical execution plan. Each plan consists of a set of operators that define a unique way of extracting data stored in secondary memory. Each operator is characterized by algorithmic implementation, input and output parameters, its degree of parallelism, and so on. Therefore, different plans may utilize system resources differently. To verify this observation in terms of system energy consumption, we estimate the energy cost of a query by varying the degree of parallelism (the number of workers) from 0 to 4. As illustrated in **Fig. 3**, the degree of parallelism (**DoP**) impacts both the time and energy of the chosen plan.

In light of these findings, the proposal for a green framework that integrates an energy model into the query optimizer could lead to energy optimizations by selecting a plan that minimizes energy consumption.

III. QUERY MODELS DEFINTION

The approach towards an energy-efficient optimizer during query processing relies on the definition and incorporation of two distinct models. The first model should have the ability to accurately and flexibly estimate the energy cost of a query by incorporating all relevant parameters related to query operations. The second model is dedicated to the selection of a plan that meets the specific needs of the user.

A. Energy Cost Model

We describe briefly our model that estimate the energy cost. The main steps of building green query processors are:

1) Identification of relevant energy-sensitive parameters belonging to hardware and software components,
2) Elaboration of mathematical cost models estimating consumed energy when executing a query on a target DBMS,
3) Setting of values of the energy-sensitive parameters using machine learning techniques.

ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)

Vol 11, Issue 2, February 2024

We employ non-linear regression techniques for estimating our energy cost units and utilize linear regression to determine the energy factor added when utilizing multiple cores. Comprehensive information on energy modeling steps can be found in our papers [5][4][3].

During the Query (Q) plan execution, which consists of a set of pipelines, two primary actions come into play:

- The CPU carries out a series of instructions for each operator, and the workload is influenced by the number of tuples. This parameter is referred to as the CPU cost and is denoted by C_{CPU}. In a parallel plan, the CPU cost is distributed among the cores according to the degree of parallelism (DoP).
- A specific number of data blocks are transferred from secondary memory to main memory, which is referred to as Disk access - number Input/Output (I/O), denoted by $C_{I/O}$.

Therefore, for the query (Q) represented by the plan ($Plan$) consisting of k pipelines denoted as PL_1, PL_2, . . ., PL_k, the energy cost is computed as follows:

$$Power\ (Q, DoP) = \frac{\sum_{i=1}^{k} Power(PL_i, DoP_i) * Time(PL_i, DoP_i)}{Time(Q)}$$

where parameters $Time(Q)$, $Time\ (PL_i, DoP_i)$ represent respectively, the execution time of Q and the time of PLi with degree DoP_i.

To estimate the energy consumption in parallel mode, we introduce an energetic factor denoted fc. This factor will be defined according to the degree of parallelism and expresses the energic difference between the power consumed during the parallel mode processing compared to the sequential processing at the CPU level.

The following formula defines this factor: $f_{cp} = (P_{mp} - P_0)/P_0$. We denote by P_{mp} the average power dissipated on n cores to treat a query and P_0 the power consumed on the sequential plan (degree of parallelism set to 0). The following equation: $f_{cDoP} = \alpha * DoP + \beta + \epsilon$ define the energetic factor between one and several cores obtained by applying the relationship defined by the simple linear regression technique. DoP is the degree of parallelism which correspond to the number of cores involve in the query processing et ϵ represents the estimation error. The parameters α and β are regression coefficients that will be estimated.

