Energy-efficient Database Systems: A Systematic Survey

BINGLEI GUO*, Hubei University of Arts and Science, School of Computer Engineering, P.R.China
JIONG YU, Xinjiang University, College of Information Science and Engineering, P.R.China
DEXIAN YANG, Xinjiang University, College of Information Science and Engineering, P.R.China
HONGYONG LENG, Beijing Institute of Technology, School of Computer Science & Technology, P.R.China
BIN LIAO, Xinjiang University of Finance and Economics, College of Statistics and Information, P.R.China

Constructing energy-efficient database systems to reduce economic costs and environmental impact has been studied for ten years. With the emergence of the big data age, along with the data-centric and -intensive computing trend, the great amount of energy consumed by database systems has become a major concern in a society that pursues green information technology. However, to the best of our knowledge, despite the importance of this matter in Green IT, there have been few comprehensive or systematic studies conducted in this field. Therefore, the objective of this article is to present a literature survey with breadth and depth on existing energy management techniques for database systems. The existing literature are organized hierarchically with two major branches focusing separately on energy consumption models and energy-saving techniques. Under each branch, we first introduce some basic knowledge, and then we classify, discuss, and compare existing research according to their core ideas, basic approaches, and main characteristics. Finally, based on these observations through our study, we identify multiple open issues and challenges, and provide insights for future research. We hope that our outcome of this work will help researchers to develop more energy-efficient database systems.

CCS Concepts: • **General and reference → Surveys and overviews**; • **Hardware → Power estimation and optimization**; • **Information systems → Data management systems**.

Additional Key Words and Phrases: Green computing, database systems, energy consumption modeling, energy management, energy efficiency, energy proportionality

1 INTRODUCTION

High energy consumption not only brings heavy economic burden to enterprises and government, but also exerts negative impact on environment sustainability [21]. Ever since the Copenhagen conference held in 2009, there has been a global consensus on constructing a sustainable and low-carbon society. For IT systems, the requirement of energy management has already spanned all levels from the hardware, system software, server, and to application. In the current data-centric computing trend, database systems, as fundamental components of

*Corresponding author.
This work is supported by the Scientific Research Project of Education Department of Hubei Province Grant Q20212604.

Authors' addresses: Binglei Guo, bingleiguo@stu.xju.edu.cn, Hubei University of Arts and Science, School of Computer Engineering, Longzhong Rd 296, Xiangyang, Hubei, P.R.China, 441053; Jiong Yu, Xinjiang University, College of Information Science and Engineering, Shengli Rd 666, Urumqi, Xinjiang Uygur Autonomous Region, P.R.China, 830046, yujiong@xju.edu.cn; Dexian Yang, Xinjiang University, College of Information Science and Engineering, Shengli Rd 666, Urumqi, Xinjiang Uygur Autonomous Region, P.R.China, 830046, 107556517108@stu.xju.edu.cn; Hongyong Leng, Beijing Institute of Technology, School of Computer Science & Technology, South Street 5, Zhongguancun, Haidian District, Beijing, P.R.China, 100081, leng@bit.edu.cn; Bin Liao, Xinjiang University of Finance and Economics, College of Statistics and Information, Beijing Middle Rd 449, Urumqi, Xinjiang Uygur Autonomous Region, P.R.China, 830012.

0360-0300/2022/6-ART $15.00
https://doi.org/10.1145/3538225

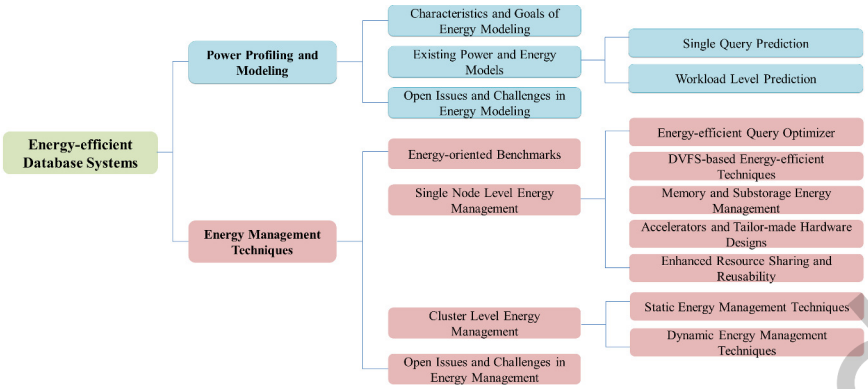

Fig. 1. An overview of the proposed taxonomy on energy consumption models and energy management techniques for green database systems.

almost every service, have been one of the main energy-consuming components in IT systems, especially for nowadays large scale data management systems (generally a datacenter). And recently, more and more researchers propose to build energy-efficient database systems to reduce energy consumption and improve energy efficiency. Constructing green database systems has become a research hotspot and has drawn increasingly attention from both the industry and academic. Compared with traditional performance-driven database designs, the main optimization and design goal of energy-efficient database systems is minimizing energy cost without sacrificing scalability as well as maintaining acceptable levels of performance required by the users [1]. In addition, green database systems not only take energy efficiency but also energy proportionality into consideration (detailed in Section 5.1). To realize such systems, the DBMSs (database management systems), i.e., the software, are generally taken as the prime driver and manager of the overall system energy consumption. As a result, being aware of energy cost of database servers as well as the workload running on them, database systems can integrate energy-efficient software and hardware techniques to achieve significant energy conservations.

There are three main contributions of this study. Firstly, an in-depth investigation on existing works that focused on constructing green database systems is conducted, which is one of the major contributions of this paper. Secondly, while there are many energy management techniques for database systems, these techniques are at different levels and largely unorganized. We provide a detailed taxonomy of the state of the art techniques that have been so far used for modeling energy consumption and saving energy in database systems. Finally, some challenges and open issues that worth pursuing and studying in future research are also presented. We believe that the comprehensive references, organization, and approach of this survey will make it a valuable resource for readers who are seeking for in-depth information and/or a comprehensible introduction into green database systems. The organization of our survey is illustrated in Figure 1.

The remainder of this paper is organized as follows. Section 2 first introduces related survey and compares them with our study. Section 3 reports how this study is performed step by step in detail. Section 4 describes the needs, characteristics, and techniques of energy profiling and modeling in database systems. Section 5 provides basic concepts, required background, and energy-efficient techniques for database systems. Section 4 and Section 5 constitute the main body of this survey. In Section 4 and Section 5, techniques are going to be classified based on their core ideas and approaches, as well as described, compared, and discussed. Moreover, in each section, we present guidelines on key issues and challenges of energy modeling and management in database systems to ease future researchers. Finally, Section 6 concludes the study.

2 RELATED SURVEYS

Energy-efficient database systems have been playing a vital role in realizing environment friendly IT systems and sustainable society. As a foundation of the Green IT, energy-efficient database systems are not limited to a certain domain or field but have been related to energy management issues at different aspects and levels including servers, energy benchmarks, network infrastructures, mobile devices and embedded systems, storage systems, data centers, and cloud computing. Related surveys published till present is shown chronologically in Table 1. These surveys are all concerned with energy issues in IT systems addressed at various levels. Although most of them can be served as general guidelines and provide insights for building green databases, they are not specially focused on energy modeling and management techniques in the context of database systems. The motivation and aim of this survey is to provide a comprehensive review and comparison of energy-related techniques for green database systems, covering their development from their inception to the current state and beyond. To this end, this survey is conducted based on a well-defined and widely known systematic literature review (SLR) method proposed by Kitchenham et al. [96]. SLR is recognized as an evidence-based method used for identifying, selecting, and critically evaluating all accessible relevant evidence to unbiasedly answer research questions focused on a topic. Since SLR is developed to be used repeatedly, we provide more details of our SLR in Section 3 for others to broaden the scope of this survey in future.

One of the most important studies on green database systems is presented by Graefe [56] from HP Labs. In his study, promising methods and techniques, such as additional layer for memory hierarchy, I/O optimization, data format, adaptive plan execution, and compression techniques, are analyzed. Also, the author discusses energy-efficient techniques at the software-level on the basis of five categories: scheduling, update techniques, I/O scheduling, and physical database designs. This classification would help any researcher understand concepts that are related to green database servers.

Harizopoulos et al. [73] discuss approaches for reducing energy consumption in database systems from three major aspects: energy-aware optimization, resource use consolidation, and redesign for maximum energy efficiency. This study points out and clearly discusses promising areas for green data management systems, including (a) data placement and query processing, (b) resource manager, and (c) new system architectures.

Graefe [56] and Harizopoulos et al. [73] both provide detailed discussion and insights on open issues and future directions of green databases. However, focusing on identifying the research trends rather than evaluating techniques, only a few research papers that addressed at building green databases are included in their studies. This is because, at that time, the area of green database systems was relatively new. And in the meanwhile, researchers who were interested in this new research field were just getting started to investigate the wide range of probably interesting problems that require extensive research works and innovations in the future.

Wang et al. [191] provide a review on energy-efficient data management methods that focused on power modeling and energy-related benchmarks. The authors discuss open issues of green database systems and present a detailed solution guideline on building such systems. However, only a limited number of references are included and investigated in their survey. Also, it lacks comprehensive study on subsequently published papers from 2011 till present.

You et al. [209] conduct a survey on energy-saving data management techniques for cloud-related systems from three aspects including data classification, data replication, and data placement strategies. Targeting at the data management level, their study and observations offer new space and opportunities for improving energy efficiency of the cloud data centers, especially with respect to data-intensive applications.

Lin et al. [114] present an overview on power models of servers in cloud data center environments, covering the entire hierarchy including the underlying hardware component level, the VM and Container instances of the virtualization level, and the upper application level. The researchers also provide a detailed taxonomy on

Table 1. Comparison of related surveys

Year	Investigators	Area of focus
2000	Benini et al. [13]	Dynamic reconfiguration techniques at system level for energy conservation
2003	Lefurgy et al. [109]	Energy management techniques and challenges for high-end servers
2008	Graefe [56]	Energy-efficient opportunities in database servers
2009	Harizopoulos et al. [73]	Software-level energy efficiency of data management systems
2010	Poess et al. [145]	Energy benchmarks for measuring energy efficiency
2010	Berl et al. [14]	Energy-efficient computer hardware and network infrastructure in cloud computing
2011	Wang et al. [191]	Techniques of energy conservation for data management
2011	Beloglazov et al. [12]	Energy-efficient designs of computer system for data center and the cloud
2012	Shuja et al. [174]	Data center's energy efficiency
2013	Valentini et al. [188]	Techniques of power management for building energy-efficient cluster computing
2012	Vallina-Rodriguez and Crowcroft [189]	Energy-saving techniques for mobile handsets
2013	Bostoen et al. [20]	Techniques to save energy for storage systems in data center
2014	Mittal [129]	Energy-efficient techniques in embedded systems
2014	Orgerie et al. [137]	Energy-efficient techniques for computing and network systems
2014	Kong and Liu [98]	Power management techniques that explicitly address carbon emission or renewable energy integration of data center
2015	Hoque et al. [81]	Power measurement and modeling techniques of mobile devices
2015	Dayarathna et al. [36]	Techniques of energy modeling for data center and its components
2017	Dabaghi et al. [33]	Energy-saving techniques that minimize energy cost of network components
2020	You et al. [209]	Energy-efficient data management techniques for cloud environments
2020	Lin et al. [114]	Power models and modeling methods of servers in the cloud
2020	Cong et al. [32]	Energy-efficient techniques for mobile devices in fog/edge computing environments

the methods of power data collection and discuss several open challenges regarding power estimation in GPU computing, new virtualized environments, and IoT services.

You et al. [209] and Lin et al. [114] perform their study with a focus on energy-saving techniques and power models for cloud-related environments, respectively. Although their research shed some light on the energy consumption issues of large-scale database deployments in the aspect of database applications and database servers, neither of them are specifically carried out for database systems. A comprehensive study covering both the energy consumption models and the energy management techniques for realizing energy-efficient database systems, as well as attempting to provide a holistic guideline with a fine-grained taxonomy, is still missing from the current literature, which is exactly the motivation and aim of our survey.

By following the SLR in Section 3, we found that there has been numerous research works on building green database systems since its inception. However, relatively few surveys have been recently conducted in this area. And we were also surprised that there is an apparent lack of SLR in energy-efficient database systems especially after 2011. Therefore, after a decade of development, there is an urgent need to provide a systematic literature review to show a clear picture of the past efforts and to propose future research trends for green database systems.

Addressing the main limitations of existing studies [56, 73, 114, 191, 209], we present an in-depth overview on green database systems by studying the existing literature published over the period 2007-2021. Moreover, this study also fills an important gap by identifying current issues and challenges for future researchers as well as providing a more detailed taxonomy of existing related research works as shown in Figure 1. And to the best of our knowledge, this study is among the first papers, currently absent from the literature, that tried to provide a comprehensive study on green database systems in a systematic way introduced by Kitchenham et al. [96]. This survey is written in a comprehensible style to provide readers (whether they are professional or outside the specialty of the article) with selected bibliography and appropriate background to help them become familiar with and learn something specific about the chosen topic. We believe that this hieratical way that our work follows and the thorough research literature that covers different aspects of energy management for database systems will make this overview a unique contribution to the green computing and the database research communities.

Table 2. List of keywords used in the search process

	Set 1	Set 2
Term 1	Database [T1, S1]	Energy-efficient [T1, S2]
Term 2	Database System [T2, S1]	Energy Efficiency [T2, S2]
Term 3	Database Management System [T3, S1]	Energy-aware [T3, S2]
Term 4	DBMS [T4, S1]	Energy [T4, S2]
Term 5		Power [T5, S2]

3 METHODOLOGY AND TAXONOMY OF LITERATURE REVIEW

In this section, we present the methodology and procedure used for conducting the survey based on general SLR (systematic literature review) guidelines proposed by Kitchenham et al. [96]. This phase is planned to report how this survey is conducted, including the following steps: the formulation of research questions, the process of searching studies, the evaluation criteria for including and excluding studies, and finally the results and how we built our taxonomy.

3.1 Research Questions

In order to identify state-of-the-art solutions and open challenges of building energy-efficient database systems, three research questions (RQs) are formulated to be answered as the main focus of our study as following.

- RQ1: What articles in green database systems are published?
- RQ2: What specific research problems are addressed by these articles in order to build green database systems?
- RQ3: What similarities and differences are among solutions of these articles?

The aim of RQ1 is to locate and identify all the relevant research articles that fall into the scope of this survey. And to answer this question we need to provide an approach to search the articles as well as a detailed criterion to include and exclude them for further analysis. The result of RQ1 is the basis for the next two research questions and guarantees the quality and reliability of this study. The aim of RQ2 is to identify the issues that the selected articles are trying to solve. The aim of RQ3 is to clarify and understand how the solutions of these studies are proposed to enable and enhance energy-efficient database systems. RQ1, RQ2, and RQ3 summarize the main purpose of this study. And the results of these questions also constitute the main body of our study.

3.2 Search Process and Evaluation Criteria of Studies

To describe our article search strategy, a set of keywords and sources were used in this phase. First, to locate relevant studies in the reference list, we employed the following four online literature retrieval services: Science Direct, ACM Digital Library, Web of Science, and Google Scholar. Second, to construct the search string, two sets of keywords were used as shown in Table 2. Each set contains a set of keywords with the same meaning or similar implication. Finally, the search string can be expressed as follows: (([T1, S1] or [T2, S1] or [T3, S1] or [T4, S1]) and ([T1, S2] or [T2, S2] or [T3, S2] or [T4, S2] or [T5, S2])). We manually performed the search string in each of the sources mentioned-above. Note that, to minimize the possibility of eliminating relevant articles, we also consider the references in the reference list of the selected studies, especially those that were cited and discussed in the related work section.

The inclusion criteria that we used to select and evaluate studies are described as follows:

- They are original academic research articles.

- Their focus of interests, research questions, and goals are clearly stated in the abstract, introduction, and conclusion sections.
- Their ideas are realized, and the experimental processes are detailed and possible for others to follow and repeat.

With respect to the exclusion criteria, an article will be discarded if it fails to meet any of the inclusion criteria or if we cannot get access to its full text. Note that, we also included studies, of which the energy-saving results are achieved as a bonus for improving performance requirements. We believe that these studies may help to shed light on exploring the relationship between performance and energy consumption in database systems.

3.3 Results and Taxonomy of Article

With respect to RQ1, by following the above steps, 135 studies were identified from 2007-2018. And during the revision of this paper, the number of finally selected studies was updated to 155 from 2007-2021. Note that, compared with previous surveys, we consider 136 more papers. We believe this is mainly due to the following two reasons: 1) the rising interest and need in building energy-efficient database systems over time and 2) the methodology that we employed to conduct this survey made it less likely to miss relevant articles. Therefore, based on the above-mentioned steps defined in our SLR as well as the rich content of related studies, the taxonomy in Figure 1 is finally built, distinguishing this survey from previous ones. We believe that this easy-to-follow, detailed, and hierarchical classification will make it more convenient for researchers to find their desired materials for future research.

To address RQ2, we found that most of the studies are either focused on the issue of power and energy models or the issue of energy-efficient techniques. As far as we are concerned, we believe that this is because, to realize green database systems, two critical aspects of effort are both required. Power and energy models aim to accurately quantify the energy consumption of database operations, while energy-efficient techniques aim to efficiently regulate and manage energy consumption to save energy and improve energy efficiency. The two aspects are complementary efforts that aim to provide one whole solution. Therefore, we naturally divided the main body of the survey into two major parts (i.e. Section 4 and Section 5), corresponding to the blue and pink branches in Figure 1 and focusing separately on energy consumption models and energy-saving techniques. Note that a few of these articles tried to target on tackling the both issues and used energy models as a basis of developing their energy-saving techniques. We discussed them with different emphasis under each branch of the main body in this survey.

In order to answer RQ3, a further classification of the existing works is required based on the characteristics extracted after reading all content of these articles. With respect to the energy modeling efforts in Section 4, we grouped the related works into two subcategories (as shown in Section 4.3) based on the types of their application scenarios where 1) queries are sequentially processed (Section 4.3.1) or 2) queries are concurrently executed as a whole workload (Section 4.3.2). And with respect to the energy management efforts in Section 5, the related studies were classified into two subcategories according to the types of their target environments in which energy management techniques were developed for 1) standalone database server environments (Section 5.4) or 2) the cluster environments (Section 5.5). Moreover, we also need to further identify the differences among these articles based on their core ideas and basic approaches, as well as discuss their major advantages and disadvantages. By doing so, we can also identify open challenges or the least reported issues that are related to energy-efficient database systems in Section 4.4 and Section 5.6 separately under each of the two major branches of this literature review, which finally leads to a detailed taxonomy as shown in Figure 1.

4 POWER PROFILING AND MODELING

Researchers, developers as well as administrators need to know how their choices of designs and polices may affect the energy cost of the whole system and its various components. Therefore, appropriate modifications and energy-efficient strategies can be taken timely. Unfortunately, building physical prototypes to assess every design is costly, and hence not always feasible.

Power profiling is of relevance to various energy management issues. Firstly, a systematic power profiling can directly provide an overview of the power consumed by the entire system. This elementary profiling can be easily completed by using an external instrument. For example, a WT3000 power meter produced by YOKOGAWA (as in https://tmi.yokogawa.com/cn/) can be used to measure and record the power consumption of the entire system under test. Since one of the important goals of power profiling is to understand how the total power cost is distributed and consumed among various hardware components, power profiling is required when building and selecting an appropriate energy model for a specific system. Moreover, power profiling can be used to provide detailed descriptions of power usage patterns for different hardware components and various applications [196]. Such profiling can be partially solved by using power sensors that designed and integrated within hardware components, or special measurement devices that are able to be embedded into the hardware platform. Normally, it requires opening the case of the system and attaching the measurement units to appropriate places, which is sometimes very difficult.

Physical measurements are straight forward and accurate. However, they cannot predict energy consumption demand in the future. Also, it fails to provide an effective correlation between resource utilization and energy consumption. Note that the correlation is necessary for fine-grained energy management strategies. To overcome this challenge, an energy model is needed, whether our goal is to enhance the design of the system and its components, or to support real-time scheduling decisions.

4.1 Energy and Power

In a database server, for a given workload, energy (E) is the power (P) consumed for running the workload over a time period (T). Generally, if the power is constant over the time interval then the energy can be calculated simply as:

$$E = P \times T \tag{1}$$

P is usually denoted as the average power during work processing and is measured in Watt. T is a period of time that is typically measured in second. And E is the energy cost measured in Joule. That is $1Joule = 1Watt \times 1Second$. A more standard way of calculating energy consumption can be enhanced as follows:

$$E(w, T) = \int_0^T P(w, t)dt \tag{2}$$

In most cases, for general workload, its instantaneous power is not always constant but changes over the time interval. Thus, the total energy consumed can be related to the power by integrating over time ([0, T]). The function $P(w, t)$ indicates the power consumed by the workload w during a given time t, where $t = 0$ denotes the beginning of the workload. Also, this equation is not limited to execution time of the workload. The function will return the idle power of the server after the work done.

4.2 Characteristics and Goals of Energy Modeling

An ideal energy model that is suitable for real-time energy management in database systems would meet several design criteria [103, 150]. We give a summary of them with detailed description as follows.

- **Accurate:** It defines how accurate the estimated power is compared with the actual measured power. For some energy-saving strategies, the accuracy is the foremost requirement. For instance, fine-grained power

and performance trade-off depends on accurate power estimation. Otherwise, the result is controversial and meaningless.

- **Fast:** The model can generate predictions quickly. The time cost for every prediction is small enough to enable real-time prediction and optimization.
- **Simple and practical:** The number of inputs and complexity of the model should be as simple as possible, while still generating accurate predictions. If the model is complicated and computationally expensive, the overhead of using it will be high, and thus not practical.
- **Portable and elastic:** The model can be applicable across various DBMSs and machines regardless of the database engine or the underlying hardware. To adapt more complicated environment, the model can be easily extended or merged with other energy models at different levels.
- **Non-intrusive:** Inputs collection should not require intrusive modifications to hardware and/or software of the system. It is hard to forecast and quantify the impact of prospective modifications on system performance as well as hardware components. Also, the complexity of DBMSs makes it difficult to add new features or modify existing features [118].
- **Full-system:** The model can be used to correlate usage information to energy consumption of the entire system, not just individual components.

Simultaneously achieving the above properties is unreasonable and unaffordable, since some of these features are mutually exclusive. For example, complex models are relatively more accurate while more expensive. Therefore, balancing these sometimes-conflicting requirements can be very difficult. Researchers have to wisely decide their key needs. And the last but not the least, a desirable model is the one that allows you to understand energy usage of a system with minimal cost and effort [173].

4.3 Existing Power and Energy Models

Generally, performance, in term of response time to query processing or throughput to transactions, has been taken as the primary goal for traditional database design. While in energy-efficient database systems, minimizing energy consumption must be considered as a first-class optimization goal. In other word, energy should be managed as an important resource. Therefore, quantifying energy consumption of database operations becomes a basic precondition for designing such systems.

Database query languages provide a high-level "declarative" interface to access data that is stored in databases. Over time, SQL has emerged as the standard for relational query language [29] and has been the most primary and common operations in DBMSs. Typically, energy modeling of queries serves two purposes. Firstly, an energy consumption modeling allows the query optimizer to select query plans with lower power/energy cost for query processing, just like a traditional time cost model that is used for selecting faster plans [205]. Secondly, being aware of accurate energy consumption of each query will help to determine the total energy consumption for the entire workload [206].

In this subsection, we survey the existing literature on a specific category of power and energy models for database applications. For that, we investigated a number of journals and conference proceedings to provide a new classification of them. Since these models are developed to predict energy consumption of common database operations that return SQL result sets, we have classified the existing models into two subcategories: 1) single query level, in which queries are executed sequentially; 2) workload level, in which multiple transactions including queries are executed concurrently.

4.3.1 Single Query Prediction. The traditional goal of query optimization in database systems is to improve performance, which means running the query as fast as possible. However, query statement itself does not specify how to access and manipulate data to get desired results. For a given query, a number of query plans can be generated for a DBMS to follow to process the query and create its answer. All the plans of the query are all

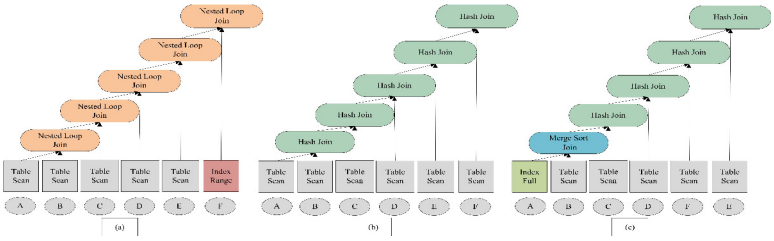

Fig. 2. Three query plans that created for Q5 of TPC-H benchmark.

created equally in their output, but they vary in their response time costs. In other words, a query's performance (i.e., response time) is determined by the final-selected query plan that is used to execute the query. Typically, the query optimizer of a DBMS is responsible for generating alternative plans and making the decision on which plan to execute (i.e., query optimization). Therefore, an important motivation behind quantifying energy cost of queries is to enable energy-aware query optimization.

Generally, a plan can be presented as a physical operator tree that represents a unique execution path of a specific query statement (as shown in Figure 2). The edge in the tree represents the data flow among physical operators. The time cost of each plan can be calculated from a series of parameters denoting resource (i.e., CPU and I/O) holding time of different basic operations. Among various types of DBMSs, there exists different ways to quantify and mapping CPU and I/O cost to response time cost of a certain query plan. For example, in PostgreSQL (a popular open source RDBMS) a parameter, denoting CPU time per tuple, is used to calculate the total CPU time for processing all the tuples that required by a physical operator. In addition, for commercial RDBMSs like Oracle databases, the parameter CPUSPEED, denoting instructions executed per second, is used to calculate the total CPU time for executing a required number of instructions that needed for processing a certain operator.

Since the processing rate of CPU stretches far beyond that of the I/O, a plan's time cost is primarily the processing time of I/O operations, which often makes the CPU time negligible when compared with the I/O time. Conventional intuition about the performance bottleneck in DBMSs has identified the disk as the most important component. Consequently, a plan with minimum I/O cost is usually selected for query processing. However, in the paradigm of green computing, the weight of CPU is greater than that of the storage system in term of their power consumption [144]. This is due to the range of dynamic power consumption of CPUs is wider than that of disks [8]. Therefore, CPUs' power consumption should be paid more attention when it comes to building accurate power models for database systems.

Xu et al. [203] utilize the multiple regression to identify a linear relationship between power consumption of a plan and its basic operations. The basic operations included are tuples or indexed tuples processed by CPU, and pages retrieved from disks. Based on this liner relationship, static power profiles for a number of physical operators are built. And finally, these profiles are used to estimate the power cost of the entire query plan. Static power model of Xu et al. [203] is an off-line approach, thus cannot provide sufficiently accurate prediction under significant workload variation or unpredictable changes of DBMS state. A supplementary work of Xu [201] presents an online power model with high accuracy using RLS (recursive least square) estimator with directional forgetting [117]. This on-line model is able to periodically adjust parameters at runtime with the help of a feedback control mechanism.

Guo et al. [62] propose a static model for query processing by mapping hardware resource consumption (i.e., CPU and I/O) to power cost. Basic operations included in the proposed model are instructions processed by CPU and blocks accessed from disk. Similarly, Flach [49] presents energy models for scan operators including

sequential scan and bitmap heap scan. Basic operations involved in the energy models are fetching and processing index page, fetching and processing tuple page.

In a slightly different power model that presented by Liu et al. [116], the product of the number of retrieved columns and the number of processed tuples are used to calculate the total power consumption of CPU for query processing. Also, the total power cost of disk for query processing is calculated by the number of accessed pages.

In two follow-up works (Xu et al. [205, 206]) , a more comprehensive energy model is developed. Other than query, update, insert, and delete operations are also investigated. The difference between their previous works [201, 203] and the latter is that the dynamic power cost of CPU is not in a linear relationship with the number of tuples processed, namely power consumption does not increase continuously, and it will level out after processing a certain number of tuples. To further identify their relationship, system identification experiment is conducted to generate a regression curve in which energy cost of CPU first grows quadratically with the total number of processed tuples, and then reaches a "hockey point". And after this point, it exhibits a linear relationship between query size and energy consumption.

Based on pipelined segmentations of query plans, a cost model for predicting average power consumption of single complex OLAP queries is provided by Roukh and Bellatreche [153]. However, different from models that derived from linear regression, a polynomial regression method is used to propose the cost model. Also, the authors suggest that parameters which are related to resource consumption (i.e., CPU and I/O) of a pipeline are not in a simple linear relationship with energy when treating the plan as a tree of pipelines.

In aforementioned models, only CPU and disk have been taken into consideration of energy modeling, while ignoring memory and taking it as a constant in term of power. In addition, previous works usually duplicate a database into several copies to ensure each query will be run on a new copy of the database. To further avoid the effect of cache, they sometimes flush the cache periodically to lower the impact of cached data on subsequent queries. High cache hit rate will reduce response time as well as power cost for the same or similar query in succession, since data might be fully or partly loaded into memory. To be more specific, high cache hit rate may produce misleading data that stop researchers from building accurate energy models. However, in real word, high cache hit rate is generally considered as a good feature of a runtime database system. Consequently, it is not reasonable and realistic to eliminate the effect of cache when it comes to building a practical energy model for queries. Additionally, memory has become a significant energy-consuming component in modern database servers, especially those consisting of large amount of memory, for example main-memory databases. Therefore, to build an accurate, comprehensive, and practical energy model, memory cannot be neglected and should be paid more attention [4, 35, 128].

Lang et al. [103] propose a simple and practical energy model for queries based on hardware abstraction in term of CPU, disk, and memory. The proposed energy model is based on the original time cost model in query optimizer. Their model requires four parameters that are directly related to basic operations during query processing. Basic operations included in the proposed model are the number of CPU instructions, The number of read and/or write requests on disk, and the number of memory accesses. However, although energy cost of memory is included, the negative effect of high cache rate on model accuracy and practicality has not been considered.

Guo et al. [63, 64] present a comprehensive energy model using fine-grained basic operations that reflect resource consumption of CPU, disk, and memory. By taking the effect of cache into consideration, the authors present a study of the impact of three main cache structures and memory size on various costs of queries, in term of their power, energy, and time costs. The proposed model is suitable for scenarios when data is partially cached in the memory. Also, by distinguishing data sources (i.e., data are accessed from disk or data are already cached in memory), the energy prediction accuracy of reading data for query processing is improved.

4.3.2 Workload Level Prediction. The second category of power/energy models for queries has been identified by considering real-world scenarios. The above-mentioned models are all for a single query on standalone servers.

In other words, a query monopolizes system resources or queries are executed sequentially. However, in practice, only a single query is processing in a commercial database server, which is rarely the case. Therefore, it is an interesting and realistic problem to study how an energy model for single query could be extended to accurately calculate concurrent queries in standalone servers or scale-out deployments.

Poess and Nambiar [144] propose a workload-level power model to estimate peak power consumption for large-scale database systems under full load. To avoid complexity introduced by the three-tier architecture and the difficulty of measuring power consumed by individual components, the proposed model is based on readily available data from full disclosure report of the published TPC-C benchmark. Power cost of CPU, memory, disk, disk enclosures, and server chassis are included in the proposed model. Also, accuracy and efficiency of the model is verified by TPC-C benchmark publications.

Meza et al. [126] develop a simple but useful model for predicating power cost of decision supporting systems running TPC-H benchmark at peak performance. The model is derived from component variation experiments by physically removing components including CPUs, DIMMs, and disks. And it is suitable for systems, in which disks consume the largest power cost and system components offer little dynamic power range. The experimental results demonstrate that the most energy-saving system configuration is depending on the workload.

Rodriguez-Martinez et al. [151] propose an energy model using multiple-linear regression with readily available statistics including the number of columns, cardinality, average tuple length, constants regarding CPU and disk speed, and memory size. These statistics can be directly extracted from DBMSs by certain SQL trace events. The proposed model can be used to predict both average and peak power for a set of concurrent queries in a parallel or distributed environment. However, the model is only for selection queries, thus needs to be extended to more complex queries and workloads.

Kunjir et al. [101] describe an approach of modeling peak power cost for both multiple queries and a single query. To predict peak power for query plans, a pipeline-based model is developed. The model is motivated by a series of empirical observations on pipelined plans, in which power cost of each operator is determined by the highest rate at which data are funneling into the pipeline by its upstream operators. Note that a "pipeline" is denoted as a sequence of concurrently executing operators. However, to model the peak power behavior of a pipeline, an impractical number of training instances are going to be computed, which may cost months in term of time.

Similarly, Roukh [152] introduces a pipeline-level modeling method to predict energy cost for a batch workload composed of multiple concurrently running queries. Since query plans can be segmented into sets of pipelines, a query workload can be taken as a number of phases of mixed pipelines, which means that the execution of the workload can be treated as a sequence of pipeline mixes. And by using the original cost model of query optimizer, CPU and I/O resource consumption that required by each pipeline included in the query workload can be estimated. Also, to obtain energy consumption behaviors of concurrent queries, statistical models are built with observed energy cost of pipelines and multivariate regression. Finally, a combination of the proposed models is used to estimate the entire energy cost of concurrent queries.

We describe the research conducted on modeling power/energy cost for database operations up to this point. All the models described above are developed in the context of relational DBMSs, and tested with well-known benchmarks. Most of these models are based on multiple-linear regression, while power model proposed by Roukh and Bellatreche [153], and Roukh [152] is an example for multiple polynomial regression. In addition, most models are all developed in a standalone server, and thus cannot be directly used for large-scale database systems. Note that models proposed in [144, 151] are examples for multi-server environments. Also, most publications have made energy estimation with enough accuracy, while only a few of them have made comparisons with other models at a same level.

We give a summary of them in Table 3. And in Section 4.4, we provide open issues and challenges that can be addressed in future papers.

Table 3. A summary of power/energy models

Authors	Focus	Level	Platform	Components	Workloads	Accuracy
Poess and Nambiar [144]	Peak Power	Workload	OLTP system	System	TPC-C	10%∼25% for workload
Meza et al. [126]	Peak Power	Workload	Commercial DBMS	CPU, Disk, Memory	TPC-H	Less than 3%
Xu et al. [203]	Average Power	Single Query	PostgreSQL DBMS	CPU, Disk	TPC-C, TPC-H	7.2% to 14.5% for individual operators
Xu [201]	Average Power	Single Query	PostgreSQL DBMS	CPU, Disk	TPC-C, TPC-H	As low as 7.2% for query
Flach [49]	Average Power	Single Query	PostgreSQL DBMS	CPU, Disk	TPC-H	N/A
Rodriguez-Martinez et al. [151]	Peak Power	Workload	PostgreSQL DBMS	CPU, Disk, Memory	TPC-H	N/A
Lang et al. [103]	Average Power	Single Query	Commercial DBMS	CPU, Disk	SPEC, TPC-E	3% to 8% for query
Kunjir et al. [101]	Peak Power	Both	Commercial DBMS	CPU, Disk	TPC-H, TPC-DS	Varies from 3% to 15%
Liu et al. [116]	Average Power	Single Query	PostgreSQL DBMS	CPU, Disk	TPC-H	Varies from 5.2% to 12.1%
Xu et al. [205, 206]	Energy	Single Query	PostgreSQL DBMS	CPU, Disk	TPC-H	10% to 13.7% for query
Roukh and Bellatreche [153]	Average Power	Single Query	Oracle DBMS	CPU, Disk	TPC-H, TPC-DS	Varies from 4% to 9%
Roukh [152]	Average Power	Workload	Oracle DBMS	CPU, Disk	TPC-H, TPC-DS	Varies from 6% to 10.5%
Guo et al. [62]	Average Power	Single Query	Oracle DBMS	CPU, Disk	N/A	6% to 9% for query
Guo et al. [63, 64]	Energy	Single Query	Oracle DBMS	CPU, Disk, Memory	TPC-C, TPC-H	Higher accuracy compared with [206]

4.4 Open Issues and Challenges in Energy Modeling

In this section, based on reviewed literature, we present several important open issues and challenges that need to be addressed by future researchers in modeling power/energy consumption for database systems.

4.4.1 Average Power vs. Peak Power. Most energy models that we have reviewed in Section 4.3 are built on top of the general formula that the total energy consumption is a product of the total time and average power. Therefore, these models can be directly used to predict average power cost. Prediction of average power cost will affect energy-related expenses in a long-term, which is also the concern of designing heat dissipation systems. However, there has been relatively fewer works for peak power prediction. Only four models described above are developed for peak power prediction [101, 126, 144, 151].

Building peak power models will benefit energy management in several ways. First, modeling peak power is associated with risk management for unexpected system failure due to a sharp increase in power and heat (i.e., overheating surges). Second, peak power prediction is more concerned with server design, capacity planning. Also, peak power provision has been playing a decisive role in the cost of building and maintaining a large-scale datacenter. Additionally, since thermal constraints limit further improvement of CPU performance, capping peak power consumption will alleviate power and cooling limitations [47], as well as minimize their negative impact on server performance. Compared with average power models, researchers are confronted with challenges of building peak power models. Specifically, building accurate peak power model needs to take parallelism and interaction of operators into consideration, since peak power is usually a consequence of maximum aggregate of concurrent operators. In other words, the model should be able to identify bursty or short-term phenomena during query processing.

4.4.2 Single Query vs. Concurrent Queries. Most power/energy models in Section 4.3 are aimed at predicting power/energy cost for a single query. Generally, the cost of a certain query can be represented by a plan that is used to execute the query. And the cost of the plan can be further denoted as the summation of cost drawn by individual operators that included in the plan. Researchers follow the intuition that building power models for operators first and then use these operator-level models to give a whole picture of the power consumed by the plan. However, by now only a limited number of operators have been considered. More efforts are required to explore wider range of plan operators.

Query level models are mostly inspired by original time cost models in different types of relational DBMSs. For example, power models developed by Xu et al. [205, 206] are based on the time cost model in PostgreSQL, while other power models developed by Guo et al. [63, 64] and Lang et al. [103] are extended on the time cost model in Oracle DBMSs. Also, the verification process of these models is implemented in corresponding DBMSs. As a consequence, the portability of these models is a big problem. Therefore, extensible studies are needed to evaluate and improve models' flexibility, portability and accuracy across different database systems and applications.

For concurrently running queries, energy modeling plays a vital role in both query execution and scheduling at a workload level. Also, peak power prediction is associated with concurrent running operators on modern multi-core processers. However, for now, only five works are related with this scenario [101, 126, 144, 151, 152]. Although they all come up with methods of modeling energy and give suggestions on building green databases, they have some limitations. For example, the model offered by Poess and Nambiar [144] is based on history data from published benchmarks, therefore it cannot adapt to future diversity and changes in database architecture and workload. Model proposed by Meza et al. [126] is only suitable for database systems that have little dynamic power range. Model proposed by Rodriguez-Martinez et al. [151] is only for selection queries, thus future efforts on more complex queries that explore different types of operators is needed. Models proposed by Kunjir et al. [101] and Roukh [152] both need a lot of pre-works to compute and to model a huge number of distinct pipelines. Additionally, blocking operators cannot be pipelined, since they will need their entire input before producing any output.

Models for standalone query help to identify the power-saving opportunity of selecting lower-power plans with desirable performance. However, in most cases queries are not executed sequentially. Therefore, these models should be further extended or modified to adapt to more general or complex queries and workloads. Also, to accurately quantify energy cost of an entire workload, energy models for concurrent queries should reflect the internal interactions among queries. Furthermore, more attention should be paid to investigate situations of which database servers run on virtual machines that have multiple tenants sending requests to a same server.

Note that most models in Section 4.3 assume a static system behavior and are offline approaches. Therefore, these models cannot adapt to complex online prediction, which also means accuracy cannot be guaranteed due to environment dynamics in term of workload fluctuations and changing DBMS configurations. We believe that a desirable energy model should be able to periodically update its parameters for consistent accuracy using real-time power measurements. The last but not the least, energy model that covers up various database transactions including update, insert, and delete needs to be paid more attention by future researchers.

4.4.3 Single Node vs. Cluster. Power and energy models of database operations at node level have achieved relatively rich results while lacking corresponding efforts at cluster level. And for large-scale database deployments [30] challenges not only exist in modeling but also in measuring their energy consumption. Moreover, cluster-level energy modeling is more urgent than that of node-level, since more servers means more energy cost.

Existing power models for standalone server mainly include two components (i.e., CPU and I/O) while ignoring memory, mainboard, network and other energy-consuming components. Only three works have explicitly taken memory power cost into consideration. Part of this phenomenon is because most of the energy modeling efforts were taken as an optional part of energy-saving strategies and these strategies only concentrated on hardware components with larger power cost like CPUs and disks. Additionally, leaving lower-power components as a constant can simplify the modeling process. However, these models might be only suitable for systems with specific configuration. In addition, models at node level may fail, if queries and transactions are executed in distributed environments with additional inter-node communication cost [122].

With the scaling out of clusters, memory has become a significant consumer of the overall system energy supply, due to its increased capacity and its relatively larger dynamic power range compared with traditional disks. Also, energy cost of network components has been an important part of the total cost of ICT [33, 110, 195].

Moreover, the increasing cluster size will simply introduce more network overheads [102]. Therefore, we suggest that more attention should be paid to build comprehensive energy models that give accurate predictions at cluster level. In addition, it is an also interesting direction to study how available single-node energy model can be extended to multiple-node environments.

Finally, previous research [182] investigated on single-node servers has revealed that the highest performing configuration of a DBMS is generally the one that saves the most energy. In other words, the opportunity of improving energy efficiency for single servers is quite small. This is due to the high static power cost of individual components in traditional database servers [8]. Fortunately, hardware vendors are aware of this issue and are paying more and more efforts to increase energy efficiency and energy proportionality of hardware and the entire server. Today's servers consume idle power as less as 20% of the peak power [94], while in 2010 [182] the static power cost was more than 50%. We believe that energy-saving opportunities for database systems not only exit in single nodes but also in database clusters as hardware evolves.

5 ENERGY MANAGEMENT TECHNIQUES

In this section, we first introduce two important metrics that are related to energy management including energy efficiency and energy proportionality, along with their motivations, goals, and impacts on energy-related research. Then we will describe and analyze the most relevant works focused on improving energy efficiency and/or energy proportionality for database systems, starting at a single node level and going through all the way to cluster level issues.

5.1 What are Energy Efficiency and Energy Proportionality?

It makes little sense when it comes to saving energy while ignoring requirements on performance. Energy efficiency is generally determined as using less energy to provide better or at least the same service output [21, 73, 182]. Usually, "service output" means the work done, and hence energy efficiency can be expressed as the rate of "useful workdone" per unit energy. Therefore, for a given amount of work, energy efficiency will be improved when the relative energy consumption is reduced. In addition, a more generic definition of energy efficiency can be denoted as the ratio of performance to power (i.e., performance-per-watt). Thus, energy efficiency will be improved when the relative improvement of performance ($\Delta performance/performance$) is greater than the relative increase of power ($\Delta power/power$) [182]. Generally, energy efficiency can be evaluated as following:

$$Energy\ Efficiency = \frac{Work\ done}{Energy} = \frac{Work\ done}{Power \times Time} = \frac{Performance}{Power} \tag{3}$$

Many energy-related works, spanning from chips to data center, have made their efforts or shared insights to improve energy efficiency. However, most of them have not defined it clearly to adapt to specific applications or given details for calculating it. To the best of our knowledge, the reason behind this phenomenon, as pointed out by Brown [21], is that it is difficult to measure service output since service output varies from application to application, especially for complex systems, for example a server or a data center. What is more, under different applications and scenarios, the notion of specific work done is different as well. For example, in OLTP (On-line Transaction Processing) systems, work done means transactions completed. Performance is usually represented as transactions per minute. Therefore, energy efficiency might be transactions/Joule. For OLAP (On-line Analytical Processing) systems, work done usually means queries completed. Performance for a given workload with fixed database size (e.g., 10GB) is typically defined as queries per hour. Thus, energy efficiency might be queries/Joule. Moreover, for a given query, performance is denoted as its response time, energy efficiency might be response time divided by the power for executing it. Producing metrics with detailed description to evaluate energy efficiency is not easy job and maybe time-consuming. However, both operators and researchers need clear definitions and

standards to identify and design promising energy-saving techniques, and to compare their works with others' for further improvements. Therefore, energy-related benchmarks are required for fair evaluation and comparison among different energy management techniques [149, 166].

The concept of energy proportionality is firstly described by Barroso and Hölzle [8], and has received extensive attention from researchers ever since its proposal [56, 79]. The motivation behind this notion is based on a six-month observation of thousands of servers in Google's datacentres. Barroso and Hölzle [8] found that the servers were hardly idle and the utilization level of servers was between 10%-50% in most of the time. Moreover, energy efficiency of a server when its utilization was between 20%-30%, at which point the server spent most of its time, dropped to less than 50% of the energy efficiency at its peak performance. This mismatch between servers' energy-efficient characteristics and the behavior of server-class workloads motivated Barroso and Hölzle [8] to promote hardware and system designers to develop energy-proportional machines in which energy consumption is proportional to the amount of their work done. In other words, power consumed by a server should be proportional to its delivered performance. To be more specific, a server would ideally consume no power when not used and the highest power only at peak performance.

When application performance is loosely defined as utilization, energy proportionality can be expressed as the application performance normalized to peak performance. Unfortunately, currently available hardware or systems are hardly energy-proportional. Designing systems that exhibit energy-proportional feature remains an open challenge. Note that to construct systems with energy efficiency and proportionality, hardware approaches are only a partial solution. Promising solutions lies in giving the software the power to intelligently control and to take full advantage of adjustable power features provided by hardware.

To conclude, energy efficiency and energy proportionality are two different aspects of energy management issues. Energy efficiency is typically driven by the need of using less energy to perform the same workload, aiming at maximizing the value of a given amount of energy. Energy proportionality is driven by an ideal vision to provide constant energy efficiency at all performance levels to further reduce energy cost for machines. Note that since for current hardware, power consumption is not in a linear relationship with performance, the value of energy efficiency will changes at different performance levels.

5.2 Energy-oriented Benchmarks

The motivation behind developing energy benchmarks is the ever-increasing energy consumption and the significantly improved energy-saving consciousness. IT researchers, computer manufacturers, as well as governmental agencies have been reporting analytic data or improvements on energy-related issues. However, due to the differences among workloads, applications, system configurations, and the lack of detailed description about them, their reported data cannot be directly compared between one and another. The development of energy benchmarks provides rigorous metric, audited and reliable measurement of performance and energy, which allows us to fairly evaluate and compare energy-efficient improvements among various energy management techniques. What is more, energy-oriented benchmarks and tools, as well as their continued refinements, will help to accelerate the development of energy-saving technologies by showing relationships among performance, energy, and design features, and to better facilitate both producers and customers to make energy-aware decisions [145, 166].

Current energy benchmarks can be classified into two major categories: 1) specialized benchmarks that designed particularly for energy evaluation, and 2) augmented benchmarks with added energy metrics to existing benchmarks [145]. In the last decade, the earliest as well as one of the most important efforts that tried to benchmark energy efficiency for database systems is contributed by a group of researchers from Stanford University and HP Labs. In SIGMOD 2007, an energy-oriented benchmark called JouleSort is presented by Rivoire et al. [149]. As an I/O-intensive benchmark, JouleSort looked into the difficulties of developing standards for

measuring systems' energy efficiency. It is also a complete energy benchmark focused on data management tasks, and consists of a workload, metrics, and guidelines. Additionally, JouleSort has become a part of a project (Sort Benchmarks) that was sponsored and administered by Jim Gray as in http://sortbenchmark.org/.

Well-known non-profit corporations and standardization organizations that used to develop performance benchmarks have actively participated in developing benchmarks for evaluating energy efficiency. The Transaction Processing Performance Council (TPC) has organized a work group to enhance energy metrics (i.e., TPC-Energy) to all of its released benchmarks (TPC Energy Specification). The Standard Performance Evaluation Corporation (SPEC) have also developed the SPECpower_ssj2008 (SPEC Benchmark Specification) to evaluate the relationship of power consumption characteristics and performance levels for server-class computing equipments. Apart from the above-mentioned benchmarks, in this survey we have identified that TPC-C (a typical benchmark for on-line transaction processing), TPC-H and TPC-DS (benchmarks for on-line analytical processing), and YCSB (a cloud serving benchmark for NoSQL) have also been frequently and widely used by the researchers to evaluate the efficiency of the proposed energy models and energy-efficient approaches. More details can be found in Table 3 of Section 4 and Table 5-9 of Section 5. Although various benchmarks have been developed to facilitate energy management, the large setup and measurement overhead of these benchmarks should be alleviated to help future researchers report more comparable results [146].

5.3 Single Node Level Energy Management

In this section, we survey the existing literature on energy management techniques for single node environments. According to their core ideas and basic approaches, we have classified the existing works into five main subcategories: A) energy-efficient query optimizer, B) DVFS-based energy-saving techniques, C) memory and substorage energy management, D) accelerators and tailor-made hardware, and E) augmented resource sharing and reusability. To overview the single-node techniques that fall into the above five subcategories, a horizontal comparison of them is firstly given in Table 4. Note that due to space limitation as well as for simplicity, we use A-E to represent sequentially the above-mentioned five subcategories.

As shown and compared in Table 4, for each subcategory, EC and P are the impact on energy consumption and performance, respectively. Specifically, H, M, and L represent a high, moderate, and low degree of influence according to the reported results, respectively. Moreover, the plus sign represents a positive influence while the minus sign denotes a negative effect. NS and NSF are the total number of selected studies and the number of selected studies published in the last five years, respectively. CHAR and LMT are the main characteristics and limitations of most studies that fall into the same class, respectively. Note that, for the third subcategories (i.e., C in Table 4) we further divided it into two parts, i.e., C1 for memory-based techniques and C2 for flash-based techniques according to their basic approaches. In the following subsections, we will discuss each of the node-level energy management approaches in detail.

5.3.1 Energy-efficient Query Optimizer. Query optimizer is a key component in database systems. It is responsible for a series of important jobs, such as generate candidate plans for a specific query, estimate the time cost of each plan, compare cost among plans, and finally select a plan with the least cost. Performance-driven query optimizer has been studied for several decades. And researchers have developed many techniques to fasten query optimization and processing. It is a natural intuition to propose a question that is the faster query plan always the more energy-efficient? The answer can be drawn from several studies as following.

Guo et al. [61] analyze energy efficiency of a number of query optimization rules that have been widely used by DBAs to rewrite queries to make them run faster. Experimental results indicate that performance-driven rules often result in increased power consumption. Höpfner and Bunse [80] and Bunse et al. [22] study energy efficiency of various algorithms in DBMSs. A same conclusion is drawn from their studies that the fastest algorithm is not the most energy-saving one, and algorithms with higher performance often require more energy than that

Table 4. An overview of the five subcategories at the single node level energy management

Class		EC	P	NS	NSF	CHAR	LMT
A		M+	L-	14	8	The energy saving result of each query is small but can be accumulated to a considerable amount in a daily business setting	Energy efficiency has not been addressed for distributed query processing
B		M+	L-	19	12	Taking advantage of the tradeoff between power and performance to save energy	Frequency combinations can result in a huge solution space and the selection strategy is not portable
C	C1	M+	L+	17	11	Scaling memory frequency and size as well as reducing I/O operations	A high risk of striking performance degeneration
	C2	M+	M+	13	2	Special characteristics of flashed-based devices to facilitate power-efficient operations	More complex storage hierarchy, algorithms, and architecturedesign for building hybrid databases
D		H+	H+	18	11	Customized architecture and system design for specific applications to achieve significant performance and energy saving improvements	Specialized designs need higher development cost, and are little reusable and difficult to integrated together
E		M+	L-	12	3	Pre-defined agreements to share and reuse various database resources	Distributed workloads and resources are not considered, and the sharing agreements are coarse-grained

of the slower ones. Additionally, Xu et al. [203] and Lang et al. [103] both provide motivating examples by showing the differences of time and power costs among alternative plans that generated for a specific query. Their experimental results show that energy efficiency is significantly improved when selecting plans with less power consumption and reasonable performance penalty. To conclude, performance improvement is not a decisive element for energy-saving opportunities in DBMSs. And in most cases, performance improvement is at the cost of aggravated power dissipation, and consequently will lead to more energy consumption. To the authors' knowledge, this is partly due to the poor energy proportionality of current hardware and systems.

A power-aware query optimizer is a key component of an energy-efficient database system. In such an optimizer both power and energy should be taken as important criteria for plan selection. Also, different from the traditional performance-driven optimizer, the goal of this new optimizer will be shifted to enable energy-efficient query processing. To realize such a green optimizer, firstly, the optimizer should be aware of energy consumption of all the plans that generated for a query. Secondly, the optimizer can select an optimal plan with the best energy efficiency for a query, while with as less as performance degeneration. Therefore, there are two major issues that need to be addressed for building such an optimizer: 1) How to make accurate energy estimation for different query plans; 2) How to balance performance against energy cost to achieve desirable trade-offs. For example, given a performance bound, how to minimize energy consumption? The first issue has been well studied in Section 4.3, while for the second issue, a number of studies have made their efforts as the following overview shows.

According to the definition of energy efficiency that it is computed by work done/energy, the plan with less energy consumption is also the one with more energy efficiency. Based on this assumption, an evaluation algorithm is proposed by Flach [49] to compare energy efficiency between two different plans of a query. Evaluation experiment is carried out in PostgreSQL. The availability of the proposed algorithm is verified by modifying the query optimizer to select plan that saves the most energy. Similarly, Liu et al. [116] present an algorithm to select the most power-efficient plan, which takes query plan sets generated from the optimizer as input and a most power-efficient plan as output.

By adding power and energy metric into query optimization, a metric model is designed to select energy-efficient plans [201, 203]. The availability of the model is evaluated by modifying the kernel of the database software. Integrated with a metric model and a power cost model, the modified query optimizer can evaluate

and compare the priority among alternative plans of a query, and finally select an optimal plan towards a user-preferred optimization goal (e.g., response time, power, and energy cost). Also, the metric model uses statistical data that extracted directly from the database, and can be used to explore alternative tradeoffs between power and performance. In a following work of Xu et al. [204], a tool named PET is developed to provide a user-friendly interface to interact with plan selection.

Plan generation is a CPU-intensive job in typical query optimizer, which means considerable energy will be consumed during query optimization. Bestgen et al. [16, 17] propose a strategy for plan generation and selection. Firstly, energy cost of generating and executing each new plan is estimated separately. And then potential energy saving is estimated by comparing energy requirement of a new plan with an existing old plan. Finally, a decision is made depending on whether potential energy saving exceeds the prospective energy requirement.

Lang et al. [103] propose an energy-efficient framework for query processing based on available slack time between the optimal performance and the performance specified by SLAs (Service Level Agreements). With a power model and an original time cost model, energy response time profiles (ERPs) for plans are generated. EPRs are structures that can be used to identify possible performance and power trade-offs for different system settings provided by hardware. In addition, with detailed ERPs, the modified optimizer can select the most energy-efficient plans, as well as meet the SLAs constraints.

Similarly, Xing [200] provides an analytical cost model to select energy-efficient plans with tolerable performance penalty. By introducing a factor that denotes performance degeneration into an evaluation model, the weight between performance and energy can be regulated for a possible maximum trade-off between the two criteria.

Roukh et al. [156, 157] introduce a methodology that aimed at saving energy during query optimization. To identify energy-saving opportunities, a tool called EnerQuery is designed to study trade-offs between energy and performance. The tool is built on top of a query optimizer in PostgreSQL DBMS. Also, to give DBAs the power to decide their desired trade-offs, the tool is augmented by a plan evaluation method that based on the weighted sum of cost functions.

Note that plan selection is driven by need not by performance or energy saving. Guo et al. [64] present a plan evaluation and selecting model after analyzing general query optimization mechanisms. By rating the importance of all the plans that created for a given query, the model is able to select plans towards a user specified optimization goal, for example energy efficiency. Additionally, the original query optimizer can be further enhanced by an accurate power consumption model with the plan evaluation model, which can take multiple cost metrics into consideration and select plans that realize the best trade-off between different metrics.

Instead of focusing on sequential query processing, Dembele et al. [37, 38] propose a green query optimizer for parallel query processing in multicore processor architecture on a single database system. To this end, a cost model is built by introducing an energetic factor denoting the degree of parallelism as well as expressing the difference between the sequential and the parallel modes in term of power cost. Based on available resource consumption information from database statistics, the cost model and its parameter are obtained using non-linear regression and neural network.

The basic idea shared by existing works that focused on building a green optimizer is to investigate and leverage possible compromise between energy cost and performance. Evaluation models and methods that have been proposed for selecting plans should be implemented in the original query optimizer to achieve real-world energy savings for query processing. Based on the above-mentioned works, we present a detailed schematic of such a green optimizer (as shown in Figure 3) by constructing an overall green framework for query optimization.

An important goal of a green optimizer is to maximize possible trade-offs between performance and energy under a given performance requirement or a green requirement. That is to select plans with acceptable performance and consuming energy as little as possible. Let $\{plan_1, plan_2, ..., plan_n\}$ represent all the plans that generated for

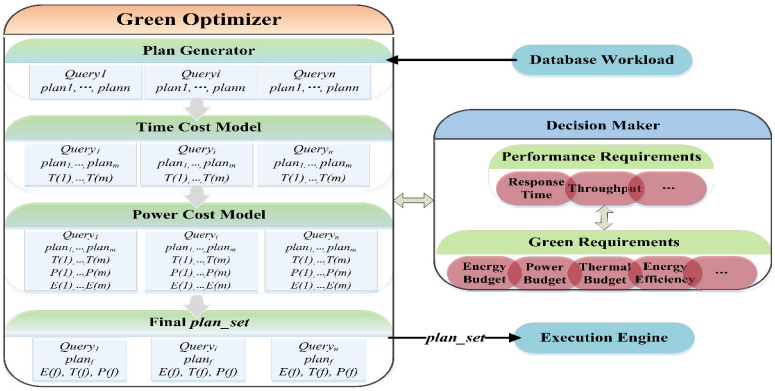

Fig. 3. Schematic of the green query optimizer.

a query, and a certain number of plans $\{plan_1, plan_2, ..., plan_m\}$ $(1 \leq m \leq n)$ can be firstly selected by setting a specific performance requirement in term of response time or throughput. Since, SLA (Service Level Agreement) has been typically considered as a standard metric to evaluate the acceptability of performance, SLA should be considered when filtering out plans that do not meet performance requirement. Then, energy cost of the initially selected plans of a query can be calculated by a power/energy model integrated within the optimizer. Finally, according to green requirements of users, such as power, energy, and thermal budget, a final plan that indicated as $plan_f$ $(1 \leq f \leq m)$ can be selected to execute the corresponding query. Following this strategