BigStep Towards Query Eco-Processing - Thinking Smart

Simon Pierre Dembele, Ladjel Bellatreche, Carlos Ordonez

Rubrique

BigStep Towards Query Eco-Processing - Thinking Smart

DEMBELE Simon Pierre*, Bellatreche Ladjel*, Ordonez Carlos+

Université: * LIAS/ISAE-ENSMA, + University of Houston
Ville: * Poitiers, + Houston
Pays: * France, + US
simon-pierre.dembele@ensma.fr, bellatreche@ensma.fr, carlos@Central.UH.EDU

ABSTRACT. Computers and electronic machines in businesses consume a significant amount of electricity, releasing carbon dioxide (CO2), which contributes to greenhouse gas emissions. Energy efficiency is a pressing concern in IT systems, ranging from mobile devices to large servers in data centers, in order to be more environmentally responsible. In order to meet the growing demands for awareness of excessive energy consumption, many initiatives have been launched on energy efficiency for big data processing covering electronic components, software and applications. Query optimizers are one of the most power consuming components of a DBMS. They can be modified to take into account the energy cost of query plans by using the energy cost models integrated into the optimizer in the aims to reduce the power consumption of computer systems. In this paper, we study, describe and evaluate the design of three energy cost models whose values of energy sensitive parameters are determined using the *Nonlinear Regression* technique and the *Random Forests* technique. To that end, we study in depth the operating principle of the selected DBMS and present an analysis of the performance time and energy consumption of typical queries in the TPC benchmarks. We perform extensive experiments on a physical testbed based on the PostgreSQL, MonetDB and Hyrise systems using workloads generated from the TPC benchmarks to validate our proposing.

RÉSUMÉ. Les ordinateurs et les machines électroniques des entreprises consomment une quantité importante d'électricité, libérant ainsi du dioxyde de carbone (CO2), qui contribue aux émissions de gaz à effet de serre. L'efficacité énergétique est une préoccupation urgente dans les systèmes informatiques, partant des équipements mobiles aux grands serveurs dans les centres de données, afin d'être plus respectueux envers l'environnement. Afin de répondre aux exigences croissantes en matière de sensibilisation à l'utilisation excessive de l'énergie, de nombreuses initiatives ont été lancées sur l'efficacité énergétique pour le traitement des données massives couvrant les composants électroniques, les logiciels et les applications. Les optimiseurs de requêtes sont l'un des composants les plus énergivores d'un SGBD. Ils peuvent être modifié pour prendre en compte le coût énergétique des plans des requêtes à l'aide des modèles de coût énergétiques intégrés dans l'optimiseur dans le but de réduire la consommation électrique des systèmes informatiques. Dans cet article, nous étudions, décrivons et évaluons la conception de trois modèles de coût énergétique dont les valeurs des paramètres sensibles à l'énergie sont définis en utilisant la technique de la *Régression non linéaire* et la technique des *forêts aléatoires*. Pour ce fait, nous donnons une étude approfondie du principe de fonctionnement des SGBD choisis et présentons une analyse des performances en terme de temps et énergie sur des requêtes typiques du benchmarks TPC-H. Nous effectuons des expériences approfondies basé sur les systèmes PostgreSQL, MonetDB et Hyrise en utilisant un jeu de données généré à partir du benchmarks TPC-H afin de valider nos propositions.

KEYWORDS : Green query processing - DBMS audit - Non Linear Regression technique - Random Forest Technique

MOTS-CLÉS : Optimiseurs verts - audit des SGBD - technique de la Régression Non Linéaire - technique des forets aléatoires

1. Introduction

Energy Consumption in data center is becoming one of the most important issues in today's world. The hardware designers are making efforts to produce the hardware that minimize the power Consumption. But despite of best efforts the providers of data storage and processing solutions still consumes more energy. Data are at the heart of the new world order. As mentioned in The Economist *"the world's most valuable resource is no longer oil, but data"*[1]. Data Storage Systems (DSS) *have to satisfy at the same time* two important, crucial and conflictual Non-Functional Requirements (NFR): **(1)** a rapid processing of the deluge of data issued by enterprise sources, social networks, Internet of Things, etc. and **(2)** an optimal energy consumption.

Tackling energy over-consumption, environmental issues and adopting environmentally sound practices is new important agenda for businesses, governments, organizations, associations, scientists, industrialists, ordinary and famous people around the world since climate change flags are flying everywhere. We are obliged to minimize or eliminate where possible the environmental impact of IT structure to help create a more sustainable environment. To reduce environmental problems and to create a sustainable environment, we must to rethink our life and work styles by the means of political and economical actions including certainly a review of the process of data deluge. A World Energy Council study found that without any change in our current practice, the world energy demand in 2020 would be 50%-80% higher than the 1990 levels. According to US Department of Energy (DoE) report, annual energy demand will increase from a current capacity of (363-750)$\times 10^6$ kW by 2020. The world's energy consumption today is estimated to be 2×10^9 kWh per year[28]. Such ever-increasing demand could place significant strain on the current energy infrastructure and potentially damage world environmental health by CO, CO_2, SO_2, and NO_x effluent gas emissions and global warming. Like any oil, data pollutes as mentioned in the latest Blog entry of the Martin Tisné published on July 24, 2019: *"Data isn't the new oil, it's the new CO2"*[2]. This pollution is caused by storing and processing this data.

Data storage and processing is an important subsystem in data centers, and they have evolved and increased their capacity due to the advent of new paradigms, such as cloud computing and virtualization technology. As database researchers, we are then obliged to sensible ourselves, academia, industry, IT companies, funding agencies, and students by promoting research, products, actions related to energy savings of the DSS by considering *small* and *big* initiatives. The DSS infrastructure includes hardware, software, and facility service components that support the delivery of business systems and IT-enabled processes. According to the statistics published by the InfoTech group, IT equipment consumes approximately 50% of the total energy. Figure 1 illustrates the power consumption distribution of major components that consume energy in IT infrastructure. DSSs regardless of their types (DBMSs, data centers, and parallel database machines, etc.) have been identified as one of the major energy-consuming components. This consumption is associated with their servers, storage devices, networks and infrastructure facilities such as cooling and power conditioning systems [43]. The processor consumes a major portion of energy followed by the storage device [38]. The first and major efforts in managing the

1. https://www.economist.com/leaders/2017/05/06/the-worlds-most-valuable-resource-is-no-longer-oil-bu
2. https://luminategroup.com/posts/blog/data-isnt-the-new-oil-its-the-new-co2

Figure 1 – Energy distribution among different components[39].

energy of DSS have particularly touched data centers [8] since they have been pointed out by several organization reports as one of the biggest energy consumers. Based on U.S. Environmental Protection Agency, in 2014, U.S. data centers consumed approximately 70 billion kilowatt-hours, totaling about 1.8 percent of domestic electricity consumption [3].

Making storage systems green has been addressed in many research efforts in the literature. The 2010s were a decade that marked the beginning of the integration of energy in designing query optimizers [24, 47], where hardware and software solutions have been discussed, evaluated, analyzed and implemented. Hardware research efforts got more attention than software ones. This is because several studies consider that the operating systems and firmware (hardware programs) manage energy and consequently save energy of query processors. This finding is *questionable* since they use techniques covering software (e.g., finding the best query plan satisfying the first NFR) and hardware (e.g., executing the appropriate algorithms associated with the selected plan on the target platform hosting the DBMS). DSS can benefit from existing energy-aware hardware (such as removable storage media [40], graphics cards [17], etc.). For software solutions, DBMSs need to accurately estimate hardware power consumption under static and dynamic loads, revisit query execution, scheduling, database systems design, and data management strategies in order to meet the response time and allow better energy management. This poses several challenges for the researchers: it is necessary to develop cost models and metrics that consider the parameters related to the database systems components, such as the database schema (for example, the size of the tables, the length of tuples, etc. .), the query workload (for example, type of queries, join selectivity factors and selection predicates, intermediate result sizes, etc.), hardware (for example, buffer size, disk page size, etc.) and the deployment platform (for example, RAM, disk size, number of nodes, etc.).

In this paper, we focus on building a green query processor – considered as one of the most important energy consumers of the DBMS [31] by proposing several *costs models estimating the energy consumption* of a query executed by an optimizer in several DBMS offering parallel mode with different functioning policies. The crucial aspect of our work that differentiates it from previous works [39][13][48][10][9] is the inclusion of the main memory (RAM) parameter in our cost model for PostgreSQL and MonetDB and the proposition of new model for in-memory system namely Hyrise. These cost models are proposed via statistical regression and random forest techniques. Our contributions can be summarized as follows:

3. https://www.epa.gov/sites/production/files/2017-11/documents/2017-06_0.pdf

– We provide extensive experiments to compare the data loading time, queries execution time and power consumption of TPC-H benchmark queries in in three DBMS (PostgreSQL, MonetDB, Hyrise).

– We have proposed three energy cost models by taking into account the new parameters (main memory, caches memories) in centralized DBMS to predict the energy cost of query processing. These energy cost models value are estimated via statistical regression and random forest techniques.

– Intensive experiments have been conducted to evaluate the quality of our models (regression and random forest) using TPC-H benchmarks on PostgreSQL, MonetDB and Hyrise.

We claim that building green query processors passes through a deep audit that allows designers identifying relevant energy-sensitive parameters that are the entries of the mathematical cost models estimating energy when executing a query. The development of such models necessitates a deep understanding of the functioning of the target DBMS hosting the database application (e.g., query execution mode). Note that the value of some energy sensitive parameters cannot be obtained from the statistic module of the DBMS, therefore they have to be computed using machine learning technique.

Paper outline: The outline of this paper is as follows: Section 2 summarizes the fundamental notions of energy and describes the design of the different DBMS used in our study. A consequence of energy overusing and the benefit of energy saving are also presented. Section 3 details a comparive study of DBMS in terms of Data loading time, queries processing time and power consumption. Section 4 describes the technical details of our mathematical energy cost models and presents Non Linear Regression and Random Forest techniques used to set the value of energy-sensitive parameters. Section 5 presents and interprets our experimental results. Section **??** surveys the related works while our conclusions are given in Section 7.

2. Preliminary

In this section, we introduce the concepts of energy and discuss the consequences of high energy consumption. We will also talk about the benefits of energy efficiency and describe the design of DBMS PostgreSQL, Monetdb and Hyrise.