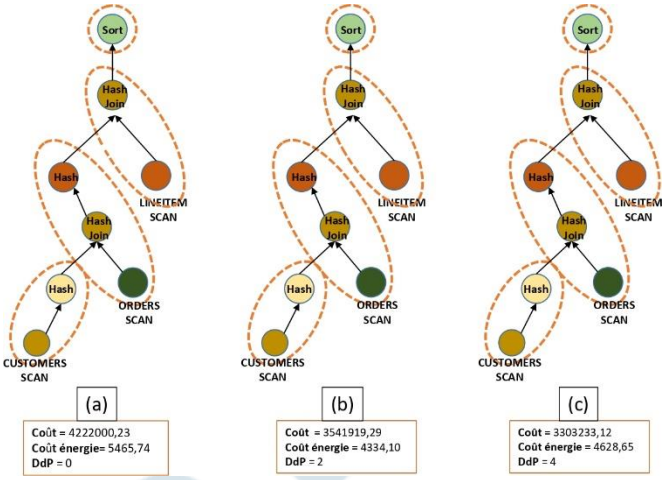

Fig 3: Query Q3 plan cost associated to Degree of Parallelism 0, 2, and 4 with $W_{time} = 1$ $W_{Power} = 0$

The power dissipated when processing a sequential and parallel pipeline is the combination of the energy consumption from the parameters identified. The formula is:

$$Power\ (PPL_i, DoP) = (f_{C_{DoP}} + 1) * W_{CPU} * \sum_{k=1}^{n} C_{CPU_k} \oplus W_{I/O} * \sum_{k=1}^{n} C_{I/O} \quad (1)$$

where W_{CPU} and $W_{I/O}$ are the model parameters (i.e., unit power costs) for the operators. The $C_{I/Ok}$ is the predicted number of I/O required for executing a specific operator. The C_{CPUk} is the predicted number of CPU Cycle and buffer cache that DBMS needs to run a specific operator. The n is the number of operators in the pipeline.

Before integrating our model into the optimizer, we conducted a quality assessment. This evaluation involved determining the prediction error, which was calculated by comparing the measured energy consumption (using a wattmeter) with the estimated energy consumption in percentage. To do this, we utilized datasets and queries from the TPC-H, SSB, and TPC-DS benchmarks, details can be found in [5].

B. Plan Evaluation Mode

Rather than choosing a plan with optimal performance, the energy cost model can be used to choose a plan that saves energy or to define a certain threshold of the trade-off between energy and performance. The adjustable trade-off between performance and energy is made by using a criterion that reflects the choice of users/ administrators. The criterion model adopted has the following formulation:

$$C_{plan} = \alpha \times log\ (T) + (1 - \alpha) \times log(P) \quad (2)$$

with T denote the performance, P denote the energy cost, $Cplan$ denote the plan cost and α is constant value in interval [0, 1]. This coefficient reflects the superiority of performance (T) or energy cost (P). In the relationship described by equation 2, extreme values of the criterion α lead to the deterioration of one of the factors. We can select plans with

ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)

Vol 11, Issue 2, February 2024

desired characteristics for different optimization goals with the choice of α:

```
Cost count_power_cost_NLR(double io_cost, double cpu_cost,int wh)
{
 // io_cost : Cout du DISK
 // cpu_cost: Cout du CPU
 // wh: Degre de parallelisme

 Cost power_cost = 0.0;
 double fcd=wh*0.0552+0.00783;

 if(wh!=0){
  power_cost = B0
  + pow(io_cost, 1) * pow(cpu_cost, 0) * B1 + pow(io_cost, 2) * pow(cpu_cost, 0) * B3
  + (pow(io_cost, 0) * pow(cpu_cost, 1) * B2 + pow(io_cost, 0) * pow(cpu_cost, 1) * fcd * B2)
  + (pow(io_cost, 0) * pow(cpu_cost, 2) * B4 + pow(io_cost, 0) * pow(cpu_cost, 2) * fcd * B4)
  + (pow(io_cost, 1) * pow(cpu_cost, 1) * B5 + pow(io_cost, 1) * pow(cpu_cost, 1) * fcd * B5);
 }
 else {
  power_cost = B0
  + pow(io_cost, 1) * pow(cpu_cost, 0) * B1 + pow(io_cost, 0)* pow(cpu_cost, 1) * B2
  + pow(io_cost, 2) * pow(cpu_cost, 0) * B3 + pow(io_cost, 0)* pow(cpu_cost, 2) * B4
  + pow(io_cost, 1) * pow(cpu_cost, 1) * B5;

 }
 return power_cost;
}
```

