PAID: Power-efficient AI-optimized Databases

Ayoub Bouhatous[1], Ladjel Bellatreche[2], El Hassan Abdelwahed[1], and Carlos Ordonez[3]

[1] Cadi Ayyad University, Marrakech, Morocco
[2] LIAS/ISAE-ENSMA, Poitiers, France
[3] University of Houston, Houston, USA

Abstract. Traditionally, query processors (QPs) have been designed to optimize response time, but not energy consumption. Heeding this new optimization goal, during the last decade, the database community turned its attention to enhancing the energy efficiency (EE) of QPs, but there are still major challenges. By analyzing recent work on EE of QPs, we observe that even though Machine Learning (ML) models can accurately predict energy consumption, they do not modify the core QP functionality (software) nor dynamically adjust CPU configuration parameters (hardware: # of cores and clock frequency), to actually save energy for a query. To address this gap, we introduce PAID, a subsystem integrated with the query optimizer that combines old AI with new AI: a Genetic Algorithm (GA) with a Neural Network (NN). The NN model predicts query energy consumption, whereas GA determines the optimal CPU configuration, deciding clock speed frequency and core allocation for each query. The GA configuration is then fed back into the NN model to tune prediction accuracy. Our experiments, conducted on the TPC-H benchmark with PostgreSQL, show that PAID effectively finds CPU configurations that exceed the performance of default settings, achieving significant energy savings.

1 Introduction

The development of efficient query processors (QP) remains an active research area. As data science and AI applications proliferate, QPs have become the backbone for data preparation and data exploration phases, enabling AI model computations. In parallel, the ICT industry is estimated to be responsible for 1.8% - 2.8% of the global carbon footprint. Therefore, growing environmental concerns will require policymakers, industry stakeholders and researchers to prioritize energy efficiency (EE) in the design of future computing systems. Thus, improving the EE of QPs will remain an important problem for green computing.

Aligned with this motivation, the database community has shown notable interest over the past two decades in developing various initiatives to address the challenge of EE in databases. These efforts include surveys [15,2,7], methodic studies for facilitating research on this topic [1], and prediction models for assessing the energy consumption of traditional DBMS QPs [6,4,13,2,3,16,9,8].

By conducting an in-depth analysis of these research initiatives, we observe that they introduce both hardware and software tactics to enhance EE [1]. Hard-

ware manufacturers have made significant strides in developing high-performance CPUs [14], GPUs [10], and specialized hardware accelerators [12]. Dynamic Voltage and Frequency Scaling (DVFS) is recognized as one of the most effective techniques for reducing power consumption in both CPUs and GPUs. By dynamically adjusting the voltage and frequency based on workload demands, DVFS helps optimize `EE` without significantly compromising performance [5,11]. Software tactics include, among others, analytical cost models designed to predict the query energy consumption. They extend conventional query optimizers by incorporating parameters such as IO, CPU usage, and memory costs. Machine learning (ML) techniques use these parameters to predict energy consumption of queries [2]. More recently, these models have been further refined by incorporating hardware-related parameters, such as the number of CPU cores and frequency [3]. However, they primarily rely on basic predictive methods such as linear regression, multiple regression, and random forest [13,4,6].

The analysis of the current state of the art reveals two important findings (F1 and F2) that need to be consolidated, as well as two main limitations (L1 and L2). **F1** The pivotal role of ML techniques in aiding the development of environmentally-friendly `QP`. **F2** The importance of ML solutions that integrate both software and hardware parameters to accurately predict energy consumption. **L1** The existing energy consumption prediction models rely on simple ML techniques. **L2** Despite the CPU's dominance in energy consumption, existing ML-based solutions often overlook the importance of optimizing CPU configurations—specifically, the number of cores and CPU frequency. Instead, they rely on the default settings predefined by the host machine of the target DBMS, potentially missing opportunities for improved`EE`.

To overcome the above two limitations, we propose a novel optimization system PAID that integrates Genetic Algorithm (`GA`) and Neural Networks (`NN`). `GA` aims at selecting the optimal number of CPU cores and a CPU frequency for a given query. The configuration chosen by the `GA` is subsequently incorporated into the `NN` model to enhance the accuracy of the prediction. Thereafter, `NN` are utilized to predict both energy consumption and response time of queries.

2 Integrating old and new AI: `GA` and `NN`

In this section, we describe the two components of PAID system: `GA` and `NN`.

2.1 NN model

Our `NN` model aims to estimate the energy consumption (output) considering critical CPU settings, including the number of cores and CPU frequency. According to the state-of-the-art, estimating the energy consumption for a given query Q_i requires considering the following key features (that can be extracted from query optimizers): CPU cost, I/O cost, memory cost, and database size. Therefore, from an energy consumption perspective, Q_i can be represented by the following feature vector $\overrightarrow{Q_i}$, which serves as input to our `NN` model:
$\overrightarrow{Q_i} = (COST_{CPU_i}, COST_{IO_i}, COST_{Memory_i}, DB_{Size_i})$, where $COST_{CPU_i}$, $COST_{IO_i}$, $COST_{Memory_i}$, and DB_{Size_i} represent respectively the number of instructions executed by the CPU, the number of pages read/written from secondary storage

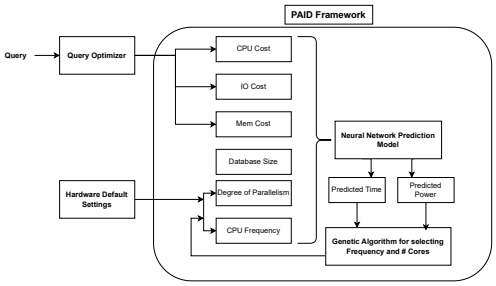

Fig. 1: The PAID Subsystem.

(persistent storage), the number of pages accessed in main memory when executing Q_i, and the size of the target database. A CPU configuration used for executing a query Q_i is represented by the following vector: $\overrightarrow{CPU_i} = (FRQ_i, CORE_i)$, where FRQ_i, and $CORE_i$ represents respectively the CPU frequency and the number of cores used during Q_i execution. *It is important to highlight that, in existing studies, all CPU vectors associated with queries are fixed. That is, the CPU configuration remains fixed across all queries, rather than being dynamically optimized based on individual query characteristics.* Based on these two vectors, a query Q_i is then represented by a vector obtained by concatenating the query feature vector $\overrightarrow{Q_i}$ and the CPU configuration vector $\overrightarrow{CPU_i}$. The query vector is initially passed through a set of fully connected layers of monotonically decreasing size.

2.2 Selecting an Optimal CPU Configuration: A Genetic Algorithm

Recall that our `GA` aims at selecting the best configuration of CPU for a given query Q_i. Let us first formalize the problem of CPU configuration selection. Given the extended query (Q_i) vector $\overrightarrow{EQ_i} = (\overrightarrow{Q_i}, \overrightarrow{CPU_i})$ and our `NN` model that predicts the energy consumption of queries under a given CPU configuration, our problem consists in setting the best CPU configuration that minimizes both the power consumption (F_{Power}) and execution time F_{Time}:
$minimize_{(j,k)} \;\; F_{power}(\overrightarrow{Q_i}, \overrightarrow{CPU_{i,j,k}}) \times F_{time}(\overrightarrow{Q_i}, \overrightarrow{CPU_{i,j,k}})$
where $min\_frequency \leq j \leq max\_frequency$ and $min\_number\_cores \leq k \leq max\_number\_cores$.

Our `NN` model predicts energy consumption and response time for a query based on considered the features, without varying them. In contrast, the `GA` selects the optimal CPU configuration for a query. By sending predicted energy from the `NN` to the`GA`, PAID enables `NN` to get the optimal CPU configuration. Figure 1 illustrates the connection between `NN` and `GA`, where the fitness function used by the `GA` is provided by the `NN` model.

3 Experimental Study

We present and discuss experimental results obtained using a Dell Precision Tower 3620 server equipped with a Core i7-6700 CPU (4 cores, 8 threads), 16 GB DDR4 RAM, and a 256 GB SSD. Our experimental setup comprises three main components: a client machine (monitor), a PostgreSQL DBMS (version 14.1) running on Ubuntu 20.04 (kernel 20.04.4 LTS), and an external power meter for energy measurement called Yocto-Watt[4] at a frequency of 1 Hz. It is directly placed between the database server and the electrical power supply and it is linked using a USB cable to the client machine for data collection. Our server is installed with We utilize the TPC-H benchmark to train and evaluate our models. Three databases are generated with sizes of 10 GB, 30 GB, and 50 GB (used to validate our proposal). In addition to the benchmark's original 22 queries, we generate 70 additional queries randomly. For each database instance, we collected query execution plans and measured energy consumption using our power meter. Each query was executed multiple times while varying the number of CPU cores (parallelism degree) from 1 to 4 and adjusting CPU frequency configurations between 0.8 GHz and 3.4 GHz. Before conducting our experiments, we deactivated unnecessary background tasks and cleared both the operating system and PostgreSQL buffers before each query execution. To assess the effectiveness of our `NN` model, we selected Random Forest Regression (`RFR`) as a baseline. The input layer of our `NN` model consists of 6 neurons, corresponding to the number of features in the input data. The subsequent three layers are densely connected, with each neuron in a layer being connected to every neuron in the previous layer. We use the rectified linear unit (ReLU) as the activation function for the hidden layers, which introduces non-linearity to the network. The output layer consists of two neurons, each producing a single output value, indicating the two outputs predicted by the network. The activation function for these output neurons is linear, which allows for continuous predictions of power consumption and query execution time. The model is compiled using the Adam optimizer with a learning rate of 0.0125, and the loss function employed during training is mean absolute error (MAE).

`NN` model vs. `RFR` Table 1 presents the evaluation results of Random Forest Regression (RFR) and our `NN` model, showcasing their performance metrics for predicting power consumption and query execution time. The `NN` model generally outperforms the `RFR` model in terms of both MAE and R^2 for predicting power and time. This indicates that the `NN` model provides superior accuracy in estimating the energy consumption and execution time of queries, making it a more reliable choice for predicting these metrics.

Impact of CPU configuration To evaluate the impact of CPU configuration, we varied the number of cores and measured response time and energy consumption for 22 queries (Fig. **??**). As expected, increasing the number of cores generally improved response time, except for simple queries (with less joins and sorting). Surprisingly, energy consumption was found to correlate with elapsed

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

Table 1: Evaluation results of RFR and NN in predicting power consumption and time of queries.

Model's Output	Evaluation Metrics	RFR	NN
Power	MAE	2.3	2.5
	R^2	63.82	64.17
Time	MAE	54.72	13.85
	R^2	47.51	83.64

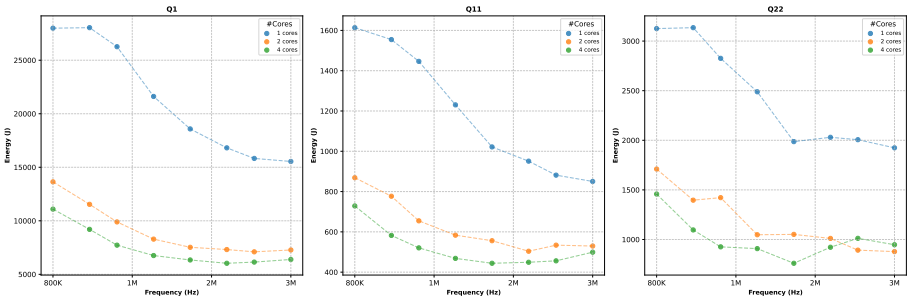

Fig. 2: Impact of variation of CPU configuration on queries Q1, Q11, and Q22

time. Using all available cores reduced energy consumption for most queries, except some queries with high number of joins and sorting using two attributes, energy usage remained high despite full core utilization. This suggests that for complex queries, the overhead of parallelism may offset energy savings, especially when managing numerous joins and sort operations. Due to the large number of queries and the wide spectrum of values, plotting all results together was impractical. Therefore, we chose to present representative results for three queries (Fig. 2): Q1 (0 join, 4 sums, 3 avg, 1 sort on two attributes), Q11 (4 joins, 2 nested queries, 3 sums, 1 sort), and Q22 (4 nested queries, 2 joins, 2, one sort).

4 Conclusions

We introduced PAID, a subsystem that integrates a GA with a NN to optimize energy consumption in query processors (QPs). PAID addresses the complexity of selecting an optimal CPU configuration for a given query due to the large number of possible combinations. On the other hand, going beyond existing linear regression models, we explained NN can learn a more accurate non-linear model to predict energy and reduce energy consumption based on a query workload. Therefore, we gave evidence both techniques need to be integrated, complementing each other. Our results on the TPC-H Benchmark show that energy consumption can be reduced by up to 30%, often accompanied by improved query performance or a minor decrease. Interestingly, increasing the number of

cores led to reductions in both energy use and response time. In contrast, the effect of CPU frequency varied across query types, contradicting earlier work aiming for a single default configuration.

References

1. Bellatreche, L., Djellali, F., Macyna, W., Ordonez, C.: Energy-aware query processing: A case study on join reordering. In: IEEE Big Data. pp. 3743–3752 (2023)
2. Binglei, G., Jiong, Y., Dexian, Y., Hongyong, L., Bin, L.: Energy-efficient database systems: A systematic survey. ACM Comput. Surv. (2022)
3. Bouhatous, A., Bellatreche, L., Abdelwahed, E.H., Ordonez, C.: The impact of multicore cpus on eco-friendly query processors in big data warehouses. In: IEEE Big Data. pp. 4463–4472 (2022)
4. Dembele, S.P., Bellatreche, L., Ordonez, C., Roukh, A.: Think big, start small: a good initiative to design green query optimizers. Clust. Comput. **23**(3), 2323–2345 (2020)
5. Etinski, M., Corbalán, J., Labarta, J., Valero, M.: Understanding the future of energy-performance trade-off via DVFS in HPC environments. J. Parallel Distrib. Comput. **72**(4), 579–590 (2012)
6. Guo, B., Yu, J., Liao, B., Yang, D., Lu, L.: A green framework for dbms based on energy-aware query optimization and energy-efficient query processing. Journal of Network and Computer Applications **84**, 118–130 (2017)
7. Harizopoulos, S., Shah, M.A., Meza, J., Ranganathan, P.: Energy efficiency: The new holy grail of data management systems research. In: CIDR (2009)
8. Kunjir, M., Birwa, P.K., Haritsa, J.R.: Peak power plays in database engines. In: EDBT. pp. 444–455 (2012)
9. Lang, W., Kandhan, R., Patel, J.M.: Rethinking query processing for energy efficiency: Slowing down to win the race. IEEE Data Eng. Bull. **34**(1), 12–23 (2011)
10. Mittal, S., Vetter, J.S.: A survey of methods for analyzing and improving GPU energy efficiency. ACM Comput. Surv. **47**(2), 19:1–19:23 (2014)
11. Psaroudakis, I., Kissinger, T., Porobic, D., Ilsche, T., Liarou, E., Tözün, P., Ailamaki, A., Lehner, W.: Dynamic fine-grained scheduling for energy-efficient main-memory queries. In: DaMoN Workshop. pp. 1:1–1:7 (2014)
12. Qasaimeh, M., Zambreno, J., Jones, P.H., Denolf, K., Lo, J., Vissers, K.A.: Analyzing the energy-efficiency of vision kernels on embedded cpu, GPU and FPGA platforms. In: FCCM. p. 336 (2019)
13. Roukh, A., Bellatreche, L., Bouarar, S., Boukorca, A.: Eco-physic: Eco-physical design initiative for very large databases. Inf. Syst. **68**, 44–63 (2017)
14. Tsirogiannis, D., Harizopoulos, S., Shah, M.A.: Analyzing the energy efficiency of a database server. In: ACM SIGMOD. pp. 231–242 (2010)
15. Wang, J., Feng, L., Xue, W., Song, Z.: A survey on energy-efficient data management. SIGMOD Rec. **40**(2), 17–23 (sep 2011)
16. Xu, Z., Tu, Y.C., Wang, X.: Dynamic Energy Estimation of Query Plans in Database Systems. In: ICDCS. pp. 83–92 (2013)