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, Texas, USA

Abstract. Over the past two decades, the database and big data research communities have devoted significant efforts enhancing energy efficiency (EE) of SQL Query Processors (QPs). However, with the rising worldwide emphasis on energy conservation, developing "green" QPs has become a key concern for sustainable computing. Traditionally, these QPs were designed to optimize response time, but not energy consumption. We identify four challenges: (i) Most work focuses on optimizing Inputs/Outputs (IO), but the CPU is responsible for the majority of energy consumption, being much higher than IO operations, (ii) Most approaches exploit Machine Learning (ML) models to predict energy consumption, but without modifying the core query processing mechanisms, (iii) AI techniques remain largely unexplored to dynamically reduce query energy consumption, and (iv) existing approaches do not dynamically adjust CPU configurations (including the number of cores and frequency) to choose the most energy-efficient setting for a given 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 based on a cost model, whereas GA determines the optimal configuration, deciding CPU frequency and core allocation for each query. The GA configuration is then fed into the NN model to tune prediction accuracy. Our experiments, conducted on the TPC-H query benchmark with PostgreSQL, show that PAID effectively finds CPU configurations that exceed the performance of default settings, achieving significant energy savings.

1 Introduction

The query optimizer, a major component within DBMS, plays a central role to analyze and pre-process data. The development of efficient query processors (QP) continues to be an active area of research, with ongoing advancements and innovations regularly presented at major database conferences and in prominent journals. As data science applications proliferate, query processors have become the backbone for data preparation and data exploration phases, enabling the efficient use of machine learning and deep learning techniques.

In parallel, the ICT industry, which is comparable in size to the aviation sector, is estimated to be responsible for 1. 8% - 2. 8% of the global carbon foot-

print, according to recent research studies[4]. Therefore, growing environmental concerns will require policymakers, industry stakeholders and researchers to prioritize energy efficiency[5] (EE) in the design of future computing systems. Thus, improving the EE of QPs is also important in achieving the goals of sustainable computing. To quantify the energy consumption during query execution, we focus on query Q21 from the TPC-H Benchmark on MySQL DBMS, running on a Dell Inc. Precision 5820 Tower. This system is powered by an Intel® Xeon® W-2123 processor (3.60 GHz, 4 cores, 8 threads) and equipped with 16GB of RAM. We measure energy consumption trends across three distinct phases: **(1)** before query execution, **(2)** during execution, and **3** after execution. Prior to executing the query, the computer consumes **51.3 watts**. However, during execution, it consumes **119.145 watts**, indicating a notable variation in power consumption. After query execution, we measure energy consumption at two intervals: after 2 seconds (54.6964 watts) and after 10 seconds (52.9054 watts), respectively. This slight variation is likely due to the DBMS collecting statistics.

Aligned with this motivation, the database community has shown significant interest over the past two decades in developing various initiatives to address the challenge of EE in databases. These efforts include surveys [28,2,10], methodic studies for facilitating research on this topic [1], and prediction models for assessing the energy consumption of traditional DBMS query processors. These models have been applied to both row-oriented DBMSs, such as PostgreSQL and Oracle [7,9,24,25,2,3,30,16,15], and column-oriented DBMSs, like MonetDB [5]. These prediction models have also been utilized to select optimization structures (e.g., materialized views [24], bitmap join indexes [7], and cache management strategies [9]) that balance response time and energy consumption.

By conducting an in-depth analysis of these research initiatives, we observe that they introduce both hardware and software tactics to enhance EE [1]. Hardware manufacturers have made significant strides in developing high-performance Central Processing Units (CPUs) [4,26], Graphics Processing Units (GPUs) [18], and specialized hardware accelerators (e.g., Tensor Processing Units, FPGAs) [23]. 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 energy efficiency without significantly compromising performance.[6,22]. Software tactics include, among others, analytical cost models designed to predict the energy consumption of queries. These models extend conventional query optimizers by incorporating parameters such as IO, CPU usage, and memory costs. Machine learning (ML) techniques are then applied to process these parameters, enhancing the accuracy of energy consumption predictions and enabling more EE query execution strategies [2] (Figure 1). More recently, these models have been further refined by incorporating hardware-related parameters, such as the number of CPU cores and frequency [3]. However,

[4] https://www.databridgemarketresearch.com/whitepaper/future-of-carbon-footprint-of-information-and-communication

[5] The achievement of the same level of services while consuming less energy.

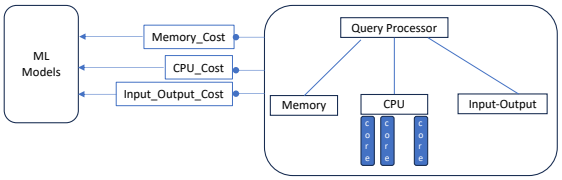


Fig. 1: Global picture of existing prediction models.

they primarily rely on basic predictive methods such as linear regression, multiple regression, and random forest [24,5,9].

The analysis of the current state of the art reveals two important findings that need to be consolidated, as well as two main limitations. The two findings are:
(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.*
The two main limitations are:
(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 energy efficiency.*

To overcome the above two limitations, we propose a novel optimization system PAID that integrates Genetic Algorithm (GA) [13] and Neural Networks (NN) [8]. 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.

To the best of our knowledge, our research work is the first to focus on selecting energy-optimal configurations for multi-core CPUs combining GA and NN. The contributions of our paper are as follows: (1) An NN-based prediction model for energy consumption and time processing, which considers Input-Output cost, Memory, CPU Number of Cores, and CPU Frequency as input features. This model is compared against a Random Forest technique used as a baseline. (2) Introduction of a GA dedicated to selecting an optimal configuration of CPU in terms of the number of cores and CPU frequency for a given query. (3) Development of a subsystem PAID that integrates a GA and NN into a DBMS (PostgreSQL).

Our paper is organized as follows: Section 2 offers a comprehensive review of related works. Section 3 provides a detailed description of our PAID subsystem, including the two main components: the NN model and GA. Section 4 outlines

the experimental setup and discusses the results obtained. Finally, Section 5 concludes the paper and presents our directions for future work.

2 Related Work

In this section, we delve into the major works proposed to examine the EE of data processors. These studies can be broadly categorized into three main approaches: (i) hardware-oriented, (ii) software-oriented, and (iii) hybrid approaches.

Hardware-oriented approaches Research efforts have focused on reducing CPU energy consumption in data storage systems. Hardware manufacturers are developing multi-core technologies and EE components, with modern CPUs incorporating Advanced Configuration and Power Interface for dynamic power management [12]. Dynamic Voltage and Frequency Scaling (DVFS) is widely used to optimize energy use, with CPU governors controlling frequency adjustments [21,14]. Studies have explored energy-saving techniques, such as Processor Voltage Control and DVFS, to balance query performance and energy consumption by lowering voltage and frequency based on workload characteristics [17]. Research has also introduced adaptive clock speed adjustments to optimize energy use in OLTP systems, achieving a 7.6% reduction in total energy consumption [11]. Experimental studies demonstrated that reducing CPU frequency significantly improves energy efficiency in main-memory DBMS [19]. Additionally, the POLARIS algorithm was developed to optimize power consumption while maintaining transaction latency requirements. It achieves this by dynamically adjusting processor frequency based on predicted execution times of ongoing and queued transactions [14].

Software-Oriented Approaches Here we review major studies on integrating energy considerations into query processors. It highlights the development of analytical cost models that predict query energy consumption using cost estimations from DBMS query optimizers. Xu et al. [29,30] proposed a cost model predicting query energy consumption based on response time in PostgreSQL, using linear regression. Kunjir et al. [15] developed a pipeline-based model with stepwise linear regression to minimize peak power consumption in query execution. Contrary to Kunjir et al. [15], Lang et al. [16] proposed an operator-based energy model for select, project, and join operations, using linear regression to guide energy-efficient query execution plans. In [24,25], the authors introduced a pipeline-structured energy model employing polynomial regression, integrated into the EnerQuery System build on the top of PostgreSQL. Guo et al. [9] examined the influence of memory and cache structures on energy consumption and developed a linear cost model incorporating memory, CPU, and I/O costs. Dembele et al. [5] etended sequential models to parallel execution environments using a combination of polynomial regression and artificial neural networks, improving energy estimation in multi-core systems with fixed settings.For more details on hardware and software solutions, please refer to [20].

Hybrid Approaches In [3], the authors conducted an empirical study to explore the impact of CPU configurations, including the number of CPU cores and their frequency, on query performance, power consumption, and energy con-

sumption during the execution of analytical queries. Building on the insights gained, they proposed an enhanced version of the simple ML model from [5], incorporating additional features related to multi-core CPUs and their frequencies. The proposed solution yielded promising results, outperforming existing major energy cost models. *However, it is important to note that these new features are derived from the default settings of the machine hosting the target DBMS, without taking into account the potential benefits of selecting the optimal hardware configuration tailored to the specific requirements of each query.*

3 Integrating old and new AI: GA and NN

Before presenting our PAID systems, we provide key definitions.

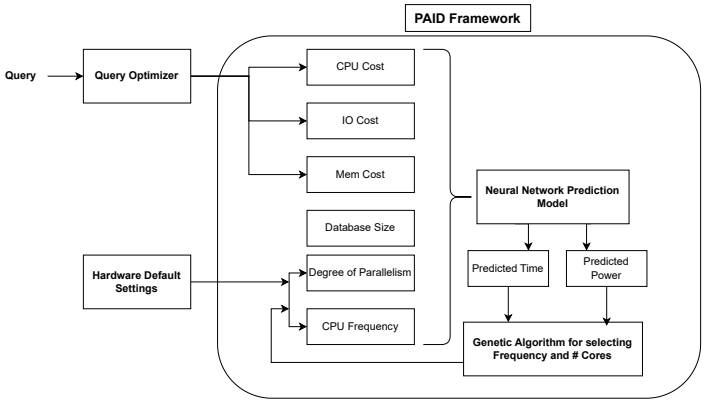


Fig. 2: The PAID Subsystem.

Definition 1. ***Power*** *is the measure of the rate at which work is performed or the amount of energy expended per time unit, generally a second. It is quantified in watts (W). More precisely, power can be defined as follows:* $P = J/T$*, where* P*,* J*,* T*, represent respectively, power in Watts, work done in Joules per time unit and a time unit, generally a second.*

Definition 2. ***Energy*** *is defined as a measure of the capacity to perform work over a time period. The unit of measurement for energy is the Joule. Formally, energy is defined as follows:* $E = P \times T$*, where* P*,* T*, and* E *represent respectively, a power in watts, a period of time in seconds, and the energy consumption in Watts-seconds.*

In the context of IT, energy typically refers to the electrical energy consumed by a computing system over a certain period of time, while power refers to the rate at which electrical energy is consumed per second [27]. For a given system, the electrical power consumption can be divided into two distinct parts:

Definition 3. ***Static Power*** *(P_{static}) that corresponds to the power required to operate, in an idle state, various components of the motherboard, such as processors, memory, IO devices and fans.*

Definition 4. ***Dynamic Power*** *($P_{dynamic}$) that refers to the power consumed during the execution of a workload.*

The total electrical power (P_{total}) consumption of a given computing system is the sum of its static power and dynamic power.

$$P_{total} = P_{static} + P_{dynamic} \tag{1}$$

We consider the average power consumed during the query execution: $P_{total}/2$.

3.1 NN model

Our `NN` model aims to estimate the energy consumption (output) considering critical CPU settings, including the number of cores (NBC) and CPU frequencies ranging from f_{min} to f_{max}, with δ increment.

According to the state-of-the-art, estimating the energy consumption for a given query Q_i requires considering the following key features: CPU cost, I/O cost, memory cost, and database size. These features can be easily extracted from data statistics and query plans generated by query optimizer calls (e.g., *EXPLAIN* in PostgreSQL and *EXPLAIN PLAN* in Oracle).

