Eco-Physic: Eco-Physical Design Initiative for Very Large Databases[☆]

Amine Roukh[a], Ladjel Bellatreche[b,*], Ahcene Boukorca[b], Selma Bouarar[b]

[a] *University of Mostaganem, Mostaganem, Algeria*
[b] *LIAS/ISAE-ENSMA, University of Poitiers, Poitiers, France*

Abstract

In the Big Data Era, the management of energy consumption by servers and data centers has become a challenging issue for companies, institutions, and countries. In data-centric applications, Database Management Systems are one of the major energy consumers when executing complex queries involving very large databases. Several initiatives have been proposed to deal with this issue, covering both the hardware and software dimensions. They can be classified in two main driven-hypothesis approaches assuming either that **(a)** the database is *already deployed* on a given platform, or **(b)** it is *not yet deployed*. In this study, we focus on the first set of initiatives with a particular interest in physical design, where optimization structures (e.g., indexes, materialized views) are selected to satisfy a given set of non-functional requirements such as query performance for a given workload. In this paper, we first propose an initiative, called *Eco-Physic*, which integrates the energy dimension into the physical design when selecting materialized views, one of the redundant optimization structures. Secondly, we provide a multi-objective formalization of the materialized view selection problem, considering two non-functional requirements: query performance and energy consumption while executing a given workload. Thirdly, an evolutionary algorithm is developed to solve the problem. This algorithm differs from the existing ones by being interactive, so that database administrators can adjust some energy sensitive parameters at the final stage of the algorithm execution according to their specifications. Finally, intensive experiments are conducted using our mathematical cost model and a real device for energy measurements. Results underscore the value of our approach as an effective way to save energy while optimizing queries through materialized views structures.

Keywords: Physical Design, Energy Efficiency, Power Management
2010 MSC: 00-01, 99-00

1. Introduction

Global energy consumption has not ceased to increase. Three key factors are behind this steady rise: **(1)** the ubiquity of the Worldwide Internet and the pervasiveness of mobile handsets, netbooks and laptop PCs. According to the Natural Resources Defense Council report, if the World-Wide Internet was a country, it would be the 12th-largest consumer of electricity in the world, between Spain and Italy [2]. **(2)** The high demand of users and decision makers to efficiently store, access and manage huge volumes of data emanating from sensors, social networks, experimentations, simulations, etc. of both everyday and analytical applications. The management of these data is usually performed by Database Management Systems (DBMSs). **(3)** The extensive usage of data centers as a *deployment platform* for such applications motivates companies owning these data to invest in expanding their use of such centers. Recently, Google announced details about the expansion of its data center by 500,000 square feet to provide a new mega data center over 1.3m square feet[1].

These three factors significantly increase the consumed energy and hence the carbon footprint of data centers. The augmentation of the energy consumption is mainly caused by the high power requirements of Information Technology (IT) equipment and cooling infrastructures. In a traditional data center, for every kilowatt of power used to drive a server, another kilowatt is needed to cool it [3, 4]. Industry experts, such as the SMARTer 2020, report that the global data center emissions will grow 7 percent every year through 2020 [5]. DBMSs are today one of the major challenges toward energy efficiency [6]. Figure 1 depicts the energy consumption distribution per component in a data center. It reveals that half of the total energy is particularly resulted from *IT* equipment (CPU, memory, disk, etc.), which is a very significant part. However, identifying the power consumption behavior of these main components is the first step toward improving their energy efficiency.

Faced with this situation, energy saving becomes a

[☆] This work is an extension of our paper [1]
[*] Corresponding author.
Email addresses: `roukh.amine@univ-mosta.dz` (Amine Roukh), `ladjel.bellatreche@ensma.fr` (Ladjel Bellatreche), `ahcene.boukorca@ensma.fr` (Ahcene Boukorca), `selma.bouarar@ensma.fr` (Selma Bouarar)

[1] `http://www.datacenterdynamics.com/design-build/`

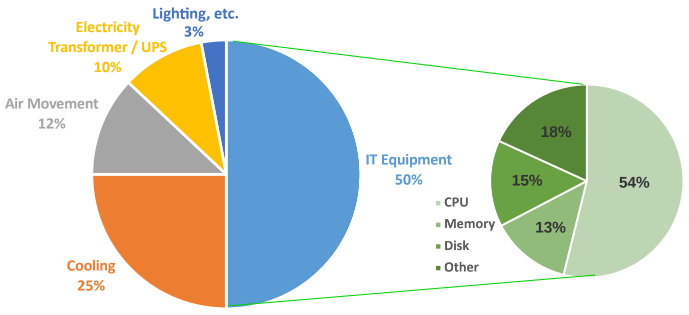


Figure 1: A breakdown of energy consumption for a different component in a data center [3, 4].

political issue for countries that are currently enacting laws on energy regulation. For instance, in the USA, the National Action Plan for Energy Efficiency – a public-private initiative – aims at creating a sustainable, aggressive national commitment to energy efficiency through the collaborative effort of gas and electricity utilities, utility regulators, as well as other partner organizations[2]. In France, similar schemes have been initiated by ADEME – the French Environment and Energy Management Agency[3]. Several, but not enough, actions to regulate and to reduce the use of energy have been taken in many domains such as housing (green building), smart cities, transportation, etc. Recently, a conference of United Nations Climate Change (COP21), was held in Paris (France) from November 30 to December 12, 2015[4], what testifies to the importance of the issue. The COP21 event showed the willingness of countries (over 145 foreign Heads of State and Government attended the conference), companies, individuals, governmental and non-governmental organizations, etc. to save the planet.

The Database Community has spared no effort to propose initiatives in this sense. Traditionally, designing a database application mainly focused on improving low-latency query processing to satisfy the needs of end users (e.g., decision makers). This has been leveraged by integrating the energy dimension into the deployment and exploitation phases of the databases [7]. The *Claremont report* on database research [8] states the importance of "*designing power-aware DBMSs that limit energy costs without sacrificing scalability*". This is also echoed in the more recent *Beckman report* on databases, which considers "*energy constrained processing as a challenging issue in Big Data*" [9]. Consequently, the energy management of database systems has emerged as an important research topic over the past few years.

The proposed research efforts include both the hardware and software dimensions. Two main hypotheses have been made for these initiatives: (i) already deployed database applications ($\mathcal{ADD}$) and (ii) not yet deployed database applications ($\mathcal{NYD}$). In the first hypothesis, we consider that the database application is already designed and deployed on a given platform. Several solutions have been proposed covering software and hardware aspects. From a software perspective, a couple of studies have mainly focused on energy-aware query processing [10, 11, 12]. These studies follow two main directions: (i) the development of mathematical cost models predicting energy consumption when executing a query. Cost models are the main component of query optimizers used to estimate the execution costs of different query plans. (ii) The proposal of intuitive and advanced optimization techniques guided by these cost models to reduce the energy. These efforts are quite insufficient for advanced databases handling an enormous amount of data and complex queries running in complex schemas involving several tables. Thus, the proposed approaches require further attention. From hardware a perspective, naive techniques have been proposed like disabling electronic components during periods of inactivity, as well as advanced ones such as dynamically changing the performance of hardware components for better energy efficiency, e.g., Dynamic Voltage and Frequency Scaling (DVFS) [13]. Regarding initiatives with the $\mathcal{NYD}$ hypothesis, a majority of the studies cover the hardware part. In this paper, we only concentrate on an initiative regarding software with the $\mathcal{ADD}$ hypothesis.

Our initiative concerns the physical design phase. Recently, several well-known researchers from academia (Stavros Harizopoulos *et al.* [14], MIT) and from industry (Goetz Graefe [15], HP) have recommended revisiting the physical design by involving energy dimension. It should be noticed that the physical configuration of the database on the storage media is specified at the physical design phase. Several optimization techniques (structures) such as materialized views, indexes, and partitioning to improve the performance of query execution are selected at this stage [16]. The importance of physical design has been amplified as query optimizers have become sophisticated enough to cope with complex decision support applications [17]. Usually, the selection of a physical structure is performed by the means of algorithms using cost models estimating the costs of one or multiple objective functions [18]. Note that these cost models use parameters related to database schemas, hardware, query workload, and constraints (storage costs, maintenance costs, etc.). Figure 2 illustrates the database physical design formulation. The selection of these optimization structures strongly impacts the database, since they become a part of it. As a consequence, they may contribute positively or negatively to increasing energy consumption. Therefore, the exploration of energy-aware physical design is now crucial to prevent energy waste when selecting and maintaining these optimization structures. To monitor this energy, we recommend its integration into the physical design process.

In this paper, we propose an initiative called *Eco-Physic*

[2]`www.epa.gov/cleanenergy/energy-programs/suca/resources.html`
[3]`www.ademe.fr/en/about-ademe`
[4]`http://www.cop21.gouv.fr/en/`

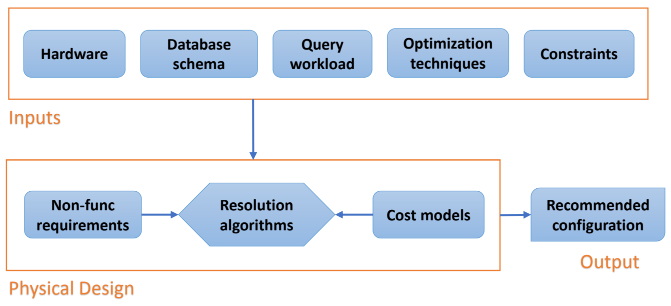


Figure 2: Database physical design formulation.

that attempts to integrate the energy into the physical design process by considering *materialized views* which is a redundant optimization structure of queries where results are stored and maintained in order to globally optimize query workload [19].

Our initiative to reduce energy dissipation when selecting physical structures is to consider energy as one of the objectives to be optimized. Therefore, the process of materialized view selection becomes a multi-objective optimization problem that may involve several potentially conflicting objectives (query performance and energy saving). It has a Pareto set [20] which represents the set of the optimal solutions in both objectives. The multi-objective evolutionary algorithms (MOEAs) are a suitable approach to approximate the *Pareto optimal set* in a single run [20] and have shown their efficiency for several problems (e.g., salesman problem). Since it is not feasible to have a unique solution which simultaneously optimizes both objectives, the use of an algorithm that offers Database Administrators (DBAs) numerous alternative solutions restricting on or near the Pareto optimal front is of great practical value [21]. For this purpose, we use elitist *Non-dominated Sorting in Genetic Algorithms* (NSGA-II) [21]. Moreover, we offer DBAs the possibility to favor one objective function over another. To satisfy this requirement, we apply the weighted sum aggregate method that assigns a weight to each objective function.

1.1. Our Contributions

The major technical contributions of this paper can be summarized as follows:

- a detailed survey of existing approaches covering both hardware and software dimensions and using the $\mathcal{ADD}$ hypothesis for energy-aware database systems;
- a deep classification of current solutions for selecting materialized views according to three criteria: (i) the use of non-functional requirements ($\mathcal{NFR}$), (ii) the use of constraints and (iii) the use of solving algorithms;
- a formalization of the view selection problem ($\mathcal{VSP}$) as a multi-objective optimization problem including query performance and energy consumption objectives, and subject to storage and maintenance cost constraints;
- an in-depth discussion on a general approach for developing cost models used to assess the execution time and energy consumption for a given query based on its I/O and CPU costs;
- the proposal and implementation of evolutionary algorithms to solve $\mathcal{VSP}$, which provide a variety of solutions offering multiple trade-offs between the fixed objectives;
- intensive experiments using real and simulation tools to study the effectiveness of our approaches. Energy savings up to 84% and active power savings up to 38% can be achieved thanks to our proposal.

1.2. Paper Organization

The remainder of this paper is organized as follows. We present a detailed classification of related works that aim at reducing the energy of database systems in Section 2. In Section 3, we discuss our energy-efficient conception initiative that concerns physical design phase of the database life-cycle. The section gives a more detailed formulation of the view selection problem based on global query plans, as well as a supporting example to illustrate performance and power relationships and behaviors. The section also presents the cost models used in our study to estimate performance and power of SQL queries. Section 4 describes our methodology that relies on evolutionary algorithms to solve a constrained multi-objective optimization problem. The algorithm and their implementations are explained in details, including selection, crossover, mutation steps and the weighting method. Section 5 presents and interprets experimental results based on several scenarios of materializing views with various parameters and data sizes. Finally, Section 6 concludes the paper by summarizing the main results and suggesting future work.

2. Related Work

As we said before, energy-efficiency optimizations in database systems have become an active research topic. There has been a plethora of studies and initiatives by the research community. While reviewing the literature of works considering the $\mathcal{ADD}$ hypothesis, we primarily distinguish two main approaches: *Hardware-Driven Approaches* ($\mathcal{HDA}$) and *Software-Driven Approaches ($\mathcal{SDA}$)*. In this section, we propose a survey and a classification of these approaches, as depicted in Figure 3.

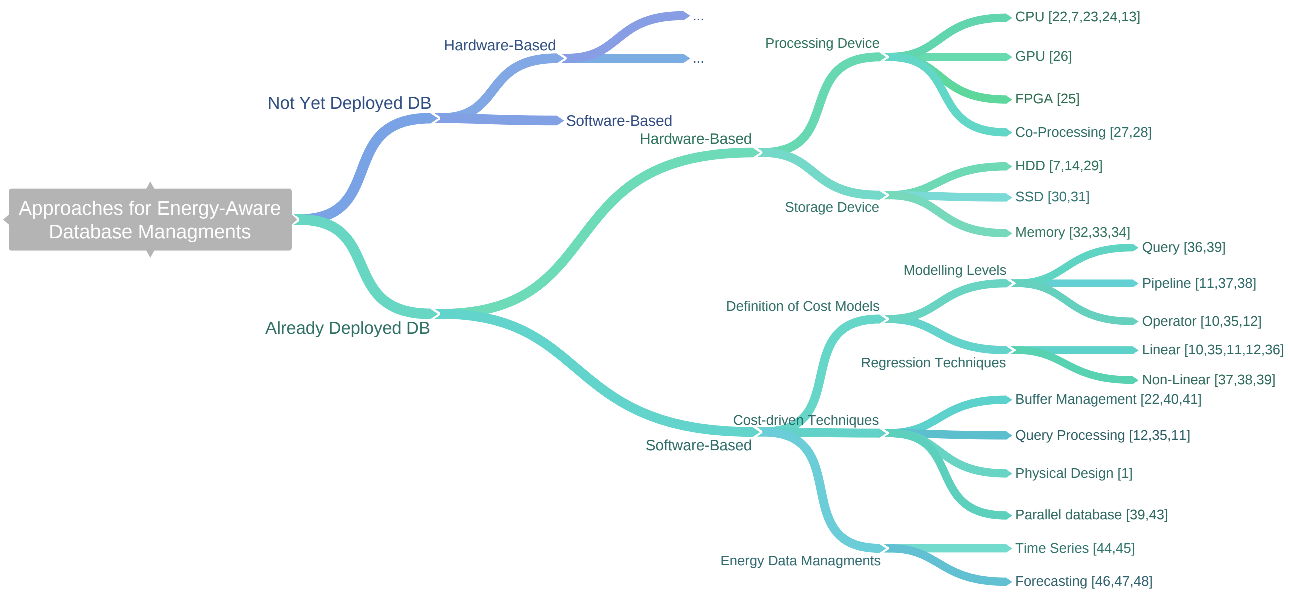


Figure 3: Classification of database energy-efficiency methods.

2.1. *HDA*

The hardware approaches use naive solutions like disabling electronic components during periods of inactivity, and advanced ones such as dynamically changing the performance of hardware components for better energy efficiency, which can be illustrated by DVFS technique offered by today's CPU [13]. Besides, techniques applied at the hardware level can be broadly divided into two categories: (i) processing device and (ii) storage management.

2.1.1. *Processing Device*

Techniques belonging to this category use novel approaches to slow down the energy consumption of computing devices. Lang *et al.* proposed a PVC (Processor Voltage/Frequency Control) mechanism to trade energy consumption for performance [22]. It aims at executing instructions at a lower processor voltage and frequency by leveraging the ability of modern processors. A similar technique is used in [7], the authors report 51.3% of energy savings. The proposed technique dynamically adjusts the DVFS level of the processor according to the performance of the DBMS or the query plan actually chosen. Xu *et al.* [23] also use DVFS technique with a feedback control mechanism for database workloads. 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 another study, a co-engineering between Oracle and Intel attempting to decrease operating costs and effectively meet green computing goals has given rise to a new version of *Oracle Exadata Database Machine* [24]. The solution stack dynamically puts Intel Xeon processor and memory resources into the lowest available power state suitable to meet the demands of the workload. However, the aforementioned techniques suffer from an overhead cost of the transitions to low-power states, which usually lead to additional power consumption and delays caused by components reinitialization [13].

To accelerate data-intensive applications, Field-Programmable Gate Array (FPGAs) are used instead of classic CPUs, mainly because they allow building custom hardware at relatively low development costs. Moreover, it has already been shown in [25] that FPGA-based solutions have the potential to improve performance and at the same time reduce energy consumption. The authors report that using their proposal, the system consumes only 216 *joules*, whereas traditional techniques consume 3888 *joules* for a certain query.

Graphics Processing Units (GPUs) have been considered as an excellent alternative to CPUs for their high-performance, high-throughput computing and also their energy saving. A study reported that using a GPU is more energy efficient when the performance improvement is above a certain bound, compared to a CPU-only solution [26]. CPU-GPUs co-processing is another attractive trend in database technology, that consists in exploiting the hardware advances as well as optimized algorithm design to improve the query processing performance and energy-efficiency. The comparison results of discrete CPU-GPUs and coupled CPU-GPUs show that the average power consumption of the discrete architecture is between 36% and 44% higher than those of the coupled architecture [27].

On the other hand, recent studies such as [28] claim that using Smart Solid State Disks (SSSDs) to process data can be more advantageous from both performance and energy consumption perspectives. SSSDs are flash storage devices that incorporate memory and computing inside the SSD device, which could be employed to run general userdefined programs. The authors in [28] improve the energy efficiency of query processing by reducing its running time and/or by running processing on the low power processor that comes with the SSSDs. The early results of running a subset of SQL queries (simple selection and aggregation queries) on SSSD deliver up to a 3x reduction in energy consumption. SSSD have a lot of potential, but the computation capabilities of the devices embedded into SSSD are too limited to manage the huge volume of data.

2.1.2. Storage Management

Processing devices have been considered as the main contributors to the server power consumption. That said, secondary storage and memory power consumption is also drawing increasing attention of researchers because of their significant capacity growth over the last few years. Storage management approaches try to find a good level of load consolidation, improving caching and prefetching techniques and optimizing power modes in disk arrays to minimize the energy dissipation [7]. In [14], the authors provide experiments to show that choices made by database systems can improve energy efficiency. Specifically, repartitioning the database across 66 disks offers a 14% increase in efficiency against 45% drop in performance. The authors in [7] proposed an integrated optimization model for data migration and disk state adjustment, in which they dynamically place records into a fragment according to the access frequency of data records. The reported results show that about 50% energy can be saved using the proposed techniques. In the same direction, the work of [29] proposes a dynamic power management model that makes real-time decisions about switching the disks to low-power modes. The model is integrated into the DBMS and used to achieve the optimal trade-off between power consumption and query response time. The report shows 60% of energy savings.

Using SSDs as a storage device by databases to improve energy-efficiency is another direction that has been investigated by recent works. In [30], the authors explored the performance behavior and the related energy consumption of SSDs under typical access patterns for I/O database-intensive applications. They further show that the SSD technology provides a new level of performance and energy efficiency, and suggest exploiting their new characteristics by database servers. Similarly, the work of [31] analyzes and evaluates the data processing performance and energy efficiency of SSDs and storage system. Like its predecessor, the authors show improvement in both performance and energy-efficiency.

Memory is another significant consumer of energy in computing systems, especially with the emerging use of in-memory column stores and key-value stores by the database systems. Many recent research papers start studying the impact of the memory device on energy consumption. The investigation made in [32] highlights that the growing use of main memory databases will soon emerge memory as the dominant power consumer instead of CPU. As with the previous components, the main memory also has different power states and frequency to be changed according to memory utilization. Based on that, the authors used DRAM frequency scaling and power-down modes to improve power consumption without sacrificing performance. In [33], the authors report a preliminary result on a rank-aware memory allocation technique to save database systems power. The technique allows the DBMS to move unneeded memory ranks to low-power states and hence reduces the overall memory power consumption. The evaluation indicates that the proposed method attains 40% decrease of power when the database size is smaller than the amount of memory already provisioned on the server. The work of [34] 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. They first analyze the execution of applications and collect memory access statistics for individual objects. These latter are evaluated using an analytical model of performance and energy to identify the most suitable memory technology to store that data. The reported results indicate that using hybrid memory can reduce memory system energy by about 80%.

The beforementioned studies have indeed made significant contributions to energy conservation. However, $\mathcal{HDA}$ are not straightforward to apply and have several drawbacks especially in the database field. First, they require using new hardware, as new technologies such as DVFS which are not available on all current processors or main memories. Second, the emerging SSDs are very expensive and have a lifetime issue. And last but not least, there exist various brands/versions for each hardware component, making these methods impractical due to the resulting portability issue. Therefore, $\mathcal{SDA}$ are much preferred in the context of databases, as they are independent of the underlying hardware. Additionally, the DBMS has detailed knowledge of internal system items, such as the query workload, execution plans, database statistics, etc. [33].

2.2. $\mathcal{SDA}$

The hardware approaches are only part of the solution for reducing power values; the software approaches are the other key part in optimization mechanisms for energy efficiency. At this level, the vision papers on energy [15, 14] recommend revisiting query optimization, scheduling algorithms, physical database design, and database update techniques by considering energy. Most important studies concern two main aspects: (i) the definition of cost models to predict energy and (ii) the proposition of cost-driven techniques for reducing energy. This situation is quite similar to that experienced in traditional performance-oriented design approaches, where most of the query optimizers of open source[5] and commercial DBMS[6] use cost-based driven approaches (CBA). In CBA, a cost-based optimizer estimates the cost of each possible execution plan by applying heuristic formulas using a set of statistics concerning database (sizes of tables, indexes, length tuples, selectivity factors of join and selection predicates, sizes of intermediate results, etc.) and hardware (size of buffer, page size, etc.).

2.2.1. Definition of Cost Models

In [10, 35], the authors propose a power-oriented extension of PostgreSQL's cost model to predict query power consumption. A static power profile for each basic database operation in query processing is defined. The power cost of a plan can be calculated from the basic SQL operations, like CPU power cost to access a tuple, power cost for reading/writing one page, and so on, via different access methods and join operations using a simple linear regression technique. The authors adapt their static model to dynamic workloads using a feedback control mechanism to periodically update model parameters using real-time energy measurements. In [11], a deep research in modeling the peak power[7] of query operations is given. A pipeline-based model of query execution plans was developed to identify the sources of peak power consumption for a query and to recommend plans with low peak power. For each of these pipelines, a mathematical function is applied, which takes as input the rates and sizes of data flowing through the pipeline operators. As outputs, an estimation of the peak power consumption is given. The authors used piecewise regression technique to build their cost model. In the same direction, the work of [12] 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. The study of [36] attempts to model energy and peak power of simple selection queries on single relations using linear regression. In [37], we proposed cost models to predict the power consumption of single running queries. In contrast to previous studies on power modeling which based their models on SQL operations level, our model relies on pipeline segmenting of the query. A pipeline is defined as the concurrent execution of a contiguous sequence of operators. Based on experiments study, we found that the power consumption behavior of queries follows the pipeline-based execution strategy implemented by the DBMS (i.e., the power value changes when current running pipeline changes). For each pipeline, we predict its power based on its I/O and CPU costs, using non-linear regression technique (polynomial regression). In [38], we studied the case of predicting power consumption of multiple concurrently running queries, since in practice, queries are executing in a concurrent way and exhausting the system resources. Building models that predict power in this scenario is useful for many database managements tasks, including admission control, query scheduling and execution control with energy efficiency as a first-class performance goal. The proposed model takes the number of queries that are concurrently running and segments them into a set of pipelines, and for each pipeline, we use the same techniques as in [37] to predict their power. The model produces results with small prediction errors. While the cited works consider only a centralized database architecture, the study of Lang *et al.* [39] proposed a cost model for a cluster database architecture design. The model takes into account the server configurations (e.g., CPU, disk and network bandwidth) and data processing parameters (e.g., predicate selectivity, hash table size), and predicts data processing performance and energy efficiency for a hash join operator. The maximum error reported is less than 10%.

2.2.2. Cost-Driven Techniques

The proposed cost models for predicting energy are then used by various techniques to optimize the database energy consumption. The authors in [22] proposed an Improved Query Energy-Efficiency by Introducing Explicit Delays (QED) mechanism, which uses query aggregation to leverage common components of queries in a workload. The explicit delay allows queries to be built up in a buffer queue, which then leads to a more efficient evaluation of the entire batch by multi-query optimization methods. The results for simple selection queries show that QED can be used to gracefully trade response time for energy, reducing energy consumption by 54% against 43% increase in average response time. The work of [40] indicates that power-aware scheduling of multiple queries can improve the energy efficiency of some basic SQL operators that can be run in parallels, such as aggregations and scans. The experimental results show that applying the proposed methods, the energy efficiency is increased by up to 4x. In the same direction, the authors of [41] propose a power-aware buffer cache management system for real-time databases on flash memory. The introduced technique creates a logical partition from the global buffer pool into read and write buffer pools, and dynamic feedback control of read/write buffer pool sizes to satisfy power consumption and deadline miss ratio performance goals. The work of [12] showed that processing a query as fast as possible does not always turn out to be the most energy-efficient way to operate a DBMS. Based on their proposed framework for predicting energy, they choose query plans that reduce energy consumption and meet traditional performance goals. The results show that energy savings can be up to 23% for a 5% increase in response time regarding equijoin queries. In a similar trend, the authors in [35] in-

[5]`http://www.postgresql.org/docs/current/static/planner-optimizer.html`
[6]`http://docs.oracle.com/cd/B10500_01/server.920/a96533/optimops.htm#51003`
[7]A peak power is the maximum power consumed during query execution

tegrate their cost model into the PostgreSQL's parameters to choose query plans having low power values during the optimization phase. The authors provide an in-depth analysis of the power profile for typical queries in TPC benchmarks to identify the resource consumption features during query processing. The experimental results show that power savings in the range of 11% - 22% can be achieved by equipping the DBMS with a query optimizer that selects query plans based on both estimated processing time and power requirements. In [11], using their cost model already introduced, the authors showed opportunities in generating query plans with attractive trade-offs between peak-power and time-efficiency considerations. The preliminary results indicate that for a certain query in the TPC benchmarks, the peak power can be reduced by 35 *watts*, with a time penalty of 80%. Moreover, the authors discuss extensions of the framework to exploit techniques such as multi-query workloads. The work of [4] conducts an in-depth study in analyzing the energy efficiency of a database server under different configuration parameters, arguing that the obtained benefits in database systems are small. However, in their settings, they only consider the CPU power rather than the system's overall active power. Nonetheless, in practice, there do exist queries that are I/O intensive. Considering only CPU instead of the overall components may significantly reduce the margin of improvements [42]. In another line of research, the authors of [39, 43] addressed the case of distributed parallel databases running on a cluster of nodes to save energy and design energy proportional[8] DBMSs. To that end, the authors outlined design decisions that have to be considered, such as the cluster's size, dynamic storage allocation and data processing over nodes, and scaling in/out the cluster according to the workload changes.