2.1. Energy concept

Energy is a measure of the ability to do work, to change system state, it comes in many forms (magnetic energy, electrical energy, chemical energy, and nuclear energy...) and can transform from one type to another[41]. In our study, we consider electrical energy. Energy is a physical quantity dependent on time. It's measured in *Joules*. The energy transfer in one second is defined as electrical power. More precisely:

Power is the amount of a system energy per unit of time or the rate of doing a job. *Watts* is the unit of measure. Formally, energy and power can be defined as follows:

$$P(t) = d/dtE(t) \quad (1)$$

$$E(t) = \int_0^t p(\tau) d\tau \tag{2}$$

where P, t, and E represent, respectively, a power, a period, and energy. Since it is hard to guarantee the accuracy of energy measure, in this paper we use the *average power* representing the average power consumed during the execution of the workload.

Energy efficiency **(EE)** expresses the optimal use of energy to offer the same service. It is expressed by [44]:

$$EE = \frac{Useful\ Energy\ output}{Total\ Energy\ input} = \frac{Performance}{P} \tag{3}$$

Based on the above equation, we remark that there are two ways to improve EE: (i) by improving performance with the same power and (ii) by reducing power consumption without sacrifying too much performance (i.e. with a reasonable performance degeneration).

2.2. Energy overusing issues

Energy is both a solution and a problem for sustainable development. It contributes to economic development, poverty reduction, education and the general improvement of the quality of life and that of humanity, but it is also one of the main causes of pollution of air and other harmful effects on human health and the environment. Current global energy consumption is estimated at 22×10^9 kWh per year. This consumption is approximately equivalent to an emission of 6.6×10^9 tonnes of carbon dioxide (CO_2) into the atmosphere [28]. The CO_2 is a huge provider of greenhouse gases (GHG). The Organization for Economic Cooperation and Development(OECD) warns that, given current trends, energy-related emissions will increase by 70% by 2050. This can accelerate the negative consequences of climate change, including higher temperatures and an increase in the frequency of extreme events.

2.3. Energy efficiency Benefit

The reduction in energy consumption leads to the reduction of greenhouse gas (GHG) emissions and other pollutants, thus improving the quality of life in the environment. For example at the level of information systems, most computer systems are energy wasters due to the high consumption of certain electronic circuits even in the inactive state. Migration to systems that improve energy efficiency offers great prospects in businesses, as it offers many benefits such as lower energy bills, increase reliability of the electricity distribution system, reduce emissions of components polluting the atmospheric layer. The increasingly efficient and greener use of the energy required to run IT infrastructures, ensures the longevity of systems and promotes the consumption of energy from renewable sources. In fact, energy dissipation has a detrimental effect on the reliability of electronic circuits [26]. The malfunction as well as the deterioration of the components is caused by many factors, including a sharp rise in temperature, overvoltage of the terminals of the electric dipole, etc. Equipment failure rate increases when the temperature exceeds 15° Celsius [2].

2.4. Panorana of our Studied DBMS

In order to demonstrate the effectiveness of our procedure to build a green query processor, we apply our proposed models on three DBMS: PostgreSQL, MonetDB and

Hyrise. We chose to work with these systems because they are very widely used, open source and offering a parallel mode for queries processing in centralized dbms. In addition, they are different in their storage model: Row-Store for PostgreSQL, Colum-Store for MonetDB and Hybrid-store for Hyrise.

2.4. PostgreSQL: It is a row-store DBMS supporting object-relational databases. It uses the server/client model and supports the standard database languages. It offers many advanced functionalities such as user-defined types, table inheritance, sophisticated locking mechanism, foreign key referential integrity, views, rules, sub-query, nested transactions, multi-version concurrency control, and asynchronous replication [32].

The support of a parallel query involves multiple background worker processes. There is a back-end process that handles all queries issued by the connected client. This back-end consists of the following subsystems:

1) **Parser:** it checks the query syntax expressed in a high-level query language like SQL to determine whether it is well formulated according to the grammar rules of the query language.

2) **Analyzer:** The query must be validated by verifying that all attributes and relationship names are valid and semantically significant in the schema of the database.

3) **Rewriter:** Using transformation rules, an internal representation of the query is then created (query tree).

4) **Planner:** It generates the cheapest plan tree that can be executed from the query tree.

5) **Executor:** It executes the query via accessing the tables that are generally stored in a single heap, with each column stored on a single tuple. A query has many possible execution strategies, and the selection of the best plan is usually conducted by cost model-driven strategies [11].

Figure 2 summarizes the different phases of the PostgreSQL query processor.

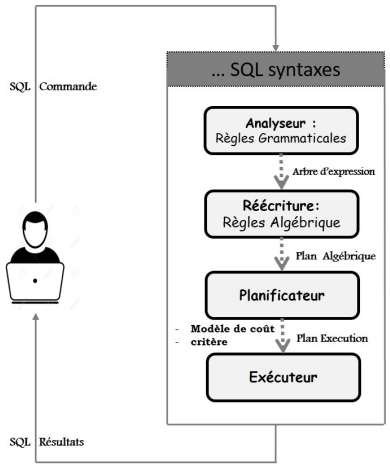


Figure 2 – The main steps PostgreSQL query processor

2.4. MonetDB: It was designed primarily for data warehouse applications. It achieves significant speedup compared to more traditional designs by innovations at all layers of a DBMS, e.g., a storage model based on vertical fragmentation (column-store), a modern CPU-tuned query execution architecture, adaptive indexing, run-time query optimization, and a modular software architecture[18].

Internally, the design, the architecture and the implementation of MonetDB reconsider all aspects and components of classical database architecture and technology by exploiting effectively the potentials of modern hardware.

1) **Storage model:** It is a significant deviation of traditional database systems. It uses the decomposed storage model (DSM) which represents relational tables using vertical fragmentation, by storing each column in a separate $\#surrogate, value\#$ table, called binary association table (BAT). The left column (the surrogate or object-identifier (oid)) is called the head, whereas, the right column is the tail. MonetDB executes a low-level relational algebra called the BAT algebra. Data in execution is always stored in (intermediate) BATs, and even the result of a query is a collection of BATs [5]. During the query evaluation, all intermediate results are in a column format. Only just before sending the final result to the client, $N-ary$ [4] tuples are constructed. This approach allows the query engine to exploit CPU- and cache-optimized vector-like operator implementations throughout the whole query evaluation relying on a bulk processing model allowing minimizing function calls, type casting, various metadata handling costs, etc. Intermediate needed results are materialized to increase their reuse [19].

2) **Query execution model:** The MonetDB kernel is an abstract machine, programmed in the MonetDB Assembly Language (MAL). The core of MAL is formed by a closed low-level two-column relational algebra on BATs. N-ary relational algebra plans are translated into two-column BAT algebra and compiled to MAL programs. These MAL programs are then evaluated in an operator-at-a-time manner. Figure 3 shows the internal design of MonetDB.

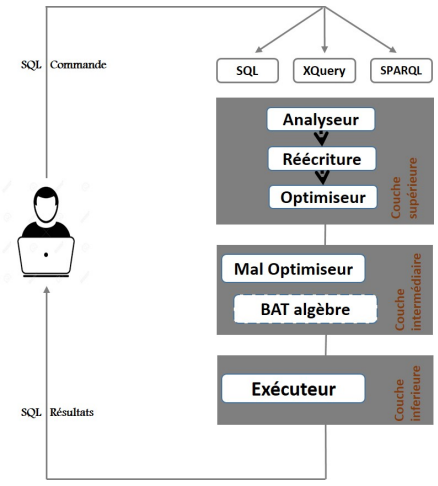


Figure 3 – Internal Architecture of MonetDB

4. https://www.cs.odu.edu/~toida/nerzic/content/relation/definition/cp_gen/index.html

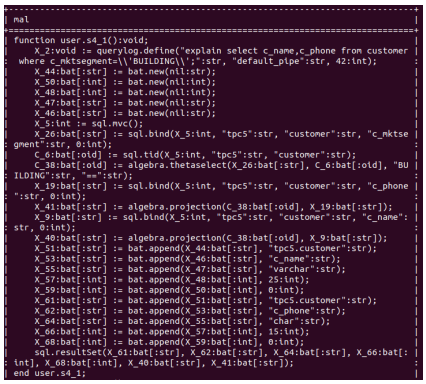


Figure 4 – MonetDB architecture: MAL plans program produced by the command *explain*.

MonetDB's query processing scheme is centered around three software layers:

– *The top layer or front-end* provides the user-level data model and query language. The query language is first parsed into an internal representation (e.g., SQL into relational algebra), which is then optimized using domain-specific rules. In general, these domain-specific strategic optimizations aim primarily at reducing the amount of data to be processed (i.e., the size of intermediate results). The optimized logical plan is then translated into MAL [18]. Figure 4 illustrates the MAL plan produced for the following query.

```
SELECT c_name,c_phone
FROM CUSTOMERS  WHERE customers.city='London';
```

– *The middle layer or back-end* consists of the MAL optimizers framework and the MAL interpreter. The MAL optimizers framework consists of a collection of optimizer modules, where each MAL code is transformed in one more efficient by adding resource management directives if necessary. The modules provide facilities ranging from symbolic processing up to just-in-time data distribution and execution. The approach breaks with the hitherto omnipresent cost-based optimizers by recognizing that not all decisions can be cast together in a single cost formula [18].

– *The bottom layer or kernel* provides BATs as MonetDB's bread-and-butter data structure, as well as the library of highly optimized implementations of the binary relational algebra operators. They maintain properties over the object accessed to gear the selection of subsequent algorithms. For example, the Select operator can benefit both from sorted-ness of the BAT or it may call for a sample to derive the expected sizes[18].
For query parallel execution in Monetdb, a sequential execution plan is generated firstly and parallelization is then added in the second optimization phase. The individual MAL operators are marked as either $blocking$ or $parallelizable$. The optimizers will alter the plan by splitting up the columns of the largest table into separate chunks, then executing the $parallelizable$ operators once on each of the chunks, and finally merging the results of these operators together into a single column before executing the blocking operators[33].