Therefore, from an energy consumption perspective, a given query Q_i can be represented by the following feature vector $\overrightarrow{Q_i}$, which serves as input to our `NN` model:
$\overrightarrow{Q_i} = (COST_{CPU_{(i,j,k)}}, COST_{IO_i}, COST_{MEMORY_i}, Tables_{size_i})$, where:
$COST_{CPU_{i,j,k}}$, $COST_{IO_i}$, $COST_{MEMORY_i}$, and $Tables_{size_i}$ represent respectively the CPU cost, the number of pages read/written from secondary storage (persistent storage), the number of pages accessed in main memory when executing Q_i, and the size of the used tables by the query Q_i. A CPU configuration used for executing a query $Q_i$is represented by the following vector: $\overrightarrow{CPU_{(i,j,k)}} = (FRQ_{(i,j,k)}, CORE_{(i,j,k)})$, where: $FRQ_{(i,j,k)}$, and $CORE_{(i,j,k)}$ represents respectively the CPU frequency j and the number of cores k 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 (frequency and number of cores) remains fixed across all queries, rather than being dynamically optimized based on individual query characteristics. In PAID, the task of filling these vectors is handled by our `GA` *which works in tandem with our* `NN`*.*

Query execution can be performed using into two distinct modes: (i) *Serial mode*: in this mode, the degree of parallelism (DOP) is limited to 1, meaning that all operators of a query are executed using a single core. In contrast, *parallel* mode allows query operators to be executed using multiple cores.

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. The parameters of our NN model are provided in Section 4.

3.2 Selecting an Optimal CPU Configuration: A Genetic Algorithm

As the number of cores on commodity processors continues to increase, selecting the optimal CPU configuration can become both time- and energy-costly. This situation motivates us to avoid the exhaustive enumeration of all possible CPU configurations in order to select the optimal one. Therefore, we propose a `GA` aimed at selecting the best CPU configuration (in terms of the number of cores and frequency) 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 (power and time) of queries under a given CPU configuration, our selection problem consists in setting the best CPU configuration that minimizes both the power consumption (F_{Power}) and execution time F_{Time}.

$$minimize_{(j,k)}F: \quad F_{power}(\overrightarrow{Q_i}, \overrightarrow{CPU_{(i,j,k)}}) \times F_{time}(\overrightarrow{Q_i}, \overrightarrow{CPU_{(i,j,k)}}) \qquad (2)$$

where $j \in \{1, 2, \ldots, NBC\}$ be the number of active CPU cores, and $j \in [f_{\min}, f_{\max}]$.

Our `GA` solves this problem, with F as the fitness function.

Our `NN` model predicts energy consumption and response time for a given query by considering static features extracted from the DBMS hosting the target database and default parameters related to the CPU, without varying them. In contrast, the `GA` selects the best configuration for a given query. By sending predicted energy from the `NN` to the`GA`, PAID enables `NN` to get the optimal CPU configuration. Figure 2 illustrates the connection between `NN` and `GA`, where the fitness function used by the `GA` is provided by the `NN` model.

4 Experimental Study

This section describes the experimental study that we conducted to validate our solution.

Setup: We conduct our experiments on a Dell Precision Tower 3620 server with the following configuration: a Dell 09WH54 motherboard, 16 GB Dual Channel DDR4 @ 2133MHz main memory, an Intel Core i7-6700 CPU @ 3.4GHz (1 CPU – 4 Cores – 8 Threads), and an SSD Disk SM951 NVMe SAMSUNG 256GB. The server has a Thermal Design Power (TDP) of 65W, and the CPU frequency can be dynamically adjusted using the DVFS technique. A detailed description of the static and dynamic power consumption of the main components of our server is provided in Table 1.

Our experimental setup consists of three main components: a client machine (monitor), a database server and a power meter called Yocto-Watt[6] (manufactured by Yoctopuce) at a frequency of 1 Hz. It is directly placed between the

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

Table 1: Static and dynamic power of our server components

Component	Static Power (Watt)	Dynamic Power (Watt)
Processor	8.97	42.90
Main Memory	2.50	4.68
Hard Disk	4.0	6.3

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 PostgreSQL DBMS (release 14.1), under Ubuntu 20.04 (kernel 20.04.4 LTS). Database statistics are also collected from the DBMS.

Datasets: We utilize the TPC-H benchmark[7] to train and evaluate our models. Three databases are generated with sizes of 10 GB, 30 GB, and 50 GB. In addition to the benchmark's original 22 queries, we generate 70 additional queries randomly.

For each database, 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 (degree of parallelism (DOP)) from 1 to 4, which corresponds to the maximum number of available cores. (=max core number), 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 as a baseline, given its demonstrated reliability in recent state-of-the-art studies.

Following that, we used a dataset comprising 50 GB and 22 TPC-H queries to validate and assess the accuracy of our cost model. Once our model was validated, we could predict the energy consumption of new queries without relying on the power meter.

Our Results: To assess our `NN` model, we implemented a Random Forest Regression (RFR) technique using the optimal parameters summarized in Table 2. This table presents the best training parameter values selected for both models after completing the training phase using the sklearn[8] tool. The RFR serves as a baseline, allowing us to compare the performance of our `NN` model and evaluate its efficiency in predicting energy consumption.

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.

[7] `https://www.tpc.org/tpch/`
[8] https://scikit-learn.org/stable/

Table 2: RFR and NN Best Parameters

RFR	n_estimators	100
	max_features	6
	max_depth	None
	min_samples_split	2
	min_samples_leaf	1
NN	Input layer	6 neurons
	1st hidden layer	32 neurons
	2nd hidden layer	16 neurons
	3rd hidden layer	8 neurons
	Output layer	2 neurons
	Activation function for hidden layers	ReLU
	Optimizer function	Adam
	Learning rate	0.0125
	Loss function	MAE

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).

Table 3 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.

Table 3: 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

For the power consumption prediction, the mean absolute error (MAE) is used as the evaluation metric. The RFR model achieved an MAE of 2.3, indicating that, on average, the predicted power consumption differed from the actual values by 2.3 watts. The `NN` model, on the other hand, achieved a slightly higher MAE of 2.5, suggesting a marginally higher prediction error compared to RFR.

This performance difference highlights that, while the NN model provides useful insights, RFR remains slightly more accurate in predicting power consumption in this particular scenario.

To assess the overall fit of the models, the coefficient of determination (R^2) is employed. The RFR model achieved an R^2 value of 63.82, indicating that it explains approximately 63.82% of the variance in power consumption. Similarly, the NN model achieved an R^2 value of 64.17, suggesting a slightly better performance in explaining the variance compared to RFR. This result indicates that while both models provide a decent explanation of the variance, the NN model performs slightly better in capturing the underlying patterns in power consumption.

For the time prediction, the evaluation metrics used are MAE and R^2. The RFR model achieved an MAE of 54.72, indicating an average prediction error of 54.72 seconds for time. On the other hand, the NN model achieved a significantly lower MAE of 13.85, suggesting a superior performance in predicting time compared to RFR. This indicates that the NN model has a more accurate and precise prediction capability for query execution time, outperforming RFR by a considerable margin.

In terms of the overall fit, the R^2 value for time prediction using the Random Forest Regression (RFR) model is 47.51, meaning the model explains approximately 47.51% of the variance in time. In contrast, the NN model achieved a much higher R^2 value of 83.64, indicating significantly better performance in explaining the variance in time. This suggests that the NN model provides a more accurate and reliable prediction for query execution time compared to the RFR model.

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 our GA in Energy Savings: We conduct an experiment to evaluate the real impact of the best configuration generated by the GA. Table 4 summarizes different parameters of our GA. To to so, we consider three CPU settings for each query:

1. **Default Energy:** This represents the energy consumption using the default configuration of the CPU.
2. **Optimized Energy:** This represents the energy consumption using the CPU settings proposed by our GA.
3. **Real Minimized Energy:** This represents the optimal energy consumption using the best CPU settings. To find the most suitable CPU configuration that minimizes the energy consumption of a query, we provide $COST_{CPU}$, $COST_{IO}$, and $COST_{MEMORY}$ as input to our GA. This later then proposes the CPU settings $\overrightarrow{CPU_{(i,j,k)}}$ that lead to minimal energy consumption. Finally, we adjust the number of cores and CPU frequency based on these settings and execute the query.

Table 4: `GA` parameters.

Parameter	**Value**
Encoding Type	Real value encoding
Selection Method	Elitist selection
Crossover Type	One-Point Crossover
Probability of Crossover	0.5
Mutation Type	Uniform mutation
Probability of Mutation	0.1
Max Evaluations	1200

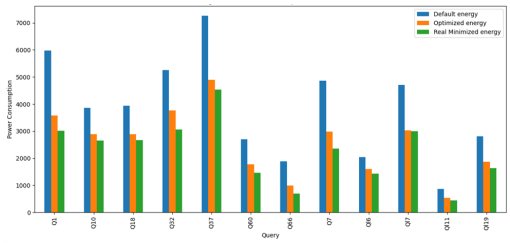


Fig. 3: Energy consumption measurements of TPC-H queries.

4.1 Connecting our NN and our GA

Figure 3 presents a summary of the results obtained. Across nearly all queries, a notable reduction in energy consumption is observed. We identified CPU configurations that achieve energy savings of up to 30% without any significant performance degradation compared to the default CPU configuration in most cases. Moreover, in addition to minimizing energy consumption, our approach also led to a decrease in execution time. Furthermore, a key observation highlighting the effectiveness of our approach is that the energy consumption and performance of queries using the CPU configurations proposed by our approach closely resemble the optimal consumption levels, demonstrating the precision and effectiveness of the configurations generated by our `GA`.

On the Joint Benefits Stemming from Integrating GA and NNs: We conducted experiments to quantify the impact of CPU frequency and parallelism, varying the number of cores, on response time and energy consumption for 22 queries, covering various analytical scenarios. Based on these results, we identified the best configuration for each query. With respect to response time, as expected, results strongly indicate that a higher number of cores yields the best response times (Fig. 4), except for Queries 3 and 22, which are less demanding than the other queries. Specifically, query 3 requires only two joins and sorting based on a single attribute, whereas query 22 contains nested queries and also involves sorting by one attribute. These differences in query complexity

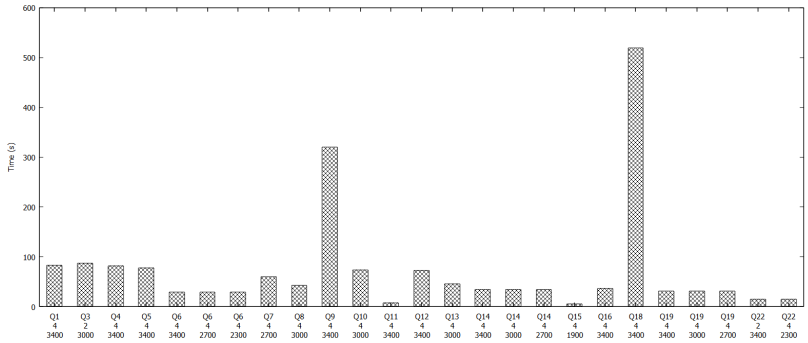


Fig. 4: Optimal configuration to achieve the best query response time per query

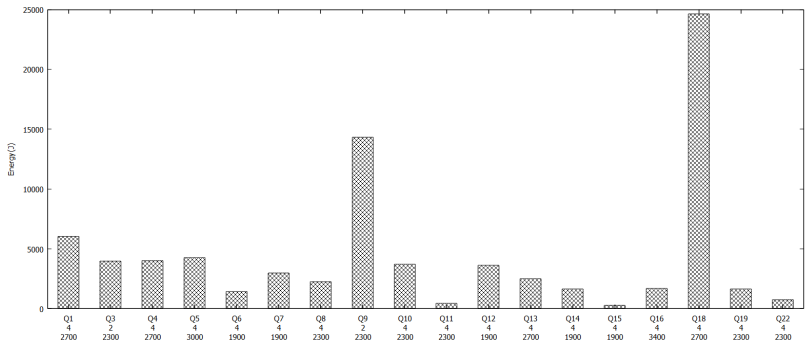


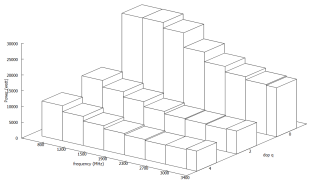
Fig. 5: Optimal configuration to achieve the best energy consumption per query

explain why the performance gains from increasing the number of cores are less pronounced for these two queries.

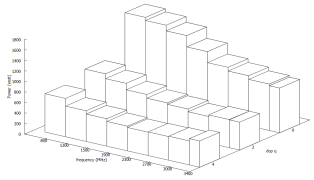
Nevertheless, we found that energy consumption is correlated to elapsed time, which is counter-intuitive. In terms of energy consumption, utilizing all available cores results in energy savings for all queries, except for Queries 3 and 9. Specifically, Query 9, which involves six joins and sorting based on two attributes, exhibits high energy consumption even when all cores are utilized. This suggests that, in certain cases, the overhead of managing parallelism or the specific nature of the query may outweigh the benefits of additional cores. The increased complexity of handling multiple joins and sorting operations in Query 9 likely leads to higher energy usage, despite the potential performance benefits from parallelism.

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: Q1, Q11, and Q22. Query Q1 involves no joins but

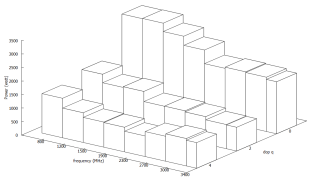
includes four sums, three averages, and a sort with two attributes. This query serves as a simple case, primarily focusing on aggregation and sorting. Query Q11 consists of four joins, two nested queries, three sums, and one sort. This query represents a more complex query structure, with multiple joins and nested operations. Query Q22 includes four nested queries, two joins, two aggregations, and a sort. This query is particularly intricate, showcasing a combination of nested queries, joins, and aggregation operations.



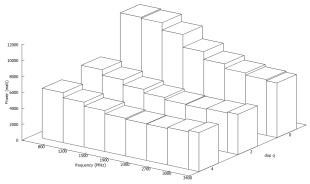
(a) The case of Query Q1



(b) The case of query Q11



(c) The case of query Q22



(d) Average of all 22 queries

Fig. 6: The impact of number of cores and CPU frequency on power savings

Figure 6d shows average energy consumption corresponding to all queries, by varying the degree of parallelism and CPU frequency.

Surprisingly, for the three queries, reveal that employing more than two cores reduces power consumption (results shown in Figures 6a, 6b, and 6c). As for CPU frequency, higher values do not necessarily translate into power savings.

An important takeaway is that employing a moderate frequency yields a compromise between energy and speed, making it our recommended solution. In addition, lower CPU frequencies lead to less CPU cooling, further reducing energy.

5 Conclusion

In this paper, we combined classical optimization techniques (genetic algorithms, popular before in AI combinatorial optimization) with modern AI prediction models (neural networks, the most popular these days for prediction) to optimize energy in a data processing system. Our optimizations are integrated into PAID, a subsystem integrated with the query optimizer. We explained there is a large number of hardware combinations, which makes selecting an optimal one,

a hard problem. On the other hand, going beyond linear regression models, we explained neural networks can learn and predict energy and consumption based on a query workload. Therefore, we argue both techniques need to be integrated, complementing each other. To validate our prototype, we conducted experiments using the TPC-H benchmark, to understand tradeoffs between energy and efficiency. We were able to reduce energy consumption by 30%, producing query acceleration as well in most cases. Additionally, results indicate that using more cores, surprisingly, reduces both energy consumption and response time, whereas the impact of CPU frequency depends on diverse query profiles (i.e. not possible to have a general solution), contradicting previous studies that rely on default settings.

There are interesting directions for future work. We would like to explore other optimization algorithms, like greedy algorithms, and dynamic programming, but we believe they will not produce a drastic difference. On the other hand, we would like to understand how the optimization problem can be rephrased as a learning problem, computable by a neural network. Our preliminary study was conducted on a server with a Solid State Drive (SSD) and Non-Volatile Memory (NVM), but in the future NVM may become prevalent, paving the way for main memory database systems. Main memory database systems can potentially produce more energy savings, keeping the CPU busy, without waiting for Inputs-Outputs. Finally, with the migration trend from on-premise to cloud-based database systems, we need to reconsider all aspects, including virtual CPUs, elastic CPU, and extending cloud database systems architectures to incorporate three dimensions: scalability, cost, but also energy.

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. Corporation, L.: Five strategies for cutting data center energy costs through enhanced cooling efficiency (2007), `http://www.rvip.ru/files/1071/Emerson_EfficiencyWP_low_041707.pdf`
5. 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)
6. 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)
7. Ghabri, I., Bellatreche, L., Yahia, S.B.: Energy efficiency vs. performance of analytical queries: The case of bitmap join indexes. In: IEEE Big Data. pp. 3066–3074 (2021)
8. Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. MIT Press (2016), `http://www.deeplearningbook.org`

9. 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)
10. Harizopoulos, S., Shah, M.A., Meza, J., Ranganathan, P.: Energy efficiency: The new holy grail of data management systems research. In: CIDR (2009)
11. Hayamizu, Y., Goda, K., Nakano, M., Kitsuregawa, M.: Application-aware power saving for online transaction processing using dynamic voltage and frequency scaling in a multicore environment. In: Architecture of Computing Systems. vol. 6566, pp. 50–61 (2011)
12. Inc, U.E.: Advanced Configuration and Power Interface Specification (2016), `http://www.uefi.org/sites/default/files/resources/ACPI_6_1.pdf`
13. Katoch, S., Chauhan, S.S., Kumar, V.: A review on genetic algorithm: past, present, and future. Multim. Tools Appl. **80**(5), 8091–8126 (2021)
14. Korkmaz, M., Karsten, M., Salem, K., Salihoglu, S.: Workload-aware CPU performance scaling for transactional database systems. In: SIGMOD. pp. 291–306 (2018)
15. Kunjir, M., Birwa, P.K., Haritsa, J.R.: Peak power plays in database engines. In: EDBT. pp. 444–455 (2012)
16. 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)
17. Lang, W., Patel, J.M.: Towards eco-friendly database management systems. In: CIDR (2009)
18. 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)
19. Noll, S., Funke, H., Teubner, J.: Energy Efficiency in Main-Memory Databases. In: Datenbank-Spektrum. pp. 335–344 (2017)
20. Ordonez, C., Macyna, W., Bellatreche, L.: Energy-aware analytics in the cloud. In: BiDEDE@SIGMOD. pp. 3:1–3:6 (2024)
21. Pallipadi, V., Starikovskiy, A.: The ondemand governor: past, present and future. In: Proceedings of Linux Symposium. pp. 223–238 (2006)
22. 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)
23. 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)
24. Roukh, A., Bellatreche, L., Bouarar, S., Boukorca, A.: Eco-physic: Eco-physical design initiative for very large databases. Inf. Syst. **68**, 44–63 (2017)
25. Roukh, A., Bellatreche, L., Ordonez, C.: Enerquery: Energy-aware query processing. In: ACM CIKM. pp. 2465–2468 (2016)
26. Tsirogiannis, D., Harizopoulos, S., Shah, M.A.: Analyzing the energy efficiency of a database server. In: ACM SIGMOD. pp. 231–242 (2010)
27. Venkatachalam, V., Franz, M.: Power reduction techniques for microprocessor systems. ACM Computing Surveys **37**(3), 195–237 (2005)
28. Wang, J., Feng, L., Xue, W., Song, Z.: A survey on energy-efficient data management. SIGMOD Rec. **40**(2), 17–23 (sep 2011)
29. Xu, Z., Tu, Y., Wang, X.: Exploring power-performance tradeoffs in database systems. In: IEEE ICDE. pp. 485–496 (2010)
30. Xu, Z., Tu, Y.C., Wang, X.: Dynamic Energy Estimation of Query Plans in Database Systems. In: ICDCS. pp. 83–92 (2013)