We would like to emphasize the originality of our line of approach which consists in proposing cost models measuring the energy consumption of queries in [37, 38] and reusing them during the process of selecting materialized views in the physical design phase of relational data warehouses [1]. To the best of our knowledge, this incorporation of the energy into the physical design is the first study that addresses this issue to date. In fact, the majority of earlier works about optimizing database energy dealt only with the query processing problem. This paper presents a significant improvement over our previous studies in matter of:

- review and classification of existing work;
- formalization of the materialized view selection problem in two different ways (energy as a constraint and energy as an objective function) as well as the inclusion of view maintenance constraint;
- proposal of an *a posteriori evolutionary algorithm* to solve our problem; and
- conduction of intensive experiments for both formalizations.

[8]proportionality is the ability to restrict hardware and software energy consumption to the actual workload.

2.2.3. Database as a Repository for Energy Data Storage and Management

Another approach towards smart and green data centers is the management of energy data. Time-based data of energy demand coming from multiple heterogeneous sources (e.g., supervisory control and data acquisition systems, smart meters, renewable generation systems) is called *time series* [44]. Therefore, forecasting, aggregating, scheduling, and disaggregating energy time series data is essential in optimizing the power flow in a smart grid and usually produces an added commercial value to the business [45]. For instance, a research report shows that a 1% increase in forecasting accuracy could yield estimated conservations in operational costs of approximately 14.14 million dollars [45].

Projects such as MIRABEL aims to develop an approach on a conceptual and an infrastructural level that allows energy distribution companies to balance the available supply of renewable energy sources and the current demand [46]. The challenge is that energy production is dependent on external factors (wind speed and direction, the amount of sunlight, etc.). Hence, available power can only be predicted but not planned, which makes it rather difficult for energy distributors to efficiently include renewable energy sources into their daily schedules. The project expects that using the proposed forecasting techniques, will result in peak-demand reductions of approximately 8-9% for the total grid [46]. In the same project, the database technologies are used for storing and querying the past and (forecast) future values of time series [47, 48]. Using database technologies allow for optimizations that improve the efficiency, consistency, and transparency of the overall forecasting process. The main idea is to compactly represent time series as models and integrate them into a DBMS as a data model. Moreover, they propose forecast queries to access to time series data, which improve the performance considerably.

3. An Initiative for Eco-Physical Design

In this section, we present in details our initiative, *Eco-Physic*, that concerns the physical design phase of the database life-cycle. It aims at proposing optimization structures that meet the needs and demands of the users in terms of the query performance and the consumed energy. The physical design phase is a funnel of the other phases of the life-cycle (Figure 4) since it integrates several parameters from these phases. Due to the diversity of optimization structures, we consider the case of materialized views. In the next section, we discuss all possibilities to integrate energy in the view selection process.

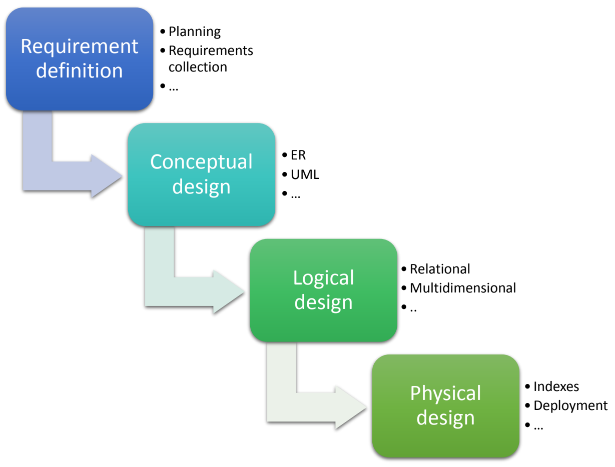


Figure 4: A systematic view of database design life-cycle.

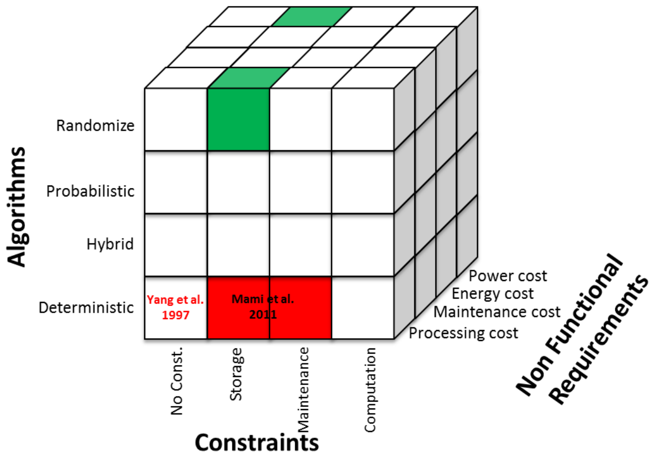


Figure 5: Warehouse of $\mathcal{VSP}$ solutions.

3.1. Integration of Energy in View Selection Problem

Recall that the $\mathcal{VSP}$ consists of finding an appropriate set of materialized views satisfying a given set of $\mathcal{NFR}$ such as query performance, reliability, usability, etc. The selected materialized views have to meet a set of constraints such as storage cost, maintenance cost, etc. The $\mathcal{VSP}$ is known to be an NP-Complete problem [19] due to the fact that the solution space grows exponentially as the problem size increases. The general formalization of $\mathcal{VSP}$ is defined as follows: Given:

- a database/data warehouse schema DB;
- a query workload: $\mathcal{W} = \{Q_1, Q_2, \cdots, Q_n\}$;
- a set of constraints: $\mathcal{C} = \{C_1, C_2, \cdots, C_p\}$;
- a set of non-functional requirement: $\mathcal{NFR} = \{nfr_1, \cdots, nfr_k\}$.

The $\mathcal{VSP}$ consists of selecting a set of materialized views: $MV = \{V_1, V_2, \cdots, V_m\}$ satisfying the fixed $\mathcal{NFR}$ and respecting the set of constraints $\mathcal{C}$.

The resolution of this problem are usually performed either by simple or advanced algorithms that span deterministic algorithms, randomized algorithms, hybrid algorithms, and constraint programming (for more details, please refer to the survey of [18]).

Several instantiations of this formalization exist. We propose to represent them by a *cube* having three main dimensions: (i) the $\mathcal{NFR}$, (ii) the constraints $\mathcal{C}$, and (iii) the algorithms used to solve the $\mathcal{VSP}$. These dimensions allow classifying the state-of-the-arts solutions. Figure 5 illustrates our cube.

If someone wants to integrate energy in the selection of materialized views, she/he has to assign it to one of the formalization inputs:

1. **Database as Repository for Storing and Managing Energy**. Energy consumption may be stored in a database. Project MIRABEL [46] is an example of this scenario.
2. **Energy as a constraint**. This scenario is feasible; for instance, a DBA wants to select a set of materialized views under a well-known energy constraint. Usually, it is not straightforward to estimate it a priori.
3. **Energy as a $\mathcal{NFR}$**. If the DBA is more sensitive about energy, she/he can consider it as the main objective of its selection or combine it with another objective such as query processing cost.

Our study considers the scenarios 2 and 3.

3.2. $\mathcal{VSP}$ under Energy as $\mathcal{NFR}$ Scenario

In this section, we present all ingredients to conduct the process of selecting materialized views: (i) the formalization of $\mathcal{VSP}$ and (ii) the methodology and the definition of cost models estimating each objective functions.

3.2.1. $\mathcal{VSP}$ Formalization

In our study, the $\mathcal{VSP}$ is formalized as follows: given (i) a database schema DB; (ii) a query workload; (iii) a storage constraint S; (iv) a maintenance time constraint T; (vi) two $\mathcal{NFR}$ representing energy consumption and query processing cost. The $\mathcal{VSP}$ consists of selecting a set of materialized views $MV = \{V_1, V_2, \cdots, V_m\}$ reducing query processing cost and saving energy, such that the size of the selected views is less than S and the total maintenance time of these views is less than T. The constraints state that the $\mathcal{VSP}$ is studied given a limited amount of resources e.g., storage space and maintenance time.

As we said before, the $\mathcal{VSP}$ is proven to be NP-Complete, due to the extremely large search space for possible materialized views. In our case, $\mathcal{VSP}$ becomes even more complicated, since we consider two objective functions (minimizing energy and time). Consequently, finding good solutions in an acceptable time is an additional challenge to this problem. This motivates us, in this first study, to deal

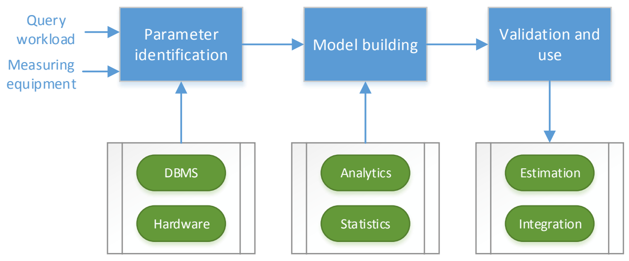


Figure 6: The development steps of our cost models.

with read-only OLAP queries that consume the major energy in data warehouses, as they are characterized by their long running time and heavy resources' utilization [11].

3.2.2. Cost Models

Our objective is to select materialized views that ensure a good trade-off between two costs: the query performance and the power consumption. As a consequence, we need mathematical models to predict these costs. In this section, we present our methodology to create the cost models. In [37], we proposed a software approach based on an empirical study to find the key parameters that impact on performance and energy when executing a query, and to identify the appropriate level to base our model. The empirical study also selects the suitable mathematical algorithm that describes the behavior of the SQL queries according to the $\mathcal{NFR}$. These choice are crucial and affect the accuracy of the final model. Assuming that the $\mathcal{NFR}$ is energy, the main steps to developing our cost model are (Figure 6):

1. **Parameter identification**. To decrease the energy dissipation of a system, it is important to measure the energy consumption of its components and study their behavior and correlation. This is related to the task of parameters identification phase. It takes as inputs: the hardware used, the DBMS, and query workload then produces the proper parameters that have an impact on energy.
2. **Model building**. The selected input parameters are used to build the energy consumption model using either *analytics* or *statistics* techniques such as regression, machine learning, etc. One of the challenging problems in this step is that some important system parameters such as the energy consumption of a particular component, cannot be measured directly. Conventional analytical methods which require a detailed knowledge of the system may not produce accurate results in such situations, and machine learning techniques can yield better results.
3. **Validation and use**. Next, the model must be validated according to different scenarios for its suitability for the intended purpose. It can then be used as a basis for predicting the energy consumption of queries. These estimates are used to improve the energy efficiency of DBMS, for example by incorporating the model in optimization techniques such as physical design, query scheduling, the dynamic voltage and frequency scaling (DVFS), etc.

Table 1: Cost model parameters notation.

Parameter	Definition
α_{cpu}	time to perform one I/O operation
α_{io}	time to perform one CPU operation
α_{net}	time to perform one communication operation
β_{cpu}	power to perform one I/O operation
β_{io}	power to perform one CPU operation
β_{net}	power to perform one communication operation
m	buffer memory size for the sort operation
$block$	DBMS page size
L	network bandwidth
T_i	the size of table i
f_q	execution frequency of query q
f_v	update frequency of view v
t_i	number of input tuple for the operator i
f	index selectivity
s	the size of input relation for the sort operator
n_{hash}	number of hash clauses in building phase
p_{hash}	number of hash partitions in probing phase
$p_{outer/inner}$	number of pages retrieved for the join operator
$t_{outer/inner}$	number of tuples retrieved for the join operator
n_{group}	number of grouping columns

The input parameters that can be considered in cost model building are listed in Table 1. Next, we will discuss in detail the development of our cost models.

3.2.3. Power Cost Model

To facilitate the understanding of our proposal, some concepts and definitions are given.

Definition 1. *The energy is referred to as the ability to do work.*

Definition 2. *The power represents the rate of doing work, or energy per a unit of time.*

Energy is usually measured in *joules* while power is measured in *watts*. Formally, energy and power can be defined as:

$$P = \frac{W}{T} \tag{1}$$

$$E = P \times T \tag{2}$$

Where P, T, W, and E represent respectively, a power, a period of time, the total work performed in that period of time, and the energy.

The power consumption of a given system can be divided into two parts (i) baseline power and (ii) active power.

Definition 3. *The Baseline power is the power dissipation when the machine is idle. This includes the power consumption of the CPU, memory, I/O, fans, and other motherboard components in their idle state.*

Definition 4. *Active power is the power dissipation due to the execution of the workload. The active power component is determined by the kind of workload that executes on the machine and the way it utilizes CPU, memory, and I/O components.*

There exist two concepts of power that have to be considered during the evaluation of power utilization in DBMS: (i) the *average power* which represents the average power consumed during the query execution and (ii) the *peak power* representing the maximum power in that duration. In this paper, we consider the average power.

The energy consumption can be reduced if either the average power consumption or the time intervals are reduced. Since optimizations in improving the performance of queries are widely studied, in this work we focus on the power part of the Equation 2. The first step is to model the power in order to estimate its consumption by the queries. To this end, we base on our power cost model proposed in [37]. Next, we describe the process its construction.

Model Parameters. In a traditional DBMS, query execution cost is treated as a linear combination of three components: CPU cost, I/O cost and communication cost [10]. We follow the same logic to propose a power cost model for a given workload. To process tuples, each operator in a query needs to perform CPU and/or I/O tasks. The cost of these tasks is the active power to be consumed in order to finish the tasks. In this paper, we focus on a single server setup. Thus, the communication cost is ignored. More formally, for a given workload W of n queries $\{Q_1, Q_2, \ldots, Q_n\}$. The power cost model for overall queries is defined as follows:

$$Power(W) = \frac{\sum_{i=1}^{n} Power(Q_i) \times Time(Q_i)}{Time(W)} \tag{3}$$

$Time(Q_i)$ and $Time(W)$ represent respectively, the execution time of the query i and the workload W. We get these factors from our performance cost model that we will describe next. As we said before, the $Power(Q_i)$ depends on the number of algebraic operations of the query Q_i. To illustrate this, let n_i be the number of these operations $\{OP_1, OP_2, \ldots, OP_{n_i}\}$. The cost $Power(Q_i)$ of a query i is given by the following equation:

$$Power(Q_i) = \beta_{cpu} \times \sum_{j=1}^{n_i} CPU\_COST_j + \beta_{io} \times \sum_{j=1}^{n_i} IO\_COST_j \tag{4}$$

Where β_{cpu} and β_{io} are the model parameters (i.e., unit power costs) for the operators. The *IO_COST* is the predicted number of *I/O* required for executing a specific operator. The *CPU_COST* is the predicted number of *CPU Cycle* and *buffer cache get* that DBMS needs to run a specific operator. These parameters are calculated with respect to the nodes (operators) currently materialized. A summary of the formulas used to calculate I/O and CPU costs for each basic operator can be found in Table 2 with part of the symbols already introduced in Table 1. These formulas are simplified from the cost model used by PostgreSQL optimizer.

Table 2: Cost model parameters for SQL operators.

Parameter	I/O Cost	CPU Cost
Sequential scan	p_{seq}	t_{seq}
Index scan	p_{index}	$t_{index} \cdot f$
Bitmap scan	p_{bitmap}	$t_{bitmap} \cdot f$
Nested loop join	$p_{outer} + p_{inner}$	$t_{outer} \cdot t_{inner}$
Sort merge join	$p_{outer} + p_{inner}$	$t_{sort(outer)} + t_{sort(inner)}$
Hash join	p_{outer}	$t_{outer} \cdot n_{hash} + t_{inner} \cdot p_{hash}$
Sort	$p_{sort}, s < m$; $0, else$	$t \cdot log_2(t)$
Aggregate	0	$t_{agg} \cdot$
Group by	0	$t_{group} \cdot n_{group}$

Polynomial Regression. Now, it remains to estimate the parameters β_{cpu} and β_{io} of the Equation 4. Our methodology uses regression technique to compute these power cost parameters. Simple linear regression technique, as used in [10, 11, 12], did not work well in our experiments, especially when data size change, this is because the relationships between data size and power are not linear. In other words, processing large files does not *always* translate in high power consumption. It depends more on the type of queries (I/O or CPU intensive) and their execution time. Therefore, we employed multiple polynomial regression techniques. This method is suitable when there is a *nonlinear* relationship between the independents variables and the corresponding dependent variable. Based on our experiments, the order m=*4* gives us the best results (the residual sum of squares is the smallest). The power cost $Power(Q_i)$ of the query Q_i is computed as:

$$\begin{aligned} Power(Q_i) = \beta_0 + \beta_1(IO\_COST) + \beta_2(CPU\_COST) + \\ \beta_3(IO\_COST^2) + \beta_4(CPU\_COST^2) + \\ \beta_5(IO\_COST \times CPU\_COST) + \\ \cdots + \\ \beta_{13}(IO\_COST^4) + \beta_{14}(CPU\_COST^4) + \epsilon \end{aligned} \tag{5}$$

Where *IO_COST*, *CPU_COST* denote the query I/O and CPU costs respectively, these costs are provided by the DBMS statistics module, and ϵ is a noise term that can account for measurement error. The β parameters are regression coefficients that will be estimated while learning the model from training data. Thus, the regression models are solved by estimating the model parameters β, and this is typically done by finding the least-squares solution [49].

By using I/O and CPU costs, we do not rely on SQL operator's type or implementation. As a result, our model can handle complex queries like those of TPC-DS benchmark for any given DBMS [37]. Our regression technique guarantees the portability of the model across machines; we can predict energy for arbitrary queries with just one-time training phase.

3.2.4. Query Performance Cost Model

In the same way, we use the algebraic operators to estimate the execution time of the query. The performance cost model for the workload is defined as follows:

$$Time(W) = \sum_{i=1}^{n} Time(Q_i) \tag{6}$$

The cost $Time(Q_i)$ of a query i is given by the following equation:

$$Time(Q_i) = \alpha_{cpu} \times \sum_{j=1}^{n_i} CPU\_COST_j + \alpha_{io} \times \sum_{j=1}^{n_i} IO\_COST_j \tag{7}$$

Where IO_COST, CPU_COST are already described. The α parameters are coefficients specified to our test machine, used to convert the query costs to time values. α_{cpu} is the CPU time required to execute one $CPUCycle$ and α_{io} is the IO time needed for the device to execute one I/O operation.

Due to the changes that may occur in the underlying base relations, materialized views have to be kept up to date. Therefore, the view maintenance cost has to be considered during $\mathcal{VSP}$. View maintenance can be computed using strategies such as incremental maintenance strategy or recomputation strategy. In our work, we assume incremental maintenance, when the base relation changes, we only calculate the changes that occurred in view and then spread such changes to materialized view. Let LV be the set of views candidates available from workload nodes, and MV ($MV \in LV$) be the set of materialized views. Each view $v \in LV$ has an update frequency f_v associated with it. Then, the cost of updating MV is:

$$U(MV) = \sum_{v \in MV} f_v \times Cost(v) \tag{8}$$

Where $Cost(v)$ is the maintenance cost of view v. It is computed by summing the number of changes in the base relations from which v is updated. To calculate this changes, we base on the techniques described in [50].

3.3. Materialized View Selection Methodology

In this section, we describe our methodology to solve $\mathcal{VSP}$. Like in the state-of-the-art studies, the selection process of materialized views starts mainly by the construction of global query graph ($\mathcal{GQG}$) merging all individual query execution plans of the workload (called *Multi-Views Processing Plan* in [51]). The $\mathcal{GQG}$ is a directed acyclic graph. It has four levels: the first level contains the leave nodes representing the base tables of the database. At the second level, we find nodes which represent the results of unary algebraic operations such as selection and projection. The third level contains nodes representing binary operations such as join, union, etc. The last level represents the results of each query. Every intermediate node of the graph is tagged with its energy cost and query processing benefit. A sample $\mathcal{GQG}$ is shown in Figure 7.

Note that each intermediate node of the $\mathcal{GQG}$ is a candidate for materialization. To select the best $\mathcal{GQG}$ which has the minimum cost, Yang *et al.* proposed in [51] two algorithms. The first algorithm generates all possible $\mathcal{GQG}$ and the plan with the minimum cost will be chosen. The second algorithm is based on 0-1 integer programming, it generates materialized views candidates who have positive benefit between query processing and view maintenance, then only candidate nodes with positive benefit are selected to be materialized. The main limitation of Yang *et al.* work is the scalability of the algorithm used to construct the $\mathcal{GQG}$ since generating all possible global plans is not feasible for applications containing a large set of queries.

To counter this problem, Boukorca *et al.* [52] proposed a new approach based on the graph data structure inspired from the Electronic Design Automation (EDA) domain. The generation process is composed of 5 steps: (1) parsing the SQL query to identify the logical operations (nodes), (2) modeling the join nodes by a hypergraph, where the vertices set represents the set of join nodes and the hyperedge set represents the queries workload, (3) computing the connected components using the partition hypergraph algorithms of electronic circuits design, resulting in a set of disjoint components of queries, each component is represented by a sub-hypergraph, (4) transforming each sub-hypergraph into an oriented graph using a cost model and implementation algorithms, (5) merging the resulting graphs to generate the global query graph ($\mathcal{GQG}$). To show these points, let us consider an example using Boukorca *et al.* algorithms [52].

Example 1. *Supposing a workload composed of 7 OLAP queries issue from the SSB benchmark [53]. The benchmark contains a fact table* `Lineorder` *(denoted by $\mathcal{L}$), and four dimension tables:* `Customer` *($\mathcal{C}$),* `Supplier` *($\mathcal{S}$),* `Part` *($\mathcal{P}$) and* `Dates` *($\mathcal{D}$). The $\mathcal{GQG}$ corresponding to these 7 queries is represented in Figure 7. From this global query plan, two main alternatives to choose relevant materialized views are possible: (i) materialize all the nodes or (ii) materialize some of the intermediate nodes. In this example, we consider the join nodes denoted by J_i as candidate views, the total number of possible configurations is 2^n, where n is the number of join nodes ($n = 4$). The fact that we have only 16 configurations, we use an exhaustive evaluation. For each configuration, we measure the total workload execution time and average active power consumption. The datasets, system setup, and the measurement tools used in our experiments can be found in Section 5. The obtained results are summarized in Table 3. Note that a configuration may contain one or many materialized views, also, we relax storage and maintenance*

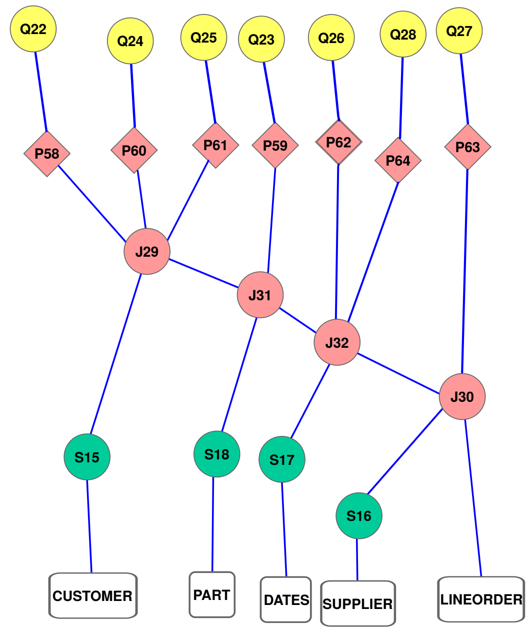


Figure 7: Example of $\mathcal{GQG}$ of 7 queries.

Table 3: Workload performance and active power consumption for different view configurations.

Materialized Views	Time (min)	Power (*watts*)
$\mathcal{C}$, $\mathcal{P}$, $\mathcal{D}$, $\mathcal{S}$, $\mathcal{L}$	10.83	16.07
J_{32}	3.2	19.73
J_{31}	5.13	18.06
J_{31}, J_{32}, J_{30}	2.28	21.17
J_{29}	6.18	17.66
J_{29}, J_{31}, J_{30}	2.45	21.01
$J_{29}, J_{30}, J_{31}, J_{32}$	1.9	23.11

constraints for simplicity. Several lessons can be learned from this experiment:

- *the scenario consisting in materializing all the candidate views gives the best query performance with the highest power consumption. Thus, processing queries as fast as possible is not always the most energy-efficient way, this has been already mentioned in previous researches, such as of [12, 35, 11];*
- *the scenario in which none view is selected offers the worst performance and lowest power consumption. In practice, this situation is not preferred since improving the performance is primordial for any database;*
- *finally, materializing some views among all candidates is the best compromise. For instance, materializing J_{32} gives a good scenario ensuring the compromise. This example motivates us to firstly define an algorithm for selecting the relevant configurations that avoid the exhaustive search and then choosing the best solution satisfying the requirements of DBA.*

4. Multi-objective Optimization

In multi-objective optimization problems, it is essential to compromise between objectives. A general multi-objective optimization includes a set of n parameters (decision variables), a set of k objective functions corresponding to $\mathcal{NFR}$, and a set of m constraints. The optimization goal is to [20]:

$$\begin{aligned} \text{minimize } y = \ & f(x) = (f_1(x), f_2(x), \cdots, f_k(x)) \\ \text{subject to } & e(x) = (e_1(x), e_2(x), \cdots, e_m(x)) \leq 0 \\ \text{where } & x = (x_1, x_2, \cdots, x_n) \in X \\ & y = (y_1, y_2, \cdots, y_k) \in Y \end{aligned}$$

Where x is the decision vector, y is the objective vector, X is denoted as the decision space, and Y is called the objective space. The objective functions in our problem are the power (Equation 4) and the performance (Equation 7) respectively, under the storage size constraint to ensure that the total space of materialized views is at most equal to the storage space capacity (S), and the maintenance cost constraint to guarantee that the update costs of views to be materialized is less than the available maintenance time (U). More formally, the multi-objective $\mathcal{VSP}$ for a solution MV can be expressed as:

$$\begin{aligned} \text{minimize } & f(x) = (Time(MV_x), Power(MV_x)) \\ \text{subject to } & e(x) = S(MV_x) \leq S,\ U(MV_x) \leq U \end{aligned}$$

Where $S(MV)$ and $U(MV)$ represent respectively the storage and the maintenance costs of MV.

In multi-objective optimization, there does not typically exist a solution that minimizes all objective functions simultaneously. Therefore, solutions that cannot be improved in any of the objectives without degrading at least one of the other objectives are called *Pareto optimal solutions*. More formally, a solution x_1 is said to *dominate* another solution x_2 if $f_i(x_1) \leq f_i(x_2)$ for all indices $i \in \{1, 2, \ldots, k\}$ and $f_j(x_1) < f_j(x_2)$ for at least one objective function j. A solution x_1 is called Pareto optimal if there does not exist another solution that dominates it. The set of Pareto optimal outcomes is called the *Pareto front*. If we consider our previous example, the performance/power values off all possible materialized configurations are plotted in Figure 8. The point D is not on the Pareto Frontier because it is dominated by both point A, B, and point C. Points A, B, and C are not strictly dominated by any other, and hence do lie on the frontier. The ultimate goal is to identify solutions in the Pareto optimal set. However, identifying the entire Pareto optimal set, for our problems, is practically impossible due to its NP-Completeness. Therefore, an efficient approach is to investigate a set of solutions that represent the Pareto optimal set as well as possible.

4.1. Resolutions Techniques

As we stated in the above section, multi-objective optimization returns a set of Pareto solutions. However, selecting a single solution to be deployed in the system by

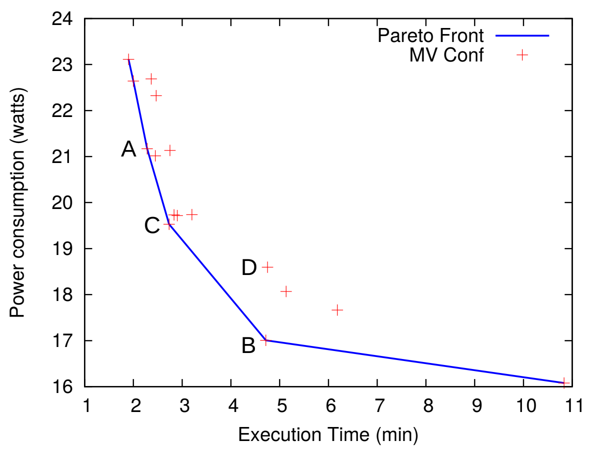


Figure 8: Pareto Front example when minimizing two objectives (performance and power).