Fig 4: The content of the function count_power_cost_NLR()

- **α is set to 1, so $time\_wt$ = 1 and $power\_wt$ = 0:** the configuration underscores that optimizing performance is the primary objective during query processing.
- **α is set to 0, so $time\_wt$ = 0 and $power\_wt$ = 1:** the configuration signifies that the primary optimization goal during query processing is energy efficiency.
- **α is set to 0.5, so $time\_wt$ = 0.5 and $power\_wt$ = 0.5:** this configuration aims to strike a balance between the two factors. An execution plan is chosen that fulfills the energy constraint without compromising performance.

C. Models integration

To incorporate the energy cost model and the evaluation model, adjustments need to be made in the source code of the PostgreSQL system. These adjustments involved several files, with particular attention paid to the following two files:

- **Path:"src/backend/optimizer/path/costsize.c":** This file contains all the functions to estimate the costs of each algebraic operator. We retrieve the I/O and CPU costs for each operator, and then estimate its power using the function that implements our cost model within the file. **In Fig. 4**, you will find the code for the $count\_power\_cost\_NLR()$ function, where Bi represents the regression coefficients, and DdP denotes the degree of parallelism.
- **Path:"src/backend/optimizer/util/pathnode.c":** We modify the following functions $compare\_fractional\_path\_costs()$ and $compare\_path\_costs()$ so that the plan selection is being based on our evaluation model which combines the two constraints.

IV. SYSTEM ARCHITECTURE

The GreenPipeline architecture consists of two main components: a back-end and a Graphical User Interface (GUI). The back-end comprises a Database Management System (DBMS) that integrates our energy cost model and a criterion model reflecting user and administrator preferences. The GUI assists users in configuring settings and displaying query results. In our research, we utilized PostgreSQL, which supports parallel query execution. This tool was developed in a Java environment. The workflow of our Framework is described in the **Fig. 5**.

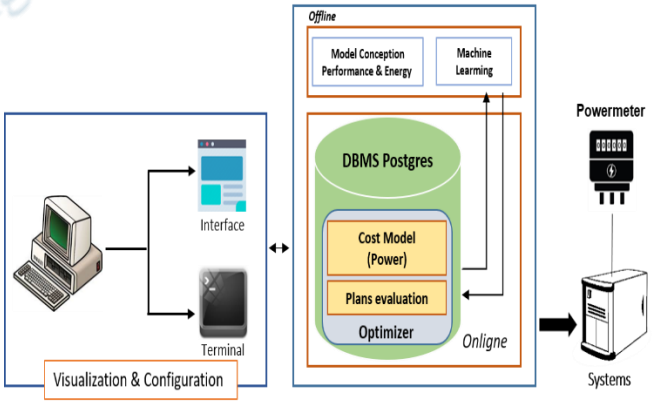

Fig 5: Workflow of GreenPipeline Framework

A. Experimental design

The adjustable parameters and the free license character make PostgreSQL the perfect tool for our experiments. As mentioned before on PostgreSQL, its open source character allows us to modify this kernel to validate the efficiency of our model. We use the PostgreSQL release 10.3 for our experiments. A graphical interface (GUI) part allowing to

ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)

Vol 11, Issue 2, February 2024

define the values of certain parameters, a part to visualize in real time the energy consumption of the demands and a part to monitor and measure the values of the performance meters. The user can compare and decide what algorithm enable or disable. The tool (demo available on https://www.youtube.com/watch?v=aAaoBnQsz-0) is available on https://github.com/dembeles/GreenPipelineDB.git. **Fig. 6** provides an overview of settings and outputs interface. Using the underlying system does not necessitate the GUI; it is solely designed to showcase various potential interactions with the system. The module enables you to accomplish the following tasks:

- Settings of system parameter values
- Plan visualization
- Visualization of consumption timeline, energy prediction values and Query results.

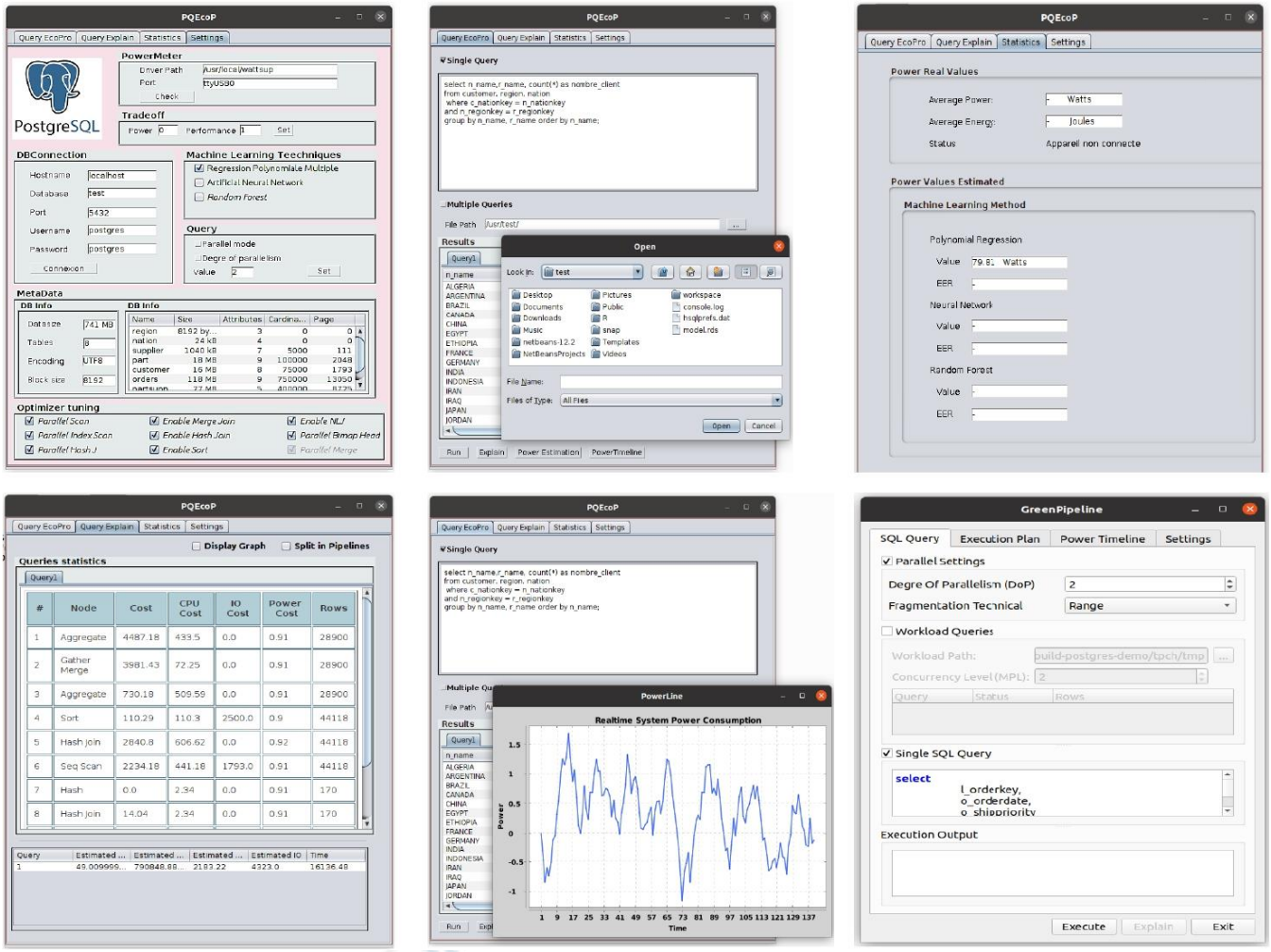

Fig 6: Framework Interfaces (Parameters settings, Plan visualization).

B. Main results

We used the TPC-H and SSB benchmarks to assess our proposal across different scaling factors. For each scaling factor, we evaluated the average energy consumption and the time required to process a query sets on each configuration. We compared the power variation and performance degradation of energy-oriented configurations (α = 0, α = 0.5) with that of performance-oriented configuration (α = 1). We noted that the α = 0 configuration exhibits superior power efficiency compared to the other configurations, approximately 1.5 times more efficient than the α = 1 configuration. However, the energy-focused α = 0 configuration takes more time to process queries, resulting in a performance degradation at that level. Although a deterioration in performance was observed, an energy saving of 26.4% is materialized. Through a series of experiments on the TPC-H and SSB benchmarks, our framework achieved an average energy saving of 12.5% in the configuration α = 0 and 13.9% in the configuration α = 0.5.

V. CONCLUSION

The Green Pipeline Framework is an initiative aimed at promoting environmentally-friendly database development. It has two primary goals: to evaluate the energy efficiency advantages of integrating an energy model into query

processing and to offer users a means of estimating system energy usage without the necessity of physical equipment.

REFERENCES

[1] Consumer Research Associates. 2014 (accessed 2019.01.22). Energy Efficiency Policy Options for Australian and New Zealand Data Centres: E3 Equipment Energy Efficiency. Report. ENERGY RATING. http://energyrating.gov.au/document/report-energy-efficiency-policy-options-australian-and-new-zealand-data-centres.

[2] Simon Pierre Dembele, Ladjel Bellatreche, Angelo Lorusso, Francesco Marongiu, and Domenico Santaniello. 2023. In-Memory Database Query Energy Estimation: Modeling & Green Strategy Support. In 2023 IEEE World Conference on Applied Intelligence and Computing (AIC). 278–285.

[3] Simon Pierre Dembele, Ladjel Bellatreche, and Carlos Ordonez. 2020. Towards Green Query Processing - Auditing Power Before Deploying. IEEE.

[4] Simon Pierre Dembele, Ladjel Bellatreche, Carlos Ordonez, Nabil Gmati, Mathieu Roche, Tri Nguyen-Huu, and Laurent Debreu. 2020. Thinking Smart [Big Steps Towards Query Eco-Processing - Thinking Smart]. ARIMA J. 34 (2020), 7.

[5] Simon Pierre Dembele, Ladjel Bellatreche, Carlos Ordonez, and Amine Roukh. 2019. Think big, start small: a good initiative to design green query optimizers. Cluster Computing (30 Oct 2019).

[6] Goetz Graefe. 2008. Database Servers Tailored to Improve Energy Efficiency. In Proceedings of the 2008 EDBT Workshop on Software Engineering for Tailor-Made Data Management (Nantes, France) (SETMDM '08). Association for Computing Machinery, New York, NY, USA, 24–28.

[7] Stavros Harizopoulos, Mehul A. Shah, Justin Meza, and Parthasarathy Ranganathan. 2009. Energy Efficiency: The New Holy Grail of Data Management Systems Research. ArXiv abs/0909.1784 (2009).

[8] Majid Khan and MNA Khan. 2013. Exploring query optimization techniques in relational databases. International Journal of Database Theory and Application 6, 3 (2013), 11–20.

[9] Emerson Liebert. 2007. Five Strategies for Cutting Data Center Energy Costs Through Enhanced Cooling Efficiency. White Paper. http://www.emersonnetworkpower.com/documentation/en-us/brands/liebert/documents/white%20papers/data-center-energy-efficiency_151-47.pdf

[10] Amine Roukh, Ladjel Bellatreche, and Carlos Ordonez. 2016. EnerQuery: Energy-Aware Query Processing. In Proceedings of the 25th ACM International on Conference on Information and Knowledge Management (Indianapolis, Indiana, USA) (CIKM '16). ACM, New York, NY, USA, 2465–2468.

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