2.4. Hyrise: Hyrise [5] was first presented in 2010, developed by *Hasso-Plattner-Institute* in Germany under the direction by Professor Hasso Plattner to introduce the concept of the hybrid storage model (row-oriented and column-oriented) for in-memory databases. The

5. https://hpi.de/plattner/projects/hyrise.html

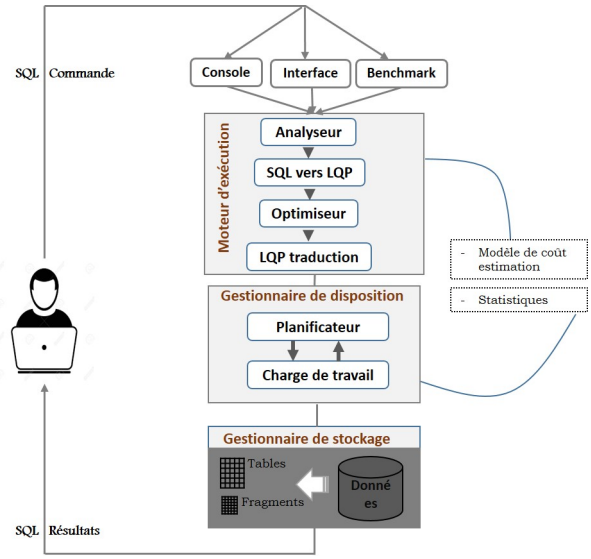


Figure 5 – Internal Architecture of Hyrise

Hyrise system is a new implementation based on the relational model, which automatically splits database tables vertically and then horizontally into small partitions, of varying widths depending on how the columns of the table are accessed [12]. The system aims at maximizing the performance of the cache memory for analytical(OLAP-type) and transactional (OLTP-type) queries by customizing the dynamic design of storage data. The goal of the Hyrise designers is to provide a flexible in-memory data management platform, allowing experiments to be conducted to validate new concepts around in-memory data management.

The overall architecture of the hyrise system consists of three main components: a storage manager, a query execution engine and a layout manager shown in figure 5. The storage manager is responsible for creating and maintaining the hybrid containers that store data. Storage is based on the vertical partitioning of relational tables into disjoint sets of attributes. Each partition is represented internally as a container. The containers are stored in a single compressed block in main memory (RAM) and can be merged after decompression to build the initial relational table. The layout manager analyzes a given query workload and demands the best possible layout (partitioning) for that workload from the storage manager.

3. Comparative analysis of our DBMS

This section presents an experimental study comparing our three DBMS in terms of rapid query performance, reduction of power consumption, data loading time and number of cahes misses without exploiting our models when queries executing. These experiments are conducted using the first 6 (SPJ) queries of the TPC-H benchmark. The system under which we perform our assessments is described in section 3.1.

3.1. Experimental Setup

Hardware: The experiments are conducted using a DELL PRECISION T1700 with Dell 073MMW motherboard(Socket 1150 LGA), Intel Core i7 4770 CPU@ 3.40GHz Haswell 22nm Technology(1 CPU-4Core-8 Threads), 12Go Dual-Channel DDR3@ 798MHz main memory, NVIDIA NVS 310 511 MB, ATA Disk Western Digital(WD) 500 GB.

OS: We run all the experiments using Ubuntu 18.04 bionic as the operating system with kernel version 5.0.0-27-generic.

Datasets: We use the TPC-H decision support benchmark, which is designed for evaluating data warehouses. It is composed of eight tables *(PART, PARTSUPP, REGIONAL, SUPPLIER, CUSTOMER, LINEITEM, NATION, and ORDERS)*, we generate data at different scale factors 1 GB, 2GB, 3GB, 5GB, 10GB, 30GB [6]. We chose to use benchmark from TPC because they are known and have many years skilling in the definition of transaction processing and database benchmark.

DBMS Systems: We analyse three systems: an open-source row-store PostgreSQL version release 10.10, an open-source column-store MonetDB release 11.33.11 and the open-source hybrid-store Hyrise release V2.

Profiling: After generated the data from TPC-H benchmarch in textual(comma-separated values - CSV) files, we use COPY command from SQL standard to load data in DBMS. We collect statistics about average time and power consumption for each query in the particular DBMS. For power measurment, we use power meters called Watt-UP-PRO [7] at a 1Hz frequency. For cahes misses analyses, we use Cachegrind tool [8] and Processor Counter Monitor(PCM) [9].

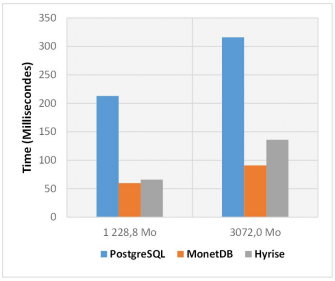
3.2. Experimental Evaluation

We conduct several experiments to evaluate the data loading performance, query execution performance and power consumption of the tested systems. We present in this section our finding.

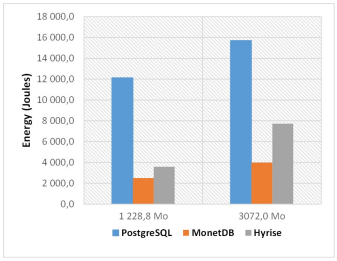
– **Data loading and Power Analysis:** This experiment investigates the behavior of PostgreSQL, MonetDB and Hyrise when loading the two datasets of size 1GB and 3GB.

Figure 6a plots the data load time for each DBMS under the two datasets. As we can see, the data load time increases for each system as the size of the dataset increases. Monet outperforms the rest of the DBMS in the two cases; when considering 3GB datasets, Monet is 3.5 faster than PostgreSQL and 1.5 faster than Hyrise. After Monet, Hyrise exhibits fastest loading time than PostgreSQL. PostgreSQL shows the worst performance, the reason is that the row-stores require more space because they store auxiliary information in each tuple (e.g., a header) and do not use compression. The increased data loading time into the hyrise system than monet is due to the additional processing required for data compression. Hyrise applies a higher compression ratio than Monet because monet only compresses string values. Like the data loading time, the energy consumed during this phase is shown in figure 6b. Energy is a physical quantity influenced by the interval of time that elapses. In the figure, we notice that the energy consumed by Postgresql is more considerable than that of monetdb and hyrise.

6. Giga Byte
7. http://www.energyalternatives.ca/pdf/wattsup_TTW.pdf
8. https://valgrind.org/docs/manual/cg-manual.html
9. https://github.com/opcm?tab=repositories



(a) Data loading time

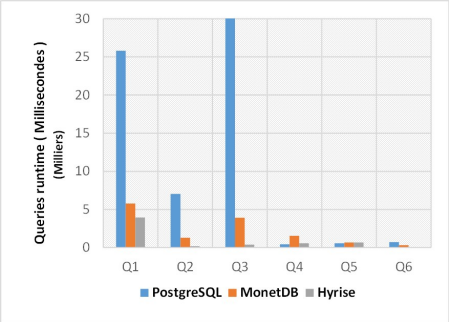


(b) Energy Consumption

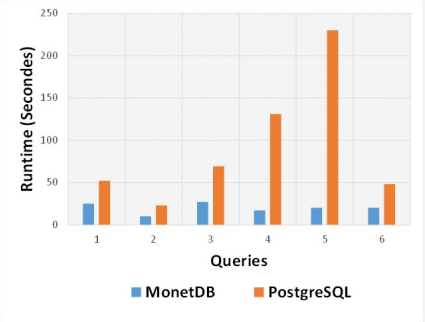
Figure 6 – Data loading time and Energy Consumption

– **Query Performance and Power Analysis:** In this experiment, we evaluate the runtime and power consumption of the Postgresql and Monetdb systems when performing queries on the two database configurations: 5 GB and 30 GB. Then, we analyze with the dataset of size 2GB the behavior of the three systems (Postgresql, monetDB,hyrise). Lastly, elapsed time for each SPJ query on the same database has been compared. Figures 7b and 7c summarize the average execution time results obtained using the SPJ queries for PostgreSQL and MonetDB. Figure 7a represents the average execution time for three systems on the 2 GB dataset. These results show that the performance of Hyrise outperforms those obtained by PostgreSQL and MonetDB on the 2GB dataset for all considered. We can see that for most queries, Hyrise performance is within an order of magnitude of the other databases. The PostgreSQL performance gaps on Figure 7a,7b,7c is mainly due to the fact that it cumulates an important number of page faults when processing queries. The row-store systems have to scan and use the entire n-tuples rather than only the needed columns values. Therefore, the entire rows plus the built-in index tree cannot reside long enough in the main memory or in the cache memories. They must be swapped on the disk this leads to many disks IOs. For column-oriented systems like MonetDB, just the values of the columns required to answer the queries are loaded. hybrid system like hyrise combines the advantages of column and row storage systems and it is well designed to support efficient parallel execution for multi-core processors.

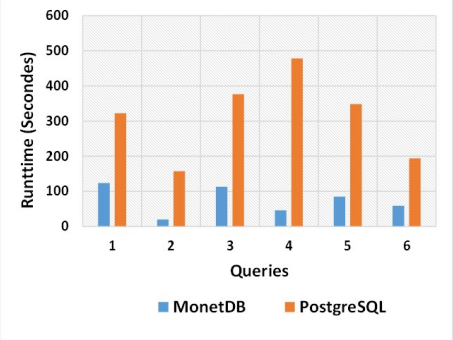
To compare the energy consumption on our three systems when executing queries, we measure the power dissipated per second using an amperage apparatus named wattmeter. At the end of these measurements, we calculate the average energy consumed by each query. Figure 8c depicts the energy consumption of three DBMS on 2GB datastes. Hyrise outperform PostgreSQL and MonetDB under the 2GB TPC-H dataset because of the rapdid processing of queries. Hyrise directly accesses compressed data in main memory and not in secondary memories, this speeds up data processing. Average energy consumed by *MonetDB*, *PostgreSQL* during the execution of the queries on SF5 and SF30 can be found in Figures 8a and 8b. *MonetDB* energy consumption during queries execution is reasonable than for *PostgreSQL*. The high consumption of the PostgreSQL query processor can be explained by the characteristics of this type of row-store DBMSs. MonetDB uses compression techniques to reduce the cost of data scans. This directly impacts query performance due to fewer I/O requests and page faults. To understand *Where does power go* on PostgreSQL DBMS, we captured the last level cache(LLC) misses and Cache level 2(L2) misses. We execute the first three queries of TPC-H benchmark on the 2GB datasets.



(a) On TPCH SF2



(b) On TPCH SF5



(c) On TPCH SF30

Figure 7 – Execution time comparison using $Q_1, Q_1, Q_2, Q_3, Q_4, Q_5, Q_6$ queries from TPC-H benchmark in parallel mode

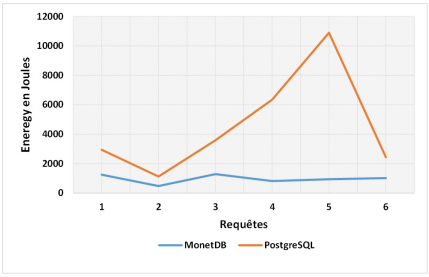
The results for these queries are shown in Figure 5. From these results illustred, the LLC caches misses and L2 caches misses of Hyrise outperforms the row-store DBMS (PostgreSQL) and column-store(MonetDB).