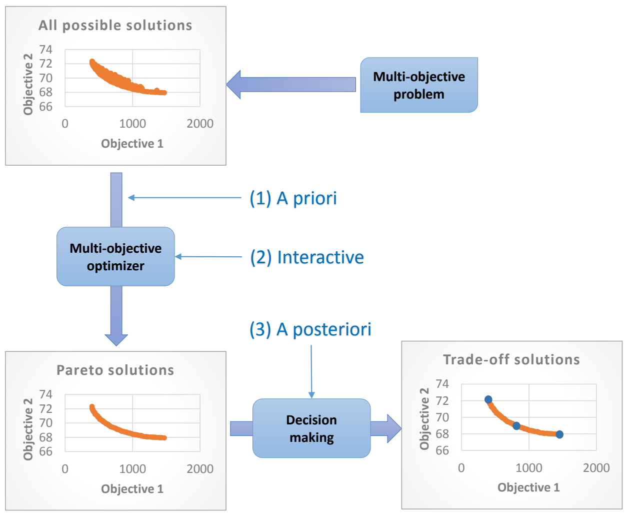


Figure 9: Process for application of multi-objective optimization.

the database administrator can be very hard regarding the large set of returned solutions. Therefore, the problem of choosing a good trade-off among the objective functions has been tackled in the literature and solved using three methods: (i) *a priori* methods (ii) *interactive* methods (iii) and *a posteriori* methods [54]. These approaches are incorporated at the processing stages of multi-objective optimization as shown in Figure 9. The description of each approach is as follows:

1. **A priori methods**. In these methods, the DBA intervenes *before* the execution of the optimization process. The DBA defines the trade-off between the objective functions and performing only one search step to obtain the desired solution. This method is interesting since it requires a single search step. However, it is not straightforward for the DBA to define the trade-off values without a knowledge about the problem and without seeing any solutions beforehand, especially in our case where we are dealing with a new non-functional requirement which is the energy.
2. **Interactive methods**. In the second type of methods, the DBA intervenes *during* the execution of the optimization process. She/he is asked to set the preference trade-off in order to gradually reorient the search space toward a preferred solution. Although this method can achieve a good solution respecting the DBA preference, it requires the full attention of the DBA evaluating each solution quality during the whole optimization process, which is time-consuming in large search space. Moreover, domain-specific knowledge of the DBA is needed to make good decisions.
3. **A posteriori methods**. In the last type of methods, the DBA intervenes *after* the execution of the optimization process. This method provides the DBA a set of solutions and choosing the best trade-off is postponed to the end of the optimization process. The aim is to show these solutions to the DBA allowing her/him to judge and select the most preferred among various proposed solutions. That is, the optimization process is fully automated and require no DBA attention or knowledge about the problem. However, if the result set is too large, the process may take a long time to finish.

In our study, we have opted for an *a posteriori* approach to give DBA the possibility to *prioritize or not* the energy over the query performance. In the next section, we discuss in details the proposed algorithms to deal with our problem.

4.2. Evolutionary Algorithms

Evolutionary Algorithms (EAs) are suitable for multi-objectives optimization problems through which large search spaces can be handled and multiple alternative trade-offs can be generated in a single optimization run [20]. The general idea behind EA is to simulate the natural evolutionary process in which the fittest individuals will survive after several generations. Details of our implementation using the EA are described in the following sections.

4.2.1. Solution Representation

To represent solutions of a $\mathcal{GQG}$, all candidates' views are tagged. The solutions are represented as a bit string of 1s and 0s such that if a particular view is selected for materialization, then the corresponding bit in the solution string is represented as 1 and else 0. For example, in the Figure 7, the solution $\{1, 1, 1, 0\}$ indicate that the nodes $\{J_{29}, J_{30}, J_{31}\}$ are selected for materialization. For each solution, the cost functions return the estimated execution cost and the power consumption of the workload; the size and the maintenance cost of the views. The smaller value the cost functions get subject to the size and maintenance constraints, the better the solution is.

4.2.2. Fitness Function

The fitness function measures how good a solution (i.e., a set of selected views to materialize) is. We use the Pareto-ranking approach as fitness function, based on the genetic algorithm NSGA-II proposed by Deb *et al.* [21], which explicitly utilizes the concept of Pareto dominance in evaluating solutions quality or assigning them a selection probability. This algorithm uses techniques to enhance the convergence and guarantee diversity and spread of solutions. We use binary tournament selection in which individuals are randomly chosen in a "tournaments" manner. The population is ranked according to a dominance rule, and then each solution is assigned a fitness value based on its rank in the population. Since we want to minimize our objectives functions, therefore, a lower rank corresponds to a better solution. The infeasible solution, i.e., a solution that violates constraints is handled as follows: basically, a feasible solution will always dominate an infeasible one, and then if all solutions are infeasible, the ones with slower constraint violation survive.

4.2.3. Crossover

Crossover encourages information swaps among different individuals. It helps the inheritance of good genes from one generation to the next and assembling better individuals. Half uniform crossover is used in our evolutionary algorithms; in this method half of the bits which are different between the parents will be exchanged. For this purpose, first, it calculates the number of different bits using the Hamming distance between the parents. The half of this number is the number of bits exchanged between parents to form the child's. For example, given two individuals P_1 and P_2, the results of the crossover are C_1 and C_2:

$$P_1 = \{0,1,0,1,1,1,0,1,0\} \quad P_2 = \{0,1,1,1,1,0,1,1,0\}$$
$$C_1 = \{0,1,0,1,1,0,1,1,0\} \quad C_2 = \{0,1,1,1,1,1,0,1,0\}$$

To generate C_1 and C_2, the algorithm counts the number of different bits between P_1 and P_2, which is 4, half of this number (i.e., 2) will be exchanged to create the new individuals (the bits number 6 and 7).

4.2.4. Mutation

Mutation is the occasional random alteration of a bit string position value. It introduces new features that may not be present in any member of the population. The mutation is performed using Flip Bit method in our work. Each bit is flipped (switched from a 0 to a 1, or vice versa) using the specified probability. For example, suppose that the 7^{th} bits of the individual L_1 is chosen for mutation, the value of the bit string is 1, it will be flipped to 0 with the specified mutation rate, the new individual is L_1':

$$L_1 = \{0,1,0,1,1,0,1,1,0\}$$
$$L_1' = \{0,1,0,1,1,0,0,1,0\}$$

4.3. Decision Making

Note that evolutionary algorithms produce a set of solutions. To give the DBA the possibility to pick the best solution among this set according to her/his fixed trade-off (between query performance and consumed energy), we propose to use the *weighted sum of the objective functions method* (WSOF). In this aggregation method, we calculate the weighted sum of the normalized objective functions to aggregate objectives and have an equivalent single objective function to be optimized. This method is defined as follows [20]:

$$\begin{gathered} minimize \quad y = f(x) = \sum_{i=1}^{k} \omega_i \cdot f_i(\overrightarrow{x}) \\ \sum_{i=1}^{k} \omega_i = 1 \end{gathered} \tag{9}$$

Where ω_i are the weighting coefficients representing the relative importance of the k objective functions of the problem. To normalize the range of objective functions values, we use the following formula:

$$x' = \frac{x - min(x)}{max(x) - min(x)} \tag{10}$$

Where x is an original value and x' is the normalized value. The WSOF method is well suited when the Pareto front is convex, such as our problem. An example of the trade-off is the point C in Figure 8 ($\omega_1 = 0.5$, $\omega_2 = 0.5$).

Algorithm 1 summarizes the process of solving the $\mathcal{VSP}$. It identifies a set of views (MV) that respecting two constraints: the storage size and the maintenance time, then applies the WSOF method to get the view configuration that has the smallest possible processing cost of the query workload.

The overall process of our methodology is described in Figure 10. It takes a query workload as input, then it creates the unified graph corresponding to that workload using Boukorca *et al.* algorithms [52]. Intermediate nodes of this graph are considered as a candidate for materialization. Finally, the EA is applied until the maximum number of iterations is reached. As we said before, the solution is a set of Pareto front points that represents the best views configuration regarding performance and power. To get a unique solution, the DBA sets the desired trade-off parameters in the WSOF method. The output is one configuration of views.

5. Experiments and Results

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

Algorithm 1 Energy-Driven $\mathcal{VSP}$

```
Input: $\mathcal{GQG}$, a global query graph
Output: $MV$, the selected set of views for materializing
 1: $LV \leftarrow \emptyset$;
 2: $CurrentIteration \leftarrow 0$;
 3:
 4: while $CurrentIteration < MaxEvaluations$ and $HasMoreCandidatNodes(\mathcal{GQG})$ do
 5:     $CV \leftarrow GAGetCandidatNodes(\mathcal{GQG})$;
 6:     $TimeCost \leftarrow GetTimeCost(CV)$;
 7:     $PowerCost \leftarrow GetPowerCost(CV)$;
 8:     $Constraint_i \leftarrow GetConstraint_i(CV)$;
 9:     if $Constraint_i \leq MaxConstraint_i$ then
10:         $LV \leftarrow CV$;
11:     end if
12:     $CurrentIteration \leftarrow CurrentIteration + 1$;
13: end while
14:
15: $WF \leftarrow \emptyset$;
16: foreach $v \in LV$ do
17:     $WF \leftarrow GetWeightFunctions(GetTimeCost(v), GetPowerCost(v))$;
18: end for
19: $MV \leftarrow GetMinimum(WF)$;
20:
21: return $MV$;
```

5.1. Experiment Setup

We use a similar setup utilized in the state-of-the-arts [12, 10, 11]. Our machine is equipped with a "Watts UP? Pro ES"[9] power meter with one second as a maximum resolution. The device is directly placed between the power supply and the database workstation under test to measure the workstation's overall power consumption. The power values are logged and processed in a separate monitor machine.

We use a Dell Precision T1500 workstation having an Intel Core i5 2.27GHz processor and 4GB of DDR3 memory. Our workstation machine is installed with the latest version of Oracle 11gR2 DBMS under Ubuntu 14.04 LTS with kernel 3.13. We use Star Schema Benchmark datasets and queries with 10GB and 100GB scale factor generated using the benchmark tools. The SSB benchmark illustrates decision support systems that examine large volumes of data, execute different types of queries with a high degree of complexity. The queries are executed in an isolated way in the experiments.

We developed a simulator tool in Java platform. This simulator implements Boukorca *et al.* algorithms that generate the $\mathcal{GQG}$. The tool can automatically extract the database's meta-data characteristics. Moreover, it integrates MOEA Framework[10], which is a Java library for multi-objective evolutionary algorithms. For each materialized view configuration produced by MOEA Framework, the simulator calculates the performance and the power consumption using the cost models. In our experiment, we consider three types of materialized views configuration: (1) Power-MV, which is the configuration that gives the minimal power cost, (2) Time-MV, is the configuration with minimal time cost, (3) Tradeoff-MV, using WSOF method with $\omega_1 = 0.5$, $\omega_2 = 0.5$. The parametric configuration of the evolutionary algorithm is given in Table 4. The tournament selection size, crossover, and mutation probabilities are the MOEA Framework default values and we decided to use those for our experiments. The maximum number of iteration fixed at 10,000 gives us good results.

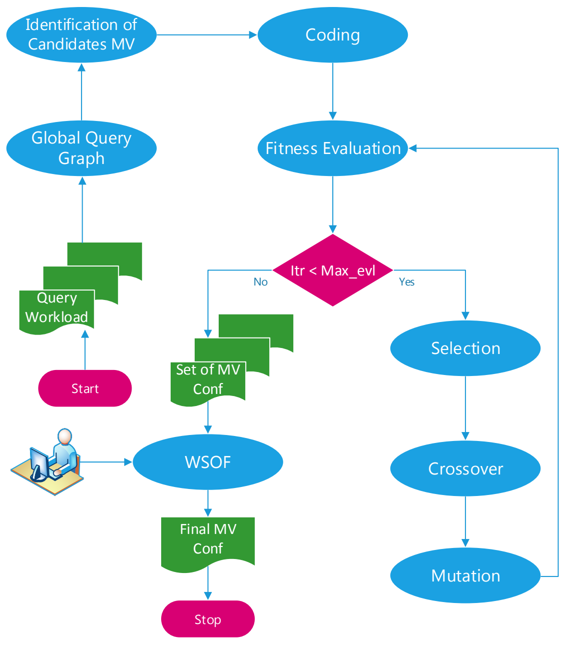


Figure 10: The process of views selection methodology.

Table 4: Default evolutionary algorithm parameters.

Parameter	Value
Encoding Type	Bit String
Selection Method	Binary Tournament
Tournament Selection Size	2
Crossover Type	Half Uniform
Probability of Crossover	1.0
Mutation Type	Flip Bit
Probability of Mutation	0.01
Max Evaluations	10000

5.2. Power Model Building

As mentioned earlier, the β parameters are estimated while learning the model from training data. To this end, we perform series of observations in which queries are well-chosen, and while running these queries, the power values

[9]https://www.wattsupmeters.com/
[10]http://www.moeaframework.org

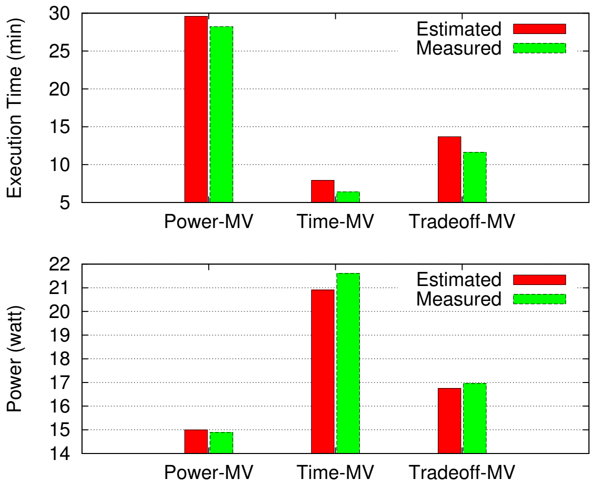


Figure 11: Estimation errors in predicting workload performance and power consumption.

consumed by the system are collected using a measurement equipment. In the same time, for each training instance, we calculate their I/O and CPU costs. To generate training instances, we create our custom query workload based on SSB datasets. The workload contains queries divided into two main categories: (i) queries with operations that exhaust the system processor (CPU intensive queries) and (ii) queries with exhaustive storage subsystem resource operations (I/O intensive queries). Note that the considered queries include: queries with a single table scan and queries with multiple joins involving different predicates. They also contain sorting/grouping conditions and simple/advanced aggregation functions as in [11]. After collecting training queries power consumption, we apply the regression equation (5) using the *R language software*[11] to find our model parameters. Once we get them, an estimation of new queries is obtained without the use of our measurement equipment.

5.3. Results

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

5.3.1. Cost Models Error

In this type of experiment, given the predicted performance and power cost predicted by our models for query workload, we compare it with the actually observed execution time and system active power consumption. To evaluate our proposal, we run a 30 OLAP queries of the SSB benchmark against 10GB database size using the three materialized views configurations. The results are shown in Figure 11. The experiment demonstrates the accuracy of our prediction model. The average estimation error is 12% for the execution time and 1.7% for the power, indicating that the models are sufficiently accurate for the intended applications.

[11]http://www.r-project.org/

Table 5: Power estimation errors in TPC-H benchmark queries with different database sizes.

Query	Error (%)		Query	Error (%)	
	10GB	100GB		10GB	100GB
$Q1$	1.03	0.2	$Q11$	4.2	-
$Q2$	-	-	$Q12$	0.9	0.02
$Q3$	1.5	1.2	$Q13$	4.7	4.4
$Q4$	0.6	0.5	$Q14$	2.8	2.4
$Q5$	1.2	3.07	$Q15$	0.4	2.7
$Q6$	4.1	2.7	$Q16$	5.4	0.03
$Q7$	0.4	1.4	$Q18$	0.4	-
$Q8$	0.07	1.09	$Q19$	1.6	0.9
$Q10$	0.6	0.3	$Q22$	1.2	0.4

5.3.2. Cost Models Robustness

In this experiment, we evaluate the robustness of our cost model to a variety of changes in the database system environment. We test our model against a new (i) system configuration, (ii) database server and (iii) benchmark datasets and queries. Specifically, we use a Dell PowerEdge R310 workstation having a Xeon X3430 2.40GHz processor and 32GB of DDR3 memory. The database server used is PostgreSQL 9.4.5, and the benchmark is TPC-H[12] with 10GB and 100GB scale factor.

In the experiment, given the estimated power cost predicted by our model *(E)*, we compare it with the actually observed system active power consumption *(M)*. To quantify the model accuracy, we used the following error ratio metric: $Error = \frac{|M-E|}{M}$. To test our model with large datasets, we run all 22 queries of the TPC-H benchmark against two database scale factor: 10GB and 100GB. The results are shown in Table 5. We note that some queries were aborted since they exceeded 72 hours of execution in our current test environment.

As we can see from the table, the average error is typically small (0.1% in both 100GB and 10GB datasets), and the maximum error is usually below 5%. Thus, the experiment confirms the robustness of our power cost model.

5.3.3. Solutions Quality

To check the quality of the solutions offered by our proposed algorithms, we create a simple workload of 15 queries. We first generate its $\mathcal{GQG}$ and then run two algorithms to obtain configurations of materialized views: (i) the exhaustive search algorithm and (ii) our evolutionary algorithm. To get the Pareto Front points from the exhaustive search space, we use the well-known Block-Nested Loops algorithm (BNL) [55]. The BNL algorithm repeatedly reads the set of solutions and keeps a window of incomparable tuples in main memory. When a solution s

[12]http://www.tpc.org/tpch/

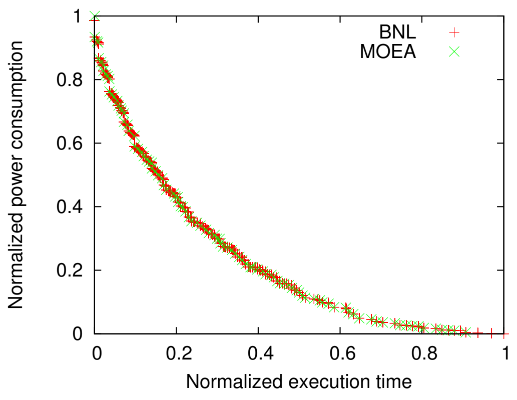


Figure 12: Comparison between evolutionary algorithm and exhaustive algorithm solutions.

is read from the input, it will be compared to all solutions of the window. Based on this comparison, s is either eliminated or placed into the window based on the dominance rule. The experiment results of the two algorithms are plotted in Figure 12. As we can see from the figure, the solutions resulted by evolutionary algorithm present a high degree of optimality as compared to the BNL solutions with a good spread in the space. However, the execution time of the evolutionary algorithm is very low compared to the exhaustive search algorithm, for a $\mathcal{GQG}$ of 200 query, the first takes 7 seconds while the second takes 4 days without completing its execution.

5.3.4. Impact of I/O and CPU Costs

The purpose of this experiment is to investigate the performance/power profiles of materialized views configuration. This helps us further understand the impact of I/O and CPU costs of views and their relationship with the performance and power consumption. Based on such knowledge, we can design database with power savings and energy efficiency strategies in mind. We used our 30 query workload to generate the correspondent $\mathcal{GQG}$, then using the evolutionary algorithm, we get the dominates materialized view configuration, we take the Time-MV, Power-MV, Tradeoff-MV configurations and calculate their execution time, power consumption, total I/O cost, and total CPU costs. The values are normalized to CPU cost; results are plotted in Figure 13. From the figure, we notice three cases: (1) when both I/O and CPU costs are small, the execution time is short, but the power is high, (2) on the other hand, when the I/O and CPU costs are high, the execution is also long, but the power consumption is small and (3) in the last configuration, the costs and the time/power are in trade-off values. The immediate observation is that the execution time and the I/O cost are nearly the same in all configurations, this already known in traditional database optimization because the CPU is very fast compared to I/O devices. The second observation is neither the I/O nor the CPU costs contribute directly to the power consumption. Our interpretation is

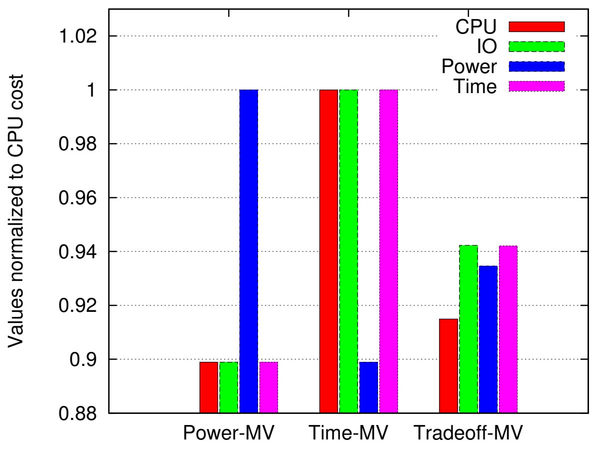
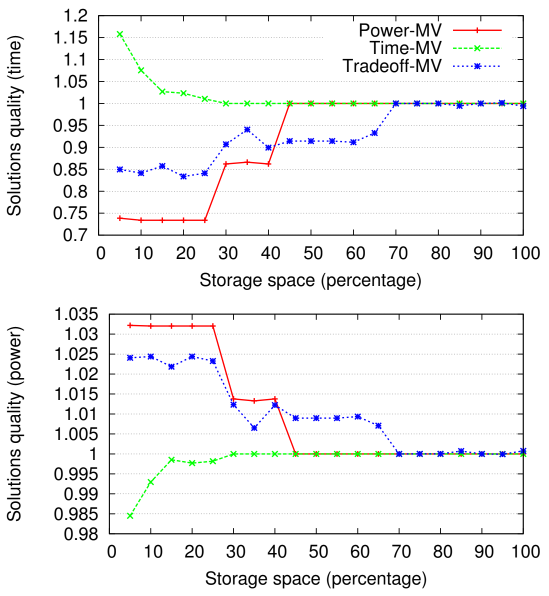


Figure 13: Materialized views characterization and their impact on performance and power consumption.

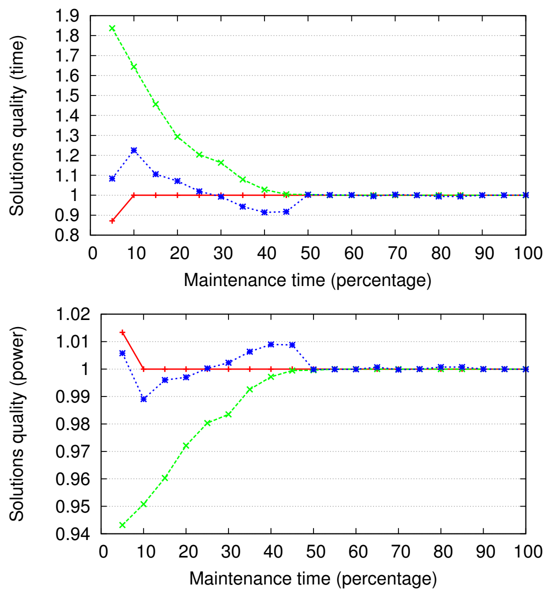
that for some queries, the operation of reading large files needs more I/O tasks. When intermediate results cannot fit into memory, they will be written on the disk and read later. Such overhead is translated in more CPU waits. In other terms, the query optimizer spends more time in reading/writing instead of processing data. Thus, queries dominated by I/O cost lead to less power consumption. On the other side, when the size of files is small, the operation of data reading finishes quickly and in its remaining time, the query is dominated by CPU processing. This can be translated in high power consumption. Thus, to produce materialized views with a trade-off between performance and power, we recommend choosing views with a medium I/O and CPU costs.

5.3.5. Impact of Storage Space

In this set of experiments, we consider the storage space constraint. Let S be the storage space that is required to save all the materialized views candidates; S is varied from 5% to 100%; the workload is the same used in previous experiments (30 OLAP queries from the SSB benchmark). We compute the quality of the solutions (Q_s) as a ratio of each solution costs over the best solution costs that our algorithm can obtain when constraints are relaxed. The Pareto optimality (cf. Section 4) implies that in some cases Q_s is greater than 1, however, the closest the Q_s-value to 1, the better solution quality. For each configuration type, we calculate the variation of the solutions quality for the workload in term of execution time and power consumption with respect to the storage space allotted for materialized views. Results are plotted in Figure 14a. We note that the solutions quality improves when S get relaxed, since there will be more disk space to store the materialized views. From the figure, we notice that the evolutionary algorithm finds the best solution for Time-MV quickly, exactly from $S = 30\%$ (i.e., $Q_s = 1$), this is not the case for the Power-MV, where the best solution are found in $S = 45\%$. In consequence, the Tradeoff-MV

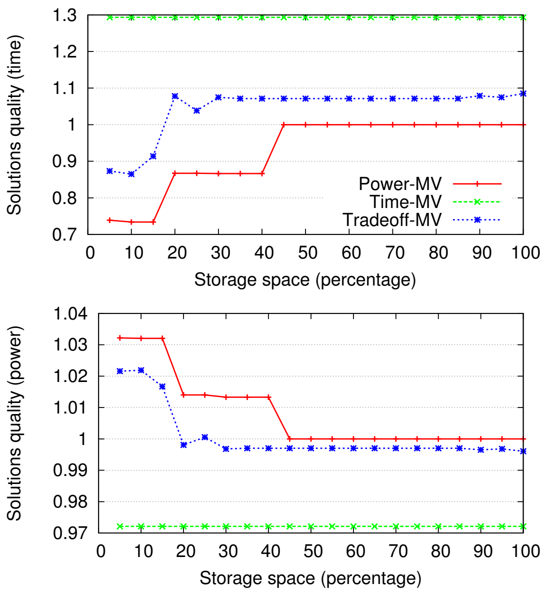


(a) Impact of size constraint.

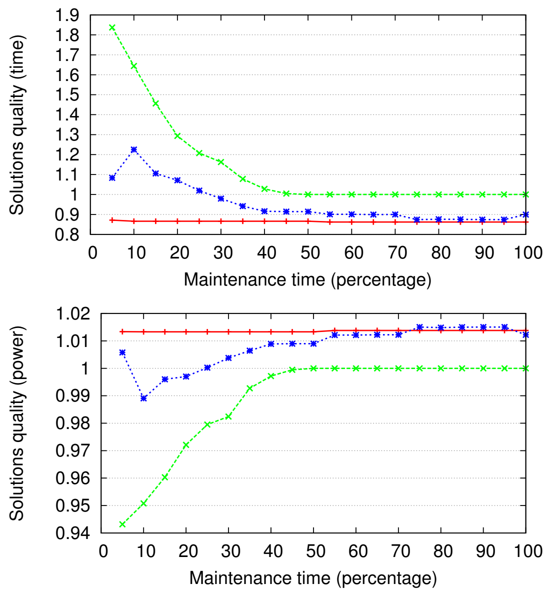


(b) Impact of maintenance cost constraint.

Figure 14: Solutions quality in term of performance and power consumption with resources constraint.



(a) Impact of size constraint ($T = 20\%$).



(b) Impact of maintenance cost constraint ($S = 40\%$).

Figure 15: Solutions quality in term of performance and power consumption with different resources constraint combinations.

became stable until $S = 70\%$. This is due to the diversity of the solutions provided by the evolutionary algorithm; a good trade-off can be seen in S between $70\% - 100\%$ of the total space. The figure also shows that the performance and the power values are in opposite direction (i.e., when the power is high the time is short and vice versa), this matches the Pareto front points previously discussed.

5.3.6. Impact of Maintenance Cost

The aim of this experiment is to select a set of views while meeting the maintenance cost constraint. We evaluate this constraint as follows. Let T be the total view maintenance cost time when the result of each candidate's nodes of the query workload is materialized; T is varied from 5% to 100%. In Figure 14b, we show the quality of the solutions found by our approach in terms of execution time and power consumption. Similarly to space constraint, the solutions quality improves when T get relaxed because there will be more time to maintain the materialized views. As we can see from the figure, the best solution for Power-MV is found when $T = 10\%$, while the other configurations get converged to the best solution until $T = 50\%$. This is due to the power-intensive characterization of the maintenance task since its involve many operations of comparison to compute the change in the base relation, which turn in more CPU tasks and more power consumption. The best trade-off solution gets found quicker compared to the space constraint, although the two constraints seem similar, they have a significant difference. The space occupied by a set of views always increases when a new view is inserted, while the maintenance cost does not; it is possible to decrease the update time of a set of views after the addition of a new one [18]. This is clearly shown in the figure, when T is between $5\% - 25\%$ of the total time window, the solutions are more performance efficient. On the other hand, when T is between $25\% - 50\%$ the solutions are more power efficient. After $T = 50\%$, the time window becomes sufficient to get the best trade-off solution.

5.3.7. Impact of Maintenance Cost and Storage Space

In order to examine the impact of both space and maintenance cost constraints on solutions quality, we conduct this set of experiments. In the first case, we set time window T to 20% and vary the values of the space constraint S from 5% to 100%. In Figure 15a, we illustrate the quality of the solutions provided by the EA for various values of S. We observe that only the Power-MV configuration converges to the best solution, while the Time-MV configuration doesn't converge in both execution time and power consumption objectives (i.e., $Q_s \approx 1.3$, $Q_s \approx 0.97$ respectively). This suggests that, in this situation, restrictive values of the maintenance constraint produces solutions that are power efficient but time-consuming. Therefore, the trade-off solution is not reached even when the S is completely relaxed because the maintenance cost constraint becomes the significant factor.

In the second case, we set storage space S to 40% and vary the values of the maintenance constraint T from 5% to 100%. Figure 15b shows the results of solutions quality when T get varied. Once again, only one configuration which is Time-MV converges to the best solution, the two others fail to converge even when T get relaxed (i.e., $Q_s \approx 0.9$ for time and $Q_s \approx 1.015$ for power), this is due to the fact that the space constraint is a dominant factor in this situation. Furthermore, the results state that restrictive values of the storage space favorites solutions with low execution time but significant power consumption. Similarly to Figure 14b, the trade-off solutions change significantly with the maintenance time variation, this supports our finding in the experiment of Section 5.3.6, the best value can be seen only at $T = 25\%$.

The results presented above clearly show the impact of the storage space and maintenance constraints on the performance/power in the database physical design process. Therefore, DBA should set the storage and maintenance constraint parameters carefully to get a good trade-off between the two objectives or to prioritize performance or power oriented solutions.

5.3.8. Evaluation of the Energy as Constraint Scenario

Supposing a scenario where a service provider such as data center or cloud computing operator wants to optimize a query workload by selecting a set of materialized views that reduce power consumption while keeping a predefined performance target. This is a common situation in the cloud environment or the current tendency of the database as a service (DBaaS) approach. In such situations, the objective is to meet service level agreements (SLAs) and optimize resource usage such as power consumption, in order to maximize the profit margins. We propose to use our methodology to help operators identifying the minimum value of power that ensures performance meeting the fixed SLA. In this problem, the power consumption becomes a constraint to satisfy. Let P be the workload power when all views are materialized minus the power of running workload queries without materialization; P is varied from 5% to 100%. The SLA is defined as the execution time of the best trade-off solution that our EA can achieve. For each value of P, we define the SLA violation of the provided solution as a ratio of the execution time to the execution time of the best solution. Note that we relax the others two constraints, we show the experiment results in Table 6. From the table we can see that using our techniques, the SLA violation falls down to 0 when $P = 45\%$. Thus, setting the power limit to this value guarantees the SLA fulfillment and produce a benefit of 20.8% in power saving.

The results confirm the effectiveness of our methodology in helping service operator's settings the right parameters for better performance and energy efficiency trade-off. Furthermore, this can be used as a dynamic resource provisioning by service providers to dynamically allocate a minimum number of resource required to satisfy a specific

Table 6: Impact of power limit constraint.

Power Limit (%)	SLA Violation (%)	Power Saving (%)
5	46.59	28.95
10	32.79	27.32
15	27.45	26.58
20	24.91	26.18
25	16.73	24.75
30	11.73	23.77
35	11.73	23.77
40	11.73	23.77
45	0	20.81

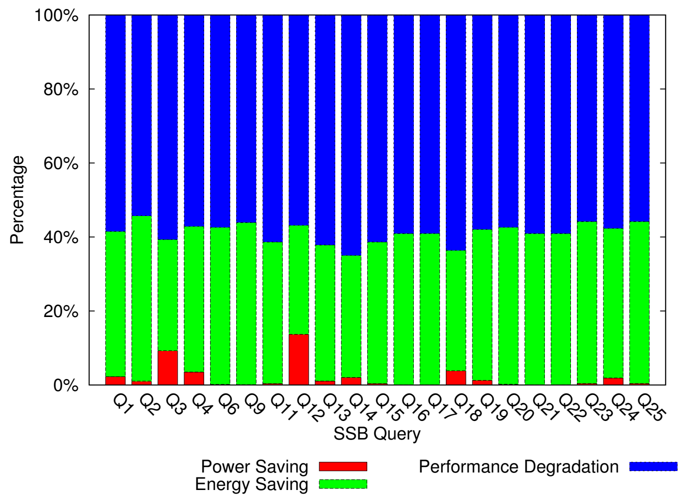


Figure 16: Impact of materialized views configurations on performance and power consumption of SSB queries.

quality of service. Next, we will evaluate the energy saving that the proposed approach can achieve.

5.3.9. Power and Energy Saving

The purpose of this set of experiments is to investigate the benefit of our approach in terms of energy efficiency. We will study two cases regarding power and energy savings: (i) per query and (ii) per workload. In the first case, using our simulator, we run all the 30 OLAP queries of the SSB benchmark against 100GB database size using Time-MV and Power-MV materialized view configurations and we collect the performance and power consumption costs for each configuration. We calculate the power/energy saving and performance degradation of Power-MV compared with Time-MV using the following formulas (i.e., the power saving case) for the query Q_i:

$$PowerSaving_i = \frac{Power_i - Power_{TimeMV}}{Power_{TimeMV}} \times 100 \qquad (11)$$

In Figure 16 we plot saving results as a percentage. As we can see, 21 of 30 queries have the potential for energy saving in the Power-MV configuration. Normally, the benefit of energy conservation for these queries has a negative impact on the processing time as shown in the same figure. However, as we will demonstrate, choosing the trade-off configuration can lead to good energy saving values with less performance degradation. These queries are characterized by an important number of SQL operators, various I/O and CPU operations, and share much common SQL components. This results in a rich $\mathcal{GQG}$ and gives the evolutionary algorithm variety choices of views configurations. Therefore, we can achieve good energy saving queries from those views since the most power intensive components will be materialized. On the other hand, the rest of queries that do not show opportunities for energy saving, are queries with a small number of SQL operators and a limited set of the shared components. This leads the evolutionary algorithm to choose the same views in both Time-MV and Power-MV configurations. We conclude that there exist opportunities for energy conservation at the *query level* using an energy-oriented physical design optimization technique.

In the second case, we move our attention to the *workload level*. Specifically, we create three workloads: (1) WL30 contains 30 query, (2) WL100 contains 100 query, and (3) WL200 contains 200 query. For every query set, we generate their $\mathcal{GQG}$ using our simulator and get the performance and power consumption costs for the three materialized view configurations (Time-MV, Power-MV, Tradeoff-MV) without considering the constraints. Also, we calculate the initial costs of the workload without optimization. We run the experiments under two different database sizes: 10GB and 100GB. We compute the power/energy saving and performance degradation of Power-MV and Tradeoff-MV compared with Time-MV configuration using the Equation 11.

In Table 7 we present the results of the experiments. We can clearly see that the workloads consume significantly lower power when we choose a materialized views configuration that favors low-power views. When we compare the power-only (Power-MV) with the performance-only (Time-MV) results, we observe a significant margin in power savings, ranging from 28% to 47%, the benefit is remarkable when we go from small to big database size and workload queries, this is due to the diversity of the possible views configurations in large $\mathcal{GQG}$ and their impact on performance/power of the systems. Thus, the experiments show comparable levels of power saving. As expected, the savings of the Tradeoff-MV configuration are smaller than those obtained by the power-only experiment, but it still achieves 11 - 38% values. On the other hand, the power-only configuration takes more time to finish executing all the queries, which translates in noticeable performance degradation. This result is not surprising since, if we gain in power, we automatically lose in performance. In the Tradeoff-MV configuration, the performance degradation is actually acceptable if we compare it with the no optimization case (*origin* row in Table 7). We note that

Table 7: Performance degradation and power/energy saving in different materialized views configurations.

DB Size	Work-load	MV Conf	Time (min)	Power (W)	Energy (kJ)	Power Sav (%)	Energy Sav (%)	Perf Deg (%)
10GB	WL30	origin	45.83	15.29	42.04	-	-	-
		time	7.92	20.91	9.93	-	-	-
		power	29.58	15	26.62	28.26	168.02	273.62
		trade-off	13.69	16.75	13.75	19.9	38.48	72.89
	WL100	origin	174.42	18.95	198.33	-	-	-
		time	87.23	23.82	124.66	-	-	-
		power	222.15	16.87	224.85	29.17	80.38	154.66
		trade-off	138.69	19.41	161.53	18.49	29.59	58.99
	WL200	origin	350.86	19.03	400.55	-	-	-
		time	179.78	23.69	255.51	-	-	-
		power	439.01	17.05	449.04	28.03	75.74	144.2
		trade-off	290.25	19.31	336.22	18.5	31.59	61.45
100GB	WL30	origin	676.44	17.45	708.3	-	-	-
		time	190.81	25.61	293.19	-	-	-
		power	1536.84	13.55	1249.14	47.1	326.04	705.43
		trade-off	241.17	22.79	329.8	11	12.49	26.39
	WL100	origin	2225.99	17.27	2306.09	-	-	-
		time	582.04	25.96	906.63	-	-	-
		power	3963.11	13.67	3251.25	47.33	258.61	580.9
		trade-off	1487.87	16.8	1499.61	35.3	65.4	155.63
	WL200	origin	4459.49	17.4	4654.45	-	-	-
		time	1231.82	25.63	1894.56	-	-	-
		power	8977.86	13.66	7358.2	46.71	288.39	628.83
		trade-off	3664.77	15.89	3493.13	38.03	84.38	197.51

the total energy saving of 12% to 84% in the energy-aware research field, are exciting numbers and very promising towards energy-aware DBMSs. Moreover, for the trade-off materialized view configuration, we supposed a weight of $\omega_1 = 0.5$, $\omega_2 = 0.5$ for performance and power objectives function, but in real world database servers, the administrator can adjust these value to obtain the desired energy conservation/performance degradation trade-off.

Summarizing, we believe our approach is worthwhile. With our modest setup, we get encouraging results in saving both power and energy. If we consider a high-performance server of 380 W average power, and always in service, the server could consume for one year about 3283 KWh of energy [56]. The total energy cost for this single server would be 328 US dollars per year (for 10 cents per kWh) ignoring the costs of cooling infrastructure and power supply systems [56]. Using the proposed techniques (12% - 84% energy saving achieved), we could save 39 - 275 US dollars per server a year. This number could increase if we go in large-scale data centers with thousands of servers.

6. Conclusion and Future Work

The rising demand for computational resources by Data Science applications, businesses and Web applications has led to the development of large-scale data centers which consume large amounts of energy. Therefore, the *energy consumption management* has become a crucial issue for hardware vendors and DBMS editors. In the area of DBMS, most of the recent studies on energy efficiency are mainly focused either on *pure hardware techniques* or *query processing and buffer management techniques*.

In this paper, we proposed an initiative to integrate the energy into the physical phase of the life cycle of database design, by considering the case of materialized views – a redundant optimization structure. Before presenting our finding to solve the problem of their selection, we give a deep analysis of the most important state of art covering both hardware and software initiatives. Different scenarios of integrating energy in the process of selecting materialized views are discussed. In this study, we focused on two scenarios: *energy as a constraint* and *energy as a non-functional requirement*. These two scenarios are formalized and solved using genetic algorithms. To evaluate the quality of the final solutions, we developed mathematical cost models to estimate the used objectives: energy consumption and query performance. Machine learning techniques have been used to estimate the parameters of the cost model dedicated to energy consumption. These cost models have been used to evaluate our proposal and their results have been confronted with the those obtained by the real validation. Furthermore, we highlighted the importance of physical database design direction towards energy efficiency DBMSs. The experimental study confirms our claim, since our proposal significantly reduces the overall the workload energy consumption., by achieving active power *savings up to 38% and total energy sav-*

ings up to 84%. Additional, we studied another important factor in today cloud environments which aims at reducing energy consumption while meeting a certain performance target.

This work opens several directions of further research. *From theoretical point of view*, we are: **(i)** integrating network communications in our cost models to have all aspects that may impact the energy, **(ii)** we are exploring other algorithms widely used in the context of materialized view selection such as integer programming approaches [57] and **(iii)** incorporating the energy in the process of selecting the schemes of other optimization structures such as indexes and horizontal partitioning. *From practical perspective*, we are conducting other experiments with very large workload and data set to evaluate this scalability. *From methodological point of view*, we are also planning to investigate the possibility of integrating the energy dimension in earlier database conception phases, such as the conceptual design, trying to check if varying conceptual schema of databases can lead to energy conservation.

References

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

[2] T. N. R. D. Council, Scaling up energy efficiency across the data center industry: Evaluating key drivers and barriers, Issue Paper, `http://www.nrdc.org/energy/files/data-center-efficiency-assessment-IP.pdf` (August 2014 (accessed 06.04.16)).

[3] E. Liebert, Five strategies for cutting data center energy costs through enhanced cooling efficiency, White Paper, `http://www.emersonnetworkpower.com/documentation/en-us/brands/liebert/documents/white%20papers/data-center-energy-efficiency_151-47.pdf` (2007 (accessed 06.04.16)).

[4] D. Tsirogiannis, S. Harizopoulos, M. A. Shah, Analyzing the energy efficiency of a database server, in: sigmod, 2010, pp. 231–242.

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

[6] M. Poess, R. O. Nambiar, Energy cost, the key challenge of today's data centers: a power consumption analysis of tpc-c results, PVLDB 1 (2) (2008) 1229–1240.

[7] Y.-C. Tu, X. Wang, B. Zeng, Z. Xu, A system for energy-efficient data management, ACM SIGMOD Record 43 (1) (2014) 21–26.

[8] R. Agrawal, A. Ailamaki, P. A. Bernstein, E. A. Brewer, M. J. Carey, S. Chaudhuri, A. Doan, D. Florescu, M. J. Franklin, H. Garcia-Molina, et al., The claremont report on database research, ACM SIGMOD Record 37 (3) (2008) 9–19.

[9] D. Abadi, R. Agrawal, A. Ailamaki, M. Balazinska, P. A. Bernstein, M. J. Carey, S. Chaudhuri, J. Dean, A. Doan, M. J. Franklin, et al., The beckman report on database research, Communications of the ACM 59 (2) (2016) 92–99.

[10] Z. Xu, Y.-C. Tu, X. Wang, Dynamic energy estimation of query plans in database systems, in: 33rd International Conference on Distributed Computing Systems (ICDCS), IEEE, 2013, pp. 83–92.

[11] M. Kunjir, P. K. Birwa, J. R. Haritsa, Peak power plays in database engines, in: EDBT, ACM, 2012, pp. 444–455.

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

[13] A. Beloglazov, R. Buyya, Y. C. Lee, A. Zomaya, et al., A taxonomy and survey of energy-efficient data centers and cloud computing systems, Advances in Computers 82 (2) (2011) 47–111.

[14] S. Harizopoulos, M. Shah, J. Meza, P. Ranganathan, Energy efficiency: The new holy grail of data management systems research, arXiv preprint arXiv:0909.1784.

[15] G. Graefe, Database servers tailored to improve energy efficiency, in: Proceedings of the 2008 EDBT workshop on Software engineering for tailor-made data management, ACM, 2008, pp. 24–28.

[16] E. Iman, A. Ashraf, C. Z. Daniel, Z. Calisto, Recommending XML physical designs for XML databases, VLDB Journal 22 (4) (2013) 447–470.

[17] S. Chaudhuri, V. R. Narasayya, Self-tuning database systems: A decade of progress, in: VLDB, 2007, pp. 3–14.

[18] M. Imene, B. Zohra, A survey of view selection methods, SIGMOD Record 41 (1) (2012) 20–29.

[19] H. Gupta, I. S. Mumick, Selection of views to materialize under a maintenance cost constraint, in: ICDT, 1999, pp. 453–470.

[20] A. Zhou, B. Qu, H. Li, S. Zhao, P. N. Suganthan, Q. Zhang, Multiobjective evolutionary algorithms: A survey of the state of the art, Swarm and Evolutionary Computation, Elsevier 1 (1) (2011) 32–49.

[21] K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, A fast and elitist multiobjective genetic algorithm: Nsga-ii, IEEE Transactions on Evolutionary Computation 6 (2) (2002) 182–197.

[22] W. Lang, J. Patel, Towards eco-friendly database management systems, arXiv preprint arXiv:0909.1767.

[23] Z. Xu, X. Wang, Y.-C. Tu, Power-aware throughput control for database management systems., in: ICAC, 2013, pp. 315–324.

[24] Intel, Oracle, Oracle exadata on intel® xeon® processors: Extreme performance for enterprise computing, `https://www.oracle.com/engineered-systems/exadata/index.html` (2014 (accessed 06.04.16)).

[25] L. Woods, Z. István, G. Alonso, Ibex: an intelligent storage engine with support for advanced sql offloading, Proceedings of the VLDB Endowment 7 (11) (2014) 963–974.

[26] M. Rofouei, T. Stathopoulos, S. Ryffel, W. Kaiser, M. Sarrafzadeh, Energy-aware high performance computing with graphic processing units, in: Workshop on power aware computing and system, 2008.

[27] A. Hurson, H. Azad, Energy Efficiency in Data Centers and Clouds, Academic Press, 2016.

[28] J. Do, Y.-S. Kee, et al., Query processing on smart ssds: opportunities and challenges, in: ACM SIGMOD, ACM, 2013, pp. 1221–1230.

[29] P. Behzadnia, Y. Tu, B. Zeng, W. Yuan, Dynamic power-aware disk storage management in database servers, Tech. rep. (2014).

[30] D. Schall, V. Hudlet, T. Härder, Enhancing energy efficiency of database applications using ssds, in: Proceedings of the Third C* Conference on Computer Science and Software Engineering, ACM, 2010, pp. 1–9.

[31] S.-K. Cheong, C. Lim, B.-C. Cho, Database processing performance and energy efficiency evaluation of ddr-ssd and hdd storage system based on the tpc-c, in: Cloud Computing and Social Networking (ICCCSN), 2012 International Conference on, IEEE, 2012, pp. 1–3.

[32] R. Appuswamy, M. Olma, A. Ailamaki, Scaling the memory power wall with dram-aware data management, in: Proceedings of the 11th International Workshop on Data Management on New Hardware, ACM, 2015, p. 3.

[33] M. Korkmaz, A. Karyakin, M. Karsten, K. Salem, Towards dynamic green-sizing for database servers, in: International Workshop on Accelerating Data Management Systems Using Modern Processor and Storage Architectures (ADMS), 2015, pp. 25–36.

[34] A. Hassan, H. Vandierendonck, D. 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, ACM, 2015, p. 1.

[35] Z. Xu, Y.-C. Tu, X. Wang, Exploring power-performance trade-

offs in database systems, in: ICDE, 2010, pp. 485–496.

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

[37] A. Roukh, L. Bellatreche, Eco-processing of olap complex queries, Big Data Analytics and Knowledge Discovery (2015) 229–242.

[38] A. Roukh, Estimating power consumption of batch query workloads, Model and Data Engineering (2015) 198–212.

[39] W. Lang, S. Harizopoulos, J. M. Patel, M. A. Shah, D. Tsirogiannis, Towards energy-efficient database cluster design, Proceedings of the VLDB Endowment 5 (11) (2012) 1684–1695.

[40] I. Psaroudakis, T. Kissinger, D. Porobic, T. Ilsche, E. Liarou, P. Tözün, A. Ailamaki, W. Lehner, Dynamic fine-grained scheduling for energy-efficient main-memory queries, in: Proceedings of the Tenth International Workshop on Data Management on New Hardware, ACM, 2014, p. 1.

[41] W. Kang, S. H. Son, J. A. Stankovic, Power-aware data buffer cache management in real-time embedded databases, in: Embedded and Real-Time Computing Systems and Applications, 2008. RTCSA'08. 14th IEEE International Conference on, IEEE, 2008, pp. 35–44.

[42] J. Wang, L. Feng, W. Xue, Z. Song, A survey on energy-efficient data management, ACM SIGMOD Record 40 (2) (2011) 17–23.

[43] D. Schall, T. Härder, Wattdb-a journey towards energy efficiency, Datenbank-Spektrum 14 (3) (2014) 183–198.

[44] F. Fusco, U. Fischer, V. Lonij, P. Pompey, J.-B. Fiot, B. Chen, Y. Gkoufas, M. Sinn, Data management system for energy analytics and its application to forecasting, in: Joint EDBT/ICDT PhD workshop, 2016.

[45] R. Silipo, P. Winters, Big data, smart energy, and predictive analytics, White Paper, `https://www.knime.org/files/knime_bigdata_energy_timeseries_whitepaper.pdf` (2013 (accessed 06.04.16)).

[46] L. Siksnys, C. Thomsen, T. B. Pedersen, Mirabel dw: Managing complex energy data in a smart grid, in: DaWaK, 2012, pp. 443–457.

[47] M. E. Khalefa, U. Fischer, T. B. Pedersen, W. Lehner, Model-based integration of past & future in timetravel, Proceedings of the VLDB Endowment 5 (12) (2012) 1974–1977.

[48] U. Fischer, L. Dannecker, L. Siksnys, F. Rosenthal, M. Boehm, W. Lehner, Towards integrated data analytics: Time series forecasting in dbms, Datenbank-Spektrum 13 (1) (2013) 45–53.

[49] J. C. McCullough, Y. Agarwal, J. Chandrashekar, S. Kuppuswamy, A. C. Snoeren, R. K. Gupta, Evaluating the effectiveness of model-based power characterization, in: USENIX Annual Technical Conf, 2011.

[50] H. Mistry, P. Roy, S. Sudarshan, K. Ramamritham, Materialized view selection and maintenance using multi-query optimization, in: ACM SIGMOD Record, Vol. 30, ACM, 2001, pp. 307–318.

[51] J. Yang, K. Karlapalem, Q. Li, Algorithms for materialized view design in data warehousing environment, in: VLDB, 1997, pp. 25–29.

[52] A. Boukorca, L. Bellatreche, S.-A. B. Senouci, Z. Faget, Coupling materialized view selection to multi query optimization: Hyper graph approach, International Journal of Data Warehousing and Mining (IJDWM) 11 (2) (2015) 62–84.

[53] P. ONeil, E. ONeil, X. Chen, S. Revilak, The star schema benchmark and augmented fact table indexing, in: Performance evaluation and benchmarking, Springer, 2009, pp. 237–252.

[54] C.-L. Hwang, A. S. M. Masud, Multiple Objective Decision Making - Methods and Applications: A State-of-the-Art Survey, Vol. 164, Springer-Verlag Berlin Heidelberg, 1979.

[55] S. Borzsony, D. Kossmann, K. Stocker, The skyline operator, in: ICDE, 2001, pp. 421–430.

[56] S. Barielle, Calculating tco for energy, IBM Systems Magazine, `http://www.ibmsystemsmag.com/mainframe/Business-Strategy/ROI/energy_estimating/` (November 2011 (accessed 06.04.16)).

[57] Z. A. Talebi, R. Chirkova, Y. Fathi, An integer programming approach for the view and index selection problem, Data Knowl. Eng. 83 (2013) 111–125.

LIAS/ISAE-ENSMA
Poitiers, France

May 9, 2016

Information Systems Journal

Dear Editors and Reviewers,

We are pleased to submit our manuscript entitled "Eco-Physic: Eco-Physical Design Initiative for Very Large Databases". The manuscript is an extension of our conference paper [1] entitled "Eco-DMW: Eco-Design Methodology for Data warehouses". The main extensions with respect to the conference paper version are:

- the presence of a detailed survey of the green initiatives when designing databases. These initiatives cover hardware and software dimensions.
- an in-depth discussion concerning the place of the energy in the formalization of the optimization problem of selecting materialized views;
- a general approach to develop cost models predicting execution time or power consumption of queries;
- adding the view's maintenance cost to the formalization, and resolution of the problem in the case where the energy is considered as a constraint;
- series of well documented experiments using different software and hardware configurations to study their impact on the returned solutions of our proposal.

We confirm that this manuscript has not been published elsewhere and is not under consideration by any other journal. We have no conflicts of interest to declare.

Thank you for your consideration of this manuscript.

Sincerely,

Ladjel Bellatreche, Professor
LIAS/ISAE - ENSMA
1 avenue Clment Ader BP 40109
86961 Chasseneuil, France
+33 (0)549498077
bellatreche@ensma.fr

In the Big Data Era, the management of energy consumption by servers and data centers has become a challenging issue for companies, institutions, and countries. In data-centric applications, Database Management Systems are one of the major energy consumers when executing complex queries involving very large databases. Several initiatives have been proposed to deal with this issue, covering both the hardware and software dimensions. They can be classified in two main driven-hypothesis approaches assuming either that (a) the database is already deployed on a given platform, or (b) it is not yet deployed. In this study, we focus on the first set of initiatives with a particular interest in physical design, where optimization structures (e.g., indexes, materialized views) are selected to satisfy a given set of non-functional requirements such as query performance for a given workload. In this paper, we first propose an initiative, called Eco-Physic, which integrates the energy dimension into the physical design when selecting materialized views, one of the redundant optimization structures. Secondly, we provide a multi-objective formalization of the materialized view selection problem, considering two non-functional requirements: query performance and energy consumption while executing a given workload. Thirdly, an evolutionary algorithm is developed to solve the problem. This algorithm differs from the existing ones by being interactive, so that database administrators can adjust some energy sensitive parameters at the final stage of the algorithm execution according to their specifications. Finally, intensive experiments are conducted using our mathematical cost model and a real device for energy measurements. Results underscore the value of our approach as an effective way to save energy while optimizing queries through materialized views structures.

Highlights

- An energy-efficiency physical design of databases is proposed.
- A multi-objective view selection methodology in which query performance and power consumption costs have to be minimized is formulated.
- An evolutionary algorithm (based on genetic algorithms) to solve the problem is given. It is constrained by the size and maintenance cost of the selected views.
- The results confirm that the proposed approach can significantly reduce the workload energy consumption.
- Active power savings up to 38% and total energy savings up to 84% are achieved.