4. Energy Cost Models Process

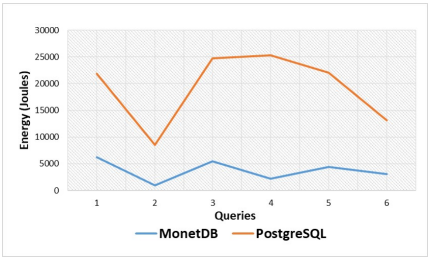
In this section, we describe our models that estimates the power consumption when executing query based on settings of a particular system. An energy model can be represented in the form of equations, graphical models, decision trees, neural networks, etc. The choice of representation affects the accuracy of the model. An accurate model of power consumption is essential to improve energy efficiency optimizations, the best model should be accurate, fast, generic, portable, simple, non-intrusive and low cost [8].

4.1. Modeling process

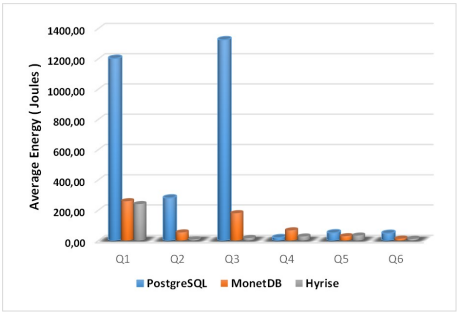
Inspired from [8], we establish the main steps of building green query processors are: (1) identification of relevant energy-sensitive parameters belonging to hardware and software components, (2) elaboration of mathematical cost models estimating consumed



(a) On TPCH SF5



(b) On TPCH SF30



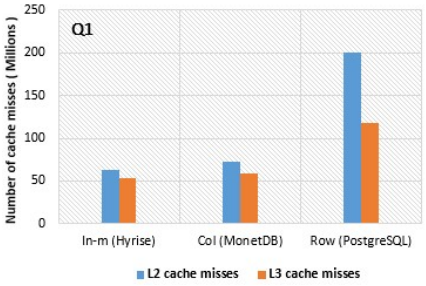
(c) On TPCH SF2

Figure 8 – Average energy consumption for PostgreSQL, MonetDB, Hyrise using $Q_1, Q_1, Q_2, Q_3, Q_4, Q_5, Q_6$ queries from TPC-H benchmark.

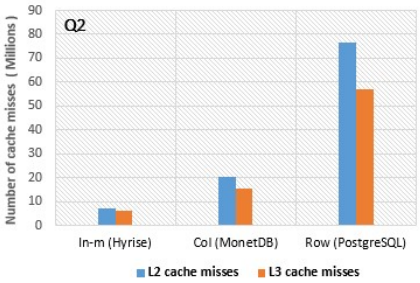
energy when executing a query on a target DBMS and (3) setting of values of the energy-sensitive parameters using machine learning techniques.

4.1. Step 1: (Energy-sensitive parameters identification) Regarding the extraction of energy-sensitive parameters, we have identified a set of parameters that cover the main components of a DSS, that we have classified them into four categories as it is sketched in Figure 10.

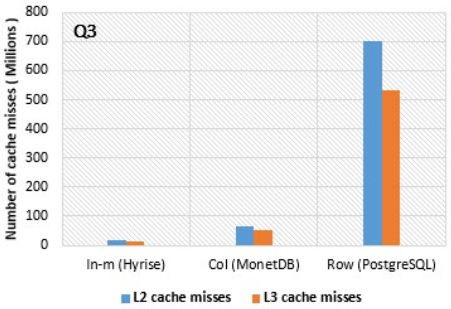
4.1. Step 2: (Energy-cost model construction) Our work focuses on modeling energy consumption of query operations related to its resources consumption on a single-node database environment, therefore we do not consider the network transmission cost. To construct our model, we investigate the executor tasks to understand how to profile the power consumption on an individual query. To do that, we execute a set of queries (simple and complex) with different scale of factors of the TPC-H benchmark. In our process, we treat energy as the resource consumed by systems operation during queries execution. The input parameters (resources) that we consider are listed in Table 1 for each systems. From Table 1, note that our models consider the main resources responsible for the energy consumption of each system. For in-memory system Hyrise, the cost of access to secondary



(a) Query 1-Cahe misses



(b) Query 2-Cache misses



(c) Query 3-Cache misses

Figure 9 – LLC and L2 cache misses for Q_1, Q_2, Q_3 queries from TPC-H benchmark.

memories is zero but it should be noted that the cost of the different caches memories ($L1$ [10], $L2$ [11], $L3$ [12]) is not uniform.

On PostgreSQL, we consider that queries are executing in pipelined fashion [36]. On MonetDB the data are processed in an *operator at a time* manner. In this processing model, the operator processes the entire column at once before moving on to the next operator because the intermediate result of every operator has to be materialized. For Monetdb, we adopt an operations-based modeling approach. Hyrise follows a push-based operator

10. Cache Memory Level 1
11. Cache Memory Level 2
12. Cache Memory Level 3

	Item	DBMS Systems		
		PostgreSQL	MonetDB	Hyrise
Input Parameters	Processor (cpu)	×	×	×
	Main Memory (mem)	×	×	×
	Disk (dio)	×	×	-
	L1 Cache memory(L1)	-	-	×
	L2 Cache memory(L2)	-	-	×
	L3 Cache memory(L3)	-	-	×

Table 1 – Parameters used in ours models

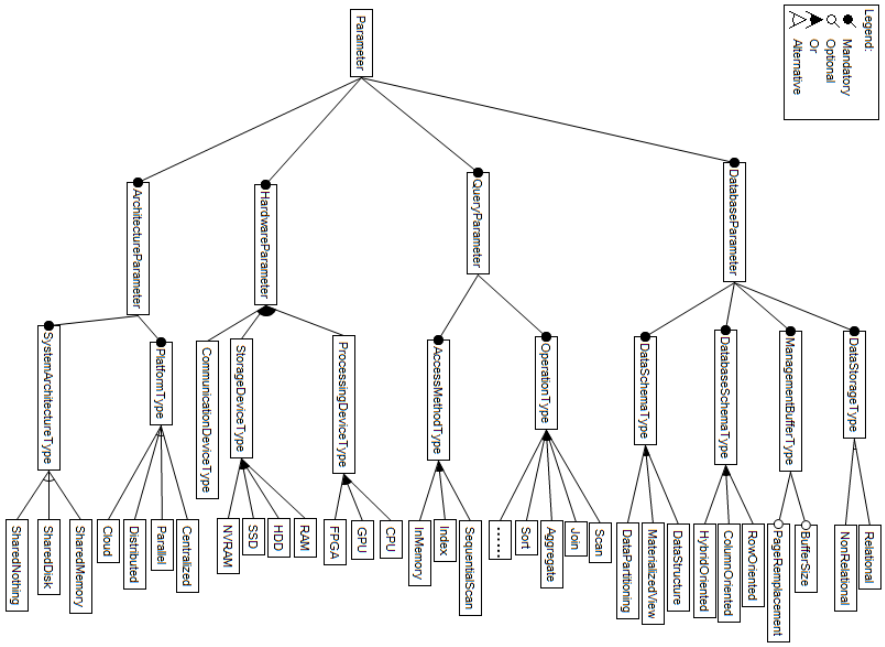


Figure 10 – Energy-sensitive parameters

model. Each operator generates a valid table, which is the only data entry for the following operator.

For a given query plan of Q_i executing in PostgreSQL, denoted by $PlanPost_i$ consisting by k pipelines noted $\{PL_1^i, PL_2^i, PL_3^i, \dots PL_k^i\}$, its average power cost is estimated as follows: $Power(Q_i) = \frac{\sum_{j=1}^{k} Power(PL_j^i) * Time(PL_j^i)}{Time(Q_i)}$, where $Time(Q_i), Time(PL_j^i)$ represent respectively, the execution time of the query Q_i and the execution time of PL_j^i. The power dissipated when processing the query is the combination of main identified resources (*CPU, Main memory and Disk*) energy consumption.

$$Power(PL_j^i) = W_{cpu} * \sum_{u=1}^{n} C_{cpu_u} + W_{mem} * \sum_{u=1}^{n} C_{mem_u} + W_{dio} * \sum_{u=1}^{n} C_{dio_u} \qquad (4)$$

where W_{cpu}, W_{mem} and W_{dio} are the model parameters. W_{cpu}, W_{mio} and W_{dio} are unit-power cost for instructions, read/write operations on memory and read/write operations on disk respectively. The C_{cpu_u} is the number of instructions executed by CPU. C_{mem_u} is the number of read or write operations accessed on memory. C_{dio_u} is the number of read or write operations accessed on disk. The n is the number of operators in the pipeline. u is the summation index.

For MonetDB, the energy cost for a given query plan (Q_i) denoted by $PlanMonetDB_i$ consisting by k operations noted $\{OP_1^i, OP_2^i, OP_3^i, \dots OP_k^i\}$ is estimated as follows: $Power(Q_i) = \frac{\sum_{j=1}^{k} Power(OP_j^i) * Time(OP_j^i)}{Time(Q_i)}$, where $Time(Q_i), Time(OP_j^i)$ represent respectively, the execution time of the query Q_i and the execution time of OP_j^i. The power

dissipated when processing is the combination of the energy consumption of the main resources identified. The formula is given by the following equation:

$$Power(OP_j^i) = W_{cpu} * C_{cpu_j} + W_{mem} * C_{mem_j} + W_{dio} * C_{dio_j} \quad (5)$$

Each Hyrise plan invokes a GetTable operator at the start. The latter is responsible for loading the table name and data blocks from the storage manager (from main memory to the CPU). The power dissipated when processing is the combination of the energy consumption of the main resources identified(CPU, main memory and caches memories). For a given query plan (Q_i) denoted by $PlanHyrise_i$ consisting by k operations noted $\{OP_1^i, OP_2^i, OP_3^i, \ldots OP_k^i\}$ is estimated as follows: $Power(Q_i) = \frac{\sum_{j=1}^{k} Power(OP_j^i) * Time(OP_j^i)}{Time(Q_i)}$, where $Time(Q_i)$, $Time(OP_j^i)$ represent respectively, the execution time of the query Q_i and the execution time of OP_j^i. The formula is given by the following equation:

$$Power(OP_j^i) = W_{cpu} * C_{cpu_j} + W_{L2} * C_{L2_j} + W_{L3} * C_{L3_j} + W_{mem} * C_{mem_j} \quad (6)$$

where W_{L2} , W_{L3}, W_{mem} and W_{CPU} are the model parameters. W_{L2} and W_{L3} are unit-power for memory cache level 2 and 3. The C_{L2_j} is the miss count of L1 cache and C_{L3_j} is the miss count of L2 cache.

4.1. Step 3: (Machine learning) To identify the different values of the power unit cost in the equations 4 , 5, 6, we use the Regression techniques at first and then the Random Forest(RF) regression.

1) **Regression Technique**: Regression is a predictive technique used in machine learning. It consists in analyzing a relationship between two quantitative variables and using it to estimate the unknown value of one using the known value of the other. It is commonly used in management and marketing techniques, profit forecasts, business planning, financial forecasting, time series prediction, biomedical modeling, and environmental modeling. There are different families of regression functions and different ways of measuring the error rate. They can be of a simple linear model, multiple linear, nonlinear, etc.[35].

In our work, we use a non-linear regression technique involving the identified energy-sensitive variables. To do that, we set to 2 the polynomial degree for PostgreSQL and Hyrise. For MonetDB, we use the degree 3. We find the values of parameters of our equations: W_{cpu}, W_{mem}, W_{L2},W_{L3} and W_{dio} by applying this setting in the regression package defined in the R language.

2) **Random forest** : Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. It was introduced by Breiman and Cutler. Recent studies have suggested that Random Forest offers features which make it very attractive algorithm for statistical learning. The method is based upon an ensemble of decision trees, from which the prediction of a continuous variable is provided as the average of the predictions of all trees. In RF regression, an ensemble of regression trees is grown from separate bootstrap samples of the training data using the CART [13] algorithm. The branches in each tree continue to be subdivided while the minimum number of observations in each leaf is greater than a predetermined value[30]. There are three possible training parameters for Random Forest:

13. https://fr.wikipedia.org/wiki/Algorithme_CART

- **ntree:** the number of trees in the Forest;
- **mtry:** the number of different descriptors tried at each split;
- **node size:** the minimum node size below which leaves are not further subdivided.

Random Forest includes a method for assessing the importance of descriptors to the model. Random Forests were trained using the random Forest library in the statistical computing environment R.

4.2. Models training

4.2. Data Sets: The data set was selected from TPC-H Benchmark [14]. The TPC-H Benchmark (TPC-H) is a decision support benchmark. It consists of a suite of business oriented ad-hoc queries and concurrent data modifications. The queries and the data populating the database have been chosen to have broad industry-wide relevance. This benchmark illustrates decision support systems that examine large volumes of data, execute queries with a high degree of complexity, and give answers to critical business questions. We generate data at different scale factors: 1 GB, 2GB, 3GB, 5GB, 10GB, 30GB. For MonetDB and PostgreSQL models training, we study and collect the characteristics of forty-four (44) Select-Project-Join (SPJ) queries and measure the energy consumed by each of them using the power meter. In the case of Hyrise, we collect the statistics of ninety-nine (99) Select-Project-Join (SPJ) queries. For the test set, we use the QGEN [15] tool from TPC-H benchmark to generate twenty-two (22) Select-Project-Join (SPJ) queries. An analytical study of training and test set prediction is provided at the end of the Model Evaluation in section 5.

4.2. System settings: Our experiments run on the server with an Intel Core i7 4770 processor (L1 cache size is 32KB, L2 cache size is 256KB and L3 cache size is 8MB), 12GB DDR3 main memory and a 500GB SATA hard drive for Hyrise system model training. For MonetDB and PostgreSQL, we use a DELL E6430 (Intel Core i5 3340 processor (1 CPU - 2 Core - 4 Threads), 8GB DDR3 main memory, ATA Disk Toshiba 500 GB). We use power meters called Watt UP PRO at a 1Hz frequency placed between the electrical power source and the database server to quantify the system consumption. The environment topology of our experimentation is shown in Figure 11.

4.2. Operations Counting: Many modern processors contain a performance monitoring unit (PMU). The design and functionality of a PMU is CPU-specific. We use Cachegrind [16] and PCM [17] to get some statistical informations from PMU when querying evaluation. Cachegrind simulates how the program interacts with a machine's cache hierarchy. Processor Counter Monitor (PCM) is an application programming interface (API) and a set of tools based on the API to monitor performance and energy metrics of Intel Core, Xeon, Atom and Xeon Phi processors. We also use tool like explain, trace to collect information about tables accessed. EIST [18] (Enhanced Intel SpeedStep Technology), an Intel DVFS implementation, changes CPU frequency and voltage to improve energy efficiency. This

14. http://www.tpc.org/tpch/
15. https://github.com/electrum/tpch-dbgen
16. https://valgrind.org/docs/manual/cg-manual.html
17. https://github.com/opcm/pcm
18. https://en.wikipedia.org/wiki/SpeedStep

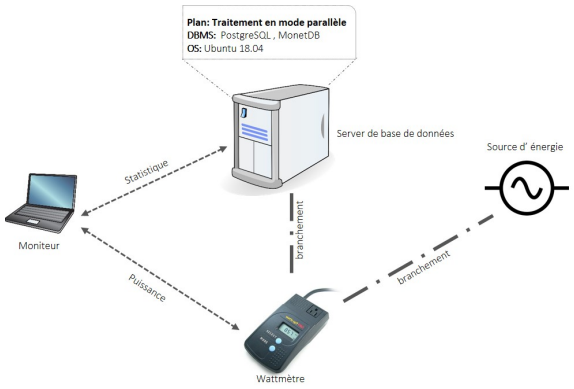


Figure 11 – Deployment of our experimental testbed.

technique can incur errors in energy measurement. We turn off these options and execute our queries under the given frequency and voltage.

5. Energy Model Evaluation Results

In this section, we will evaluate our energy model and present the results obtained after intensive experimentations on PostgSQL, MonetDB, Hyrise.

5.1. Parameters values

After having made a series of observations in which the queries chosen for models training are executed one after the others, we use regression and Random Forest in R language to determine parameters values for our models describe in equations 4, 5, 6. The following equations 7, 8, 9 describe our model equations with parameter values for the three systems respectively for MonetDB, PostgreSQL and Hyrise.

$$\begin{aligned} P(OP_j^i) = {} & -2,44 * 10^{-7} * C_{cpu} + 9,17 * 10^{-5} * C_{mem} + 1,99 * 10^{-6} * C_{dio} + 1,51 \\ & * 10^{-13} * C_{cpu} * C_{mem} - 7,22 * 10^{-11} * C_{mem} * C_{dio} + 1,67 * 10^{-13} * C_{cpu} \\ & * C_{dio} - 2,82 * 10^{-18} * C_{cpu} * C_{mem} * C_{dio} + 3,27 * 10^{-14} * C_{cpu}^2 - 1,46 \\ & * 10^{-9} * C_{mem}^2 - 1,56 * 10^{-12} * C_{dio}^2 - 3,74 * 10^{-19} * C_{cpu}^2 * C_{mem} + 1,07 \\ & * 10^{-16} * C_{cpu} * C_{mem}^2 + 9,67 * 10^{-21} * C_{cpu}^2 * C_{dio} + 2,93 * 10^{-16} * C_{mem}^2 \\ & * C_{dio} + 1,83 * 10^{-20} * C_{cpu} * C_{dio}^2 + 2,58 * 10^{-17} * C_{mem} * C_{dio}^2 + 3,87 \\ & * 10^{-21} * C_{cpu}^3 - 2,30 * 10^{-15} * C_{mem}^3 + 2,07 * 10^{-19} * C_{dio}^3 + 47,26 \end{aligned} \tag{7}$$

$$\begin{aligned} P(PL_j) = {} & 4,53 * 10^{-6} * C_{dio} - 1,46 * 10^{-6} * C_{mem} - 1,12 * 10^{-6} * C_{cpu} + 2,30 * 10^{-14} \\ & * C_{cpu} * C_{mem} - 5,88 * 10^{-12} * C_{mem} * C_{dio} + 5,97 * 10^{-15} * C_{cpu} * C_{dio} \\ & + 2,46 * 10^{-17} * C_{cpu}^2 + 8,01 * 10^{-12} * C_{mem}^2 - 9,25 * 10^{-13} * C_{dio}^2 + 48,8 \end{aligned} \tag{8}$$

Variables	PostgreSQL		Monet		Hyrise	
	NLR	RF	NLR	RF	NLR	RF
Multiple R-squared	0.83	-	0.69	-	0.61	-
Adjusted R-squared	0.78	-	0.45	-	0.54	-
Residual interval	$]-2,2[$	-	$]-3,3[$	-	$]-10,15[$	-
Root Mean (RMSE)	-	1.62	-	0.79	-	4.35

Table 2 – Regression statistics and residual outputs

$$\begin{aligned} P(OP_j^i) = {} & 23,61 - 4,63 * 10^{-10} * C_{cpu} + 3,53 * 10^{20} * C_{cpu}^2 - 7,30 * 10^8 * C_{L2} - 1,69 \\ & * 10^{-17} * C_{cpu} * C_{L2} + 6,59 * 10^{-15} * C_{L2}^2 + 9,02 * 10^{-6} * C_{L3} + 1,96 \\ & * 10^{-16} * C_{cpu} * C_{L3} - 1,43 * 10^{-13} * C_{L2} * C_{L3} + 8,67 * 10^{-14} * C_{L3}^2 \\ & + 1,85 * 10^{-7} * C_{mem} + 1,99 * 10^{-17} * C_{cpu} * C_{mem} - 2,20 * 10^{-14} \\ & * C_{L2} * C_{mem} + 2,39 * 10^{-13} * C_{L3} * C_{mem} + 1,92 * 10^{-14} * C_{mem}^2 \end{aligned} \quad (9)$$

The Random Forest was trained upon all the statistics collected from the queries. Training parameters were set by changing each parameter one-by-one and by regenerating the regression model. The optimum value of each parameter was selected from the following ranges: mtry- from 1 to 300; ntree-from 1 to 3. For nodesize value, it is fixed to the default. After analysis of output, the parameters were selected with values : for Hyrise model training(mtry=3 and ntree=260), for PostgreSQL(mtry=2 and ntree=30) and for monetdb(mtry=2 and ntree=40). To assess the quality of our models with the different machine learning techniques, the fit to the training data was taken into account. Table 2 gives more information about the fit to the training data. We give for each models the correlation's coefficient (squared correlation) denoted by R^2 , Adjusted R-squared and Root Mean of Squared Error(RMSE).

5.2. Models Validation

In this subsection, we present the results of a series of experiments running the standard workload provided by the two benchmarks TPC-H and TPCDS [19]. We used our energy models (Non-Linear Regression or Random Forest) defined in equations 7 ,8 and 9 to execute the 22 queries on a $5GB$ data size for PostgreSQL and Monetdb. for Hyrise, we run the queries on $2GB$ and $3GB$. We compare the energy estimated (EE) by our cost models with the real energy (ER) consumed by the database server during the processing of the query in parallel mode. To validate the energy model as an accurate prediction of cost for database system, we use the metric called Estimation Error (ERR) to quantify the model accuracy. The metric is defined by the following formula:$EER = \frac{|ER-EE|}{ER}$. ER denotes the real values of power measured by the power meter and EE denotes the prediction values of our models using Non Linear Regression (NLR) or Random Forest Regression(RFR).

5.2. Results Analysis on TPC-H benchmark:

19. http://www.tpc.org/tpcds/

Table 3 – EER for the TPC-H benchmark queries with Postgresql DBMS SF5 - NLR Technics

Queries	*Prediction*	*Measured*	EER
1	56.37	56.66	0.5%
2	50.09	48.86	2.5%
3	51.22	51.98	1.5%
4	51.21	48.51	5.6%
5	51.17	47.36	8.0%
6	51.74	50.67	2.1%
7	52.51	51.62	1.7%
8	51.13	47.96	6.6%
9	46.57	51.48	9.5%
10	50.49	54.18	6.8%
11	50.69	52.07	2.6%
12	51.74	50.76	1.9%
13	52.81	58.97	10.4%
14	52.13	51.71	0.8%
15	49.84	53.19	6.3%
16	50.32	53.19	5.4%
17	50.86	53.67	5.2%
19	52.25	52.24	0.0%
20	51.82	50.93	1.7%
22	50.64	49.40	2.5%

When collecting system analysis data, the *Explain Analysis* command on queries 18 and 21 took a long time without giving any answers. We did not integrate them in the case of $PostgreSQL$.

Table 4 – EER for the TPC-H benchmark queries with Postgresql DBMS SF5 - RFR Technics

Queries	*Prediction*	*Measured*	EER
1	56.39	56.66	0.5%
2	54.40	48.86	11.4%
3	51.22	51.98	1.5%
4	51.45	48.51	6.1%
5	52.29	47.36	10.4%
6	51.72	50.67	2.1%
7	50.96	51.62	1.3%
8	51.52	47.96	7.4%
9	52.71	51.48	2.4%
10	51.01	54.18	5.8%
11	54.42	52.07	4.5%
12	51.33	50.76	1.1%
13	51.26	58.97	13.1%
14	51.04	51.71	1.3%
15	51.02	53.19	4.1%
16	53.96	53.19	1.5%
17	51.37	53.67	4.3%
19	50.95	52.24	2.5%
20	51.39	50.93	0.9%
22	51.50	49.40	4.3%

The tables 3 and 4 present the estimation error rates obtained using our energy cost model described in 8 on the PostgreSQL system using the Non Linear Regression(NLR) technique and the Random Forest Regression(RFR) technique. Note that in Table 3, the average of the estimation errors is 4.1% and the maximum error is 10.4% using the Non Linear Regression technique and in Table 4, the average estimate errors is 4.3% and the maximum error is 13.1% in the case of Random Forest Regression(RFR).

In tables 5 and 6, we give the estimation error rates obtained on the MonetDB system by using the two techniques. The average of the estimation errors is 2.5% and the maximum error is 13% by using the Non Linear Regression technique. By using the Random Forest technique, the average of errors is 1.7% and the maximum error is 5.3%.

The estimation error rates for Hyrise system are presented in tables 7 and 8. The average of the estimation errors is 9.0% and the maximum error is 22.14% using the Non Linear Regression technique and for Random Forest Regression, it is 7.4% and the maximum error is 17,9%.

To summarize, we present in Figure 12 the average errors obtained on the three database systems by exploiting the two learning techniques: Non Linear Regression (NLR) and Random Forest(RF).

Our finding demonstrates the ability of ours models to be exploitable. Almost, in all cases, the estimated energy is very close to the real values obtained by the measuring equipment. These finding allow us to pronounce on the quality and accuracy of our models on the TPC-H benchmark, however this is not an indication of predictive ability on other benchmark or on the other hardware configuration. As shown in the figure 12, by comparing the estimation errors obtained using the two training techniques on our models, we find that the Random Forest technique fits better than the nonlinear regression in our study. However, in the case of the PostgreSQL system, the average error of the random

Table 5 – EER for the TPC-H benchmark queries with Monet DBMS SF5 - NLR Technics

Queries	*Prediction*	*Measured*	EER
1	48.85	49.82	2.0%
2	48.10	48.30	0.4%
3	48.72	47.83	1.9%
4	48.40	48.15	0.5%
5	47.83	47.34	1.0%
6	57.71	51.50	13%
7	48.24	47.90	0.5%
8	47.98	47.96	0.01%
9	47.41	48.27	1.8%
10	50.13	50.28	0.3%
11	47.27	48.99	3.5%
12	48.62	47.16	3.1%
13	48.07	48.01	0.1%
14	48.03	46.86	2.5%
15	47.51	45.55	4.3%
16	48.80	47.71	2.3%
17	48.92	47.57	2.8%
18	48.56	47.82	1.6%
19	44.08	48.19	8.5%
20	48.40	47.3	2.3%
21	48.70	48.27	0.9%
22	48.21	47.36	1.8%

Table 6 – EER for the TPC-H benchmark queries with Monet DBMS SF5 - RFR Technics

Queries	*Prediction*	*Measured*	EER
1	48.60	49.82	2.4%
2	48.28	48.30	0.0%
3	49.15	47.83	2.8%
4	48.09	48.15	0.1%
5	47.90	47.34	1.2%
6	49.27	51.50	3.5%
7	48.00	47.90	0.0%
8	47.94	47.96	0.0%
9	47.63	48.27	1.3%
10	49.38	50.28	1.8%
11	48.04	48.99	1.9%
12	49.18	47.16	4.3%
13	48.01	48.01	0.0%
14	47.96	46.86	2.3%
15	47.94	45.55	5.3%
16	48.28	47.71	1.2%
17	49.19	47.57	3.4%
18	48.08	47.82	0.6%
19	48.94	48.19	1.6%
20	48.32	47.3	2.2%
21	48.06	48.27	0.4%
22	48.09	47.36	1.6%

Table 7 – EER for the TPC-H benchmark queries with Hyrise DBMS SF2 - NLR Technics

Queries	*Prediction*	*Measured*	EER
1	70.22	61.1	14.9%
2	46.42	43.93	5.7%
3	47.08	42.10	11.8%
4	46.95	48.40	3.0%
5	46.92	48.83	3.9%
6	48.44	39.66	22,1%
7	46.95	42.15	11.4%
8	45.42	42.90	5.9%
9	45.57	50.03	8.9%
10	48.80	46.63	4.6%
11	45.93	41.66	10.2%
12	46.92	43.90	6.9%
13	48.25	46.50	3.8%
14	47.02	41.90	12.2%
15	46.74	42.62	9.7%
16	46.53	48.85	4.8%
17	47.63	43.92	8.4%
18	45.41	52.30	13.1%
19	46.70	50.06	6.7%
20	46.93	42.30	10.9%
21	48.83	55.02	11.2%
22	46.99	43.30	8.5%

Table 8 – EER for the TPC-H benchmark queries with Hyrise DBMS SF2 - RFR Technics

Queries	*Prediction*	*Measured*	EER
1	63.84	61.1	4.5%
2	45.90	43.93	4.5%
3	47.51	42.10	12.9%
4	48.02	48.40	0.8%
5	48.98	48.83	0.3%
6	44.75	39.66	12.9%
7	47.34	42.15	12.3%
8	50.56	42.90	17.9%
9	52.37	50.03	4.7%
10	49.82	46.63	6.9%
11	45.74	41.66	9.8%
12	47.30	43.90	7.8%
13	50.63	46.50	8.9%
14	44.67	41.90	6.6%
15	44.95	42.62	5.5%
16	46.84	48.85	4.1%
17	46.96	43.92	6.9%
18	51.80	52.30	1.0%
19	46.91	50.06	6.3%
20	46.89	42.30	10.9%
21	59.44	55.02	8.0%
22	47.26	43.30	9.1%

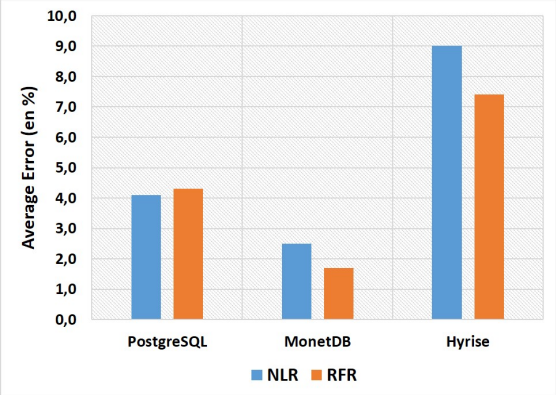


Figure 12 – Queries energy consumption average errors

forest is similar to that of the Non Linear Regression. The reasons for the performance of Random Forest can be explained by that it is easier to train as it includes a descriptor selection procedure and is not strongly dependent up on training parameters. Random Forests are immune to the problems of overfitting[30].

5.2. Results Analysis on TPCDS benchmark

To evaluate the quality of the model proposed for the Hyrise system, we used it on another benchmark (new DB diagram) in order to evaluate its portability. For this, we use the benchmark TPCDS which is more complex than the TPC-H benchmark because of its diverse scheme, its distribution data and its decision support request load. We used a dataset with a size of 3 GB. We selected 11 queries out of the 99 random queries so that the aggregation functions defined within the queries are implemented in the Hyrise system.

The results of queries execution are shown in Table 9. Note that the results of this new configuration are also of good quality with the Random Forest learning technique, as we can see in Table 9. The average error is 10% and the maximum error is below 19%. The model on the other hand with the Non Linear Regression technique was not able to predict the energy values accurately. We found an average error greater than 50%.

Energy modelization and optimization in database systems is the basic work to design energy-efficient database systems. The methods used in the literature to predict the energy

Table 9 – EER for the TPCDS benchmark queries with Hyrise DBMS SF3 - RFR Technics

Queries	*Prediction*	*Measured*	EER
1	46.67	43,06	8.4%
2	46.63	42,74	9.1%
3	46.65	41,9	11.3%
4	46.63	43,28	7.7%
5	46.63	47,66	2.2%
6	46.63	39,34	18,5%
7	46.63	40,82	14.2%
8	46.63	41,98	10.8%
9	46.63	42,08	11.1%
10	46.63	42,9	8.7%
11	46.63	43,02	8.4%

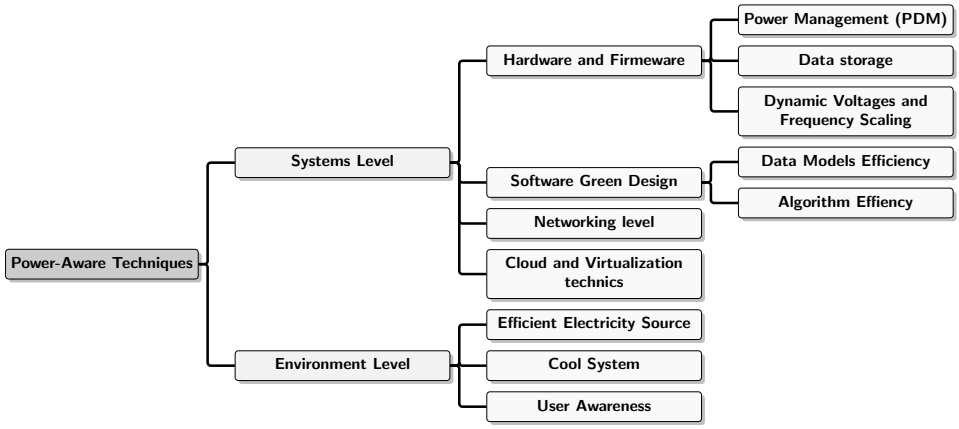


Figure 13 – Overview of techniques to improve the EE

consumption of queries in databases are roughly divided between those, which use linear regression methods, and those, which employ Non Linear Regression. Here we have used Random Forest Regression technique and we found that it has performed better than Non-linear Regression for the data set used.

6. Related Work

In this section, we review then main studies and research efforts dedicated to reduce the energy consumption when processing data. This review covers major elements of the DBMS environment. Figure 13 lists the major approaches explored to increase the EE.

6.1. Hardware level

6.1. Dynamique Power Management

DPM techniques applied at the hardware and firmware level can be broadly divided into two categories: Dynamic Component Deactivation (DCD) and Dynamic Performance Scaling (DPS). Computer components that do not have the ability to automatically adjust their performance must be completely deactivated to save power consumption, this approach is called Dynamic Component Deactivation. Dynamic Performance Scaling applies different techniques like Dynamic Voltage and Frequency Scaling (DVFS) technique to computer components in the aims to adjust dynamically their performance to the power consumption. Adopting dynamic power-management could help improve energy-efficiency. For example the studies in [25],[45],[49] use the frequency and/or voltage mechanism that allow DBMS to trade energy consumption for performance. In order to save energy, they manage the DVFS level based on throughput target and workload characteristics (e.g., I/O or CPU intensive). In the aims to save power consumption of online transaction processing (OLTP) in a multicore environment, the authors in [16] have proposed an application-aware that dynamically scale the operating frequency of processors based on response time. Application-aware use a trade-off between power consumption and response time accepted as system level agreement. The authors report a reduction of 7.6% of total power consumption on TPC-C benchmark queries. The authors of [27], experimentally studied the impact of reducing the clock frequency of the processor on the

energy efficiency of common database algorithms such as scans, aggregations, hash joins. They showed that reducing unused computing power significantly improves the energy efficiency of memory-bound database algorithms. [6] proposed the Predictive Energy Saving Online Scheduling (PESOS) algorithm in the context of Web search. Using the DVFS technique, PESOS selects the most appropriate CPU frequency to process a query. The goal of PESOS is to reduce the CPU energy consumption of a query-processing node while imposing a required tail latency on the query response times. The paper of [22] presents a technique called *POLARIS* for reducing the power consumption of transactional database systems. POLARIS directly manages processor DVFS within the DBMS and controls database transaction scheduling. Its goal is to minimize power consumption while ensuring the transactions are completed within a specified latency target. The work of [1] proposes a control-theoretic approach for DVS in multiprocessor system-on-chip pipelined architectures. It uses linear and non-linear feedback control policies to adapt frequencies using queues occupancy for incoming streams. It aims at controlling the inter-processor queue occupancy. In the linear control technique, the same feedback strategy can be iterated backward and applied in sequence to each previous stage using linear functions. Non-linear techniques use non-linear feedback functions to control the operation of systems that are either non-linear, time-varying, or a combination of both.

6.1. Microarchitectural Power Saving

Microarchitectural techniques allowing energy saving in specific components (eg main memory, cache memories, etc.) are widely known and studied in the literature. the authors in [3] used DRAM frequency scaling and power-down modes to improve power consumption without sacrifying performance under both transactional and analytical workloads. They start by identifying the hardware features that help reduce DRAM power consumption, and then they perform an experimental analysis of the impact of those features on power and performance. The work of [15] proposes a hybrid memory architecture consisting of both Dynamic Random Access Memory (DRAM) and Non-Volatile Memory (NVM) to reduce database energy consumption, through an application-level data management policy that decides to place data on DRAM or NVM. In a recent work, the authors of [21] proposed the Energy-Control Loop (*ECL*) as a DBMS-integrated approach for adaptive energy-control on in-memory database systems that obeys a query latency limit as a soft constraint and actively optimizes energy efficiency and performance of the DBMS. The ECL relies on adaptive workload-dependent energy profiles that are continuously maintained at runtime. In[51] authors present survey on cache power/energy-consumption reduction techniques. They have surveyed various cache tuning techniques based on two categories, offline static cache tuning and online dynamic cache tuning, ranging from hardware support to cache tuning techniques. In[7] the effect of power and performance are analyzed by reducing cache capacity. Reducing cache size in some applications give significant energy saving in order of 3.17% and 24.16% in dynamic and static energy respectively while degrade the performance by increasing cache misses.

6.1. Telecommunication Network Power Management

Telecommunication networks constitute a major sector of ICT and they undergo a tremendous growth. Energy is mostly consumed in network transmission and switching equipment such as routers. In the literature, we observe that the focus of many research is on control and optimization strategies for computer network equipment enabling energy saving, by adapting network capacities and computing resources to the current traffic load

and demands, while ensuring end-to-end quality of service (QoS)[4]. The approaches to re-duce energy consumption in networks is divided into four categories: (i) selectively turning off network elements,(ii) energy-efficient network design, (iii) energy-efficient IPpacket forwarding, and (iv) green routing. by Zhang and al. [52]. In [20], the authors propose that WLANs be re-designed so that the power consumed by the WLAN scales with its offered load. Such strategies allow powering on or off WLAN access points dynamically, based on the volume and location of user demand. To achieve a power saving router, Authors in [50] proposed to follow two approaches; power efficient designing and power saving designing. The former is an approach to create a high performance router at low power consumption and the latter is an approach to save wasted power. For power efficient designing, they have developed technologies for integrating the ASICs/FPGAs and memories of routers. for power saving design, the authors worked on static performance control, which allows turning off unused ports and modules. Furthermore, they proposed a technology that lowers the frequency of lightly utilized modules to save the wasted power. Running under the low frequency mode, the power efficiency improved by 10-20%.

6.2. Software Level

6.2. OS-Based Power Management

The power-efficient resource management at the OS level plays an important role in energy efficiency and it completes the effort provided at the hardware level. The authors in [29] have developed a real-time power manager in the kernel for the Linux OS called the On-Demand Governor (LGD). The goal of the on-demand governor is to keep the performance loss due to reduced frequency to the minimum. [42] discusses the behavior of the current task management subsystems (scheduler and load balancer) in the Linux kernel on a multi-core SMP system and also it covers its effectiveness in saving energy consumption under several situations (e.g. idle, moderate load). It then describes several techniques such as timer migration, task wakeup biasing and related heuristics for reducing energy consumption. Rajkumar et al. [34] have proposed several algorithms for application of DVFS in real-time systems and have implemented a prototype as a modified Linux kernel Linux/Resource Kernel (Linux/RK). The objective is to minimize the energy consumption, while maintaining the performance isolation between applications.

6.2. Energy-Effiency in DBMS query processing

In this Approaches, most important studies concern (i) the definition of cost models to predict the energy and (ii) the proposition of cost-driven techniques for reducing energy. In [46, 47] the authors build a static power profile for each basic database operation using a simple linear regression technique. In [23], the authors use pipeline-based modeling to the sources of peak power consumption for a query and to recommend plans with low peak power. the work of [24] proposes a framework for energy-aware database query processing. It extends 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. In [37], the authors proposed cost models to predict the power consumption of single running queries. the proposed model relies on pipeline segmenting of the query. The model is built based on the CPU, I/O cost of each pipeline and measured energy in an offline task, using non-linear regression technique (polynomial regression). The authors in [14] analyze the effect of the three main cache structures (Database Buffer Cache, Dictionary Cache, and Library Cache). Based on this, they have taken the cost of memory into account in their linear model cost dedicated to sequential query processing

mode. In [10][9], the authors confront the sequential and parallel execution modes and study their impact on EE. They extended the sequential cost model proposed in [37] in order to predict the energy consumption of queries when they are executed in parallel.

6.3. Environmental management and conventional rules

In response to the over energy consumption in datastores and the need to reduce the related environmental, economic and energy supply security impacts, the governments, and corporate sector impose regulations and acts. The European Commission (EC) created a Code of Conduct for Energy Efficiency [20] in Data Centers in 2008 with the aim of improving the energy efficiency. Like EC, the U.S. Department of Energy has the ENERGY STAR [21] program, which offers energy efficiency guidelines for all types of buildings including data centers. In addition to the Code of Conduct for Energy Efficiency, there are several energy-efficiency standards imposed to control the manufacture and use of equipment.

7. Conclusion

In this paper, we have presented three energy cost model, each for one of our database management systems selected. Our proposals center around four main steps for the construction of an energy model: (1) establishment of a deep audit that allows understanding the query processor functioning, (2) identification of relevant energy-sensitive parameters belonging to hardware and software components, (3) elaboration of mathematical cost models estimating consumed energy when executing a query on a target DBMS and (4) setting of values of the energy-sensitive parameters using a Non Linear Regression technique and Random Forest technique. This procedure highlighted the importance of audit of DBMSs by understanding their functioning, before proposing any energy modeling. For that purpose, we perform thorough experiments on the power consumption patterns of various workloads generated from the TPC-H and TPCDS benchmarks on three DBMS system PostgreSQL, MonetDB and Hyrise to validate our proposing.

20. https://ec.europa.eu/jrc/en/energy-efficiency/code-conduct/datacentres
21. https://www.energystar.gov/about

References

[1] Andrea Alimonda, Andrea Acquaviva, Salvatore Carta, and Alessandro Pisano. A control theoretic approach to run-time energy optimization of pipelined processing in mpsocs. In *Proceedings of the Conference on Design, Automation and Test in Europe*, pages 876–877, 2006.

[2] Dave Anderson, Jim Dykes, and Erik Riedel. More than an interface scsi vs ata. USENIX, 2003.

[3] Raja Appuswamy, Matthaios Olma, and Anastasia Ailamaki. Scaling the memory power wall with dram-aware data management. In *Proceedings of the 11th International Workshop on Data Management on New Hardware*, page 3. ACM, 2015.

[4] Kashif Bilal, Saif Ur Rehman Malik, Osman Khalid, Abdul Hameed, Enrique Alvarez, Vidura Wijaysekara, Rizwana Irfan, Sarjan Shrestha, Debjyoti Dwivedy, Mazhar Ali, et al. A taxonomy and survey on green data center networks. *Future Generation Computer Systems*, 36:189–208, 2014.

[5] Peter A. Boncz, Martin L. Kersten, and Stefan Manegold. Breaking the memory wall in monetdb. *Commun. ACM*, 51(12):77–85, December 2008.

[6] M. Catena and N. Tonellotto. Energy-efficient query processing in web search engines. *IEEE Transactions on Knowledge and Data Engineering*, 29(7):1412–1425, July 2017.

[7] Shounak Chakraborty, Dipika Deb, Dhantu Buragohain, and Hemangee Kapoor. Cache capacity and its effects on power consumption for tiled chip multi-processors. pages 1–6, 02 2014.

[8] M. Dayarathna, Y. Wen, and R. Fan. Data center energy consumption modeling: A survey. *IEEE Communications Surveys Tutorials*, 18(1):732–794, Firstquarter 2016.

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

[10] Simon Pierre Dembele, Amine Roukh, and Ladjel Bellatreche. Vers des optimiseurs verts de requêtes en mode parallèle. In *EDA*, 2018.

[11] Ramez Elmasri and Shamkant Navathe. *Fundamentals of Database Systems*. Addison-Wesley Publishing Company, USA, 6th edition, 2010.

[12] Martin Grund, Jens Krüger, Hasso Plattner, Alexander Zeier, Philippe Cudre-Mauroux, and Samuel Madden. Hyrise: A main memory hybrid storage engine. *Proc. VLDB Endow.*, 4(2), 2010.

[13] Binglei Guo, Jiong Yu, Bin Liao, Dexian Yang, and Liang Lu. 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.

[14] Binglei Guo, Jiong Yu, Bin Liao, Dexian Yang, and Liang Lu. 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.

[15] Ahmad Hassan, Hans Vandierendonck, and Dimitrios S Nikolopoulos. Energy-efficient in-memory data stores on hybrid memory hierarchies. In *Proceedings of the 11th International Workshop on Data Management on New Hardware*, page 1. ACM, 2015.

[16] Yuto Hayamizu, Kazuo Goda, Miyuki Nakano, and Masaru Kitsuregawa. Application-aware power saving for online transaction processing using dynamic voltage and frequency scaling in a multicore environment. Springer Berlin Heidelberg, 2011.

[17] Ali Hurson and Hamid Azad. *Energy Efficiency in Data Centers and Clouds*. Academic Press, 2016.

[18] Stratos Idreos, Fabian Groffen, Niels Nes, Stefan Manegold, K. Sjoerd Mullender, and Martin L. Kersten. Monetdb: Two decades of research in column-oriented database architectures. *IEEE Data Eng. Bull.*, 35(1):40–45, 2012.

[19] Milena G. Ivanova, Martin L. Kersten, Niels J. Nes, and Romulo A.P. Gonçalves. An architecture for recycling intermediates in a column-store. In *ACM SIGMOD*, pages 309–320, 2009.

[20] Amit P. Jardosh, Gianluca Iannaccone, Konstantina Papagiannaki, and Bapi Vinnakota. Towards an energy-star WLAN infrastructure. pages 85–90. IEEE Computer Society, 2007.

[21] Thomas Kissinger, Dirk Habich, and Wolfgang Lehner. Adaptive energy-control for in-memory database systems. In *Proceedings of the 2018 International Conference on Management of Data*, pages 351–364. ACM, 2018.

[22] Mustafa Korkmaz, Martin Karsten, Kenneth Salem, and Semih Salihoglu. Workload-aware cpu performance scaling for transactional database systems. In *Proceedings of the 2018 International Conference on Management of Data*, pages 291–306. ACM, 2018.

[23] Mayuresh Kunjir, Puneet K Birwa, and Jayant R Haritsa. Peak power plays in database engines. In *EDBT*, pages 444–455. ACM, 2012.

[24] Willis Lang, Ramakrishnan Kandhan, and Jignesh M Patel. Rethinking query processing for energy efficiency: Slowing down to win the race. *IEEE Data Eng. Bull.*, 34(1):12–23, 2011.

[25] Willis Lang and Jignesh Patel. Towards eco-friendly database management systems. *arXiv preprint arXiv:0909.1767*, 2009.

[26] Sparsh Mittal. A survey of techniques for improving energy efficiency in embedded computing systems. *CoRR*, abs/1401.0765, 2014.

[27] Stefan Noll, Henning Funke, and Jens Teubner. Energy efficiency in main-memory databases. *Datenbank-Spektrum*, 17(3):223–232, 2017.

[28] Abdeen Omer. Energy use and environmental impacts: A general review. *Journal of Renewable and Sustainable Energy*, 1, 09 2009.

[29] Venkatesh Pallipadi and Alexey Starikovskiy. The ondemand governor. In *Proceedings of the Linux Symposium*, volume 2, pages 215–230. sn, 2006.

[30] David S. Palmer, Noel M. O'Boyle, Robert C. Glen, and John B. O. Mitchell. Random forest models to predict aqueous solubility. *J. Chem. Inf. Model.*, 47(1):150–158, 2007.

[31] Meikel Poess and Raghunath Othayoth Nambiar. Energy cost, the key challenge of today's data centers: a power consumption analysis of tpc-c results. *PVLDB*, 1(2):1229–1240, 2008.

[32] PostgreSQLfr. PostgreSQL Documentation the postgresql global development group. `https://www.postgresql.org/docs/11/intro-whatis.html`. Accessed: 2019-09-26.

[33] Mark Raasveldt and Hannes Mühleisen. Monetdblite: An embedded analytical database. *CoRR*, abs/1805.08520, 2018.

[34] Raj Rajkumar, Kanaka Juvva, Anastasio Molano, and Shuichi Oikawa. Readings in multimedia computing and networking. pages 476–490. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2001.

[35] J.O. Rawlings. *Applied regression analysis: a research tool*. Wadsworth & Brooks/Cole statistics/probability series. Wadsworth & Brooks/Cole Advanced Books & Software, 1988.

[36] Amine Roukh and Ladjel Bellatreche. Eco-processing of olap complex queries. In *International Conference on Big Data Analytics and Knowledge Discovery*. Springer, 2015.

[37] Amine Roukh and Ladjel Bellatreche. Eco-processing of olap complex queries. In *International Conference on Big Data Analytics and Knowledge Discovery*, pages 229–242. Springer, 2015.

[38] Amine Roukh, Ladjel Bellatreche, Ahcène Boukorca, and Selma Bouarar. Eco-physic: Eco-physical design initiative for very large databases. *Information Systems*, 68:44–62, 2017.

[39] Amine Roukh, Ladjel Bellatreche, and Carlos Ordonez. Enerquery: Energy-aware query processing. In *Proceedings of the 25th ACM International Conference on Information and Knowledge Management*, 10 2016.

[40] Daniel Schall, Volker Hudlet, and Theo Härder. Enhancing energy efficiency of database applications using ssds. In *Proceedings of the Third C* Conference on Computer Science and Software Engineering*, pages 1–9. ACM, 2010.

[41] NASA SCIENCE. Anatomy of an electromagnetic wave, December 2018.

[42] Vaidyanathan Srinivasan, Gautham R Shenoy, Srivatsa Vaddagiri, and Dipankar Sarma. Energy-aware task and interrupt management in linux. 2009.

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

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

[45] Yi-Cheng Tu, Xiaorui Wang, Bo Zeng, and Zichen Xu. A system for energy-efficient data management. *ACM SIGMOD Record*, 43(1):21–26, 2014.

[46] Zichen Xu, Yi-Cheng Tu, and Xiaorui Wang. Exploring power-performance tradeoffs in database systems. In *ICDE*, pages 485–496, 2010.

[47] Zichen Xu, Yi-Cheng Tu, and Xiaorui Wang. Dynamic energy estimation of query plans in database systems. In *ICDCS*, pages 83–92. IEEE, 2013.

[48] Zichen Xu, Yi-Cheng Tu, and Xiaorui Wang. Dynamic energy estimation of query plans in database systems. In *2013 IEEE 33rd International Conference on Distributed Computing Systems*, pages 83–92. IEEE, 2013.

[49] Zichen Xu, Xiaorui Wang, and Yi-Cheng Tu. Power-aware throughput control for database management systems. In *ICAC*, pages 315–324, 2013.

[50] M. Yamada, T. Yazaki, N. Matsuyama, and T. Hayashi. Power efficient approach and performance control for routers. In *2009 IEEE International Conference on Communications Workshops*, pages 1–5, 2009.

[51] Wei Zang and Ann Gordon-Ross. A survey on cache tuning from a power/energy perspective. *ACM Comput. Surv.*, 45(3):32:1–32:49, 2013.

[52] Yi Zhang, Pulak Chowdhury, Massimo Tornatore, and Biswanath Mukherjee. Energy efficiency in telecom optical networks. *IEEE Commun. Surv. Tutorials*, 12(4):441–458, 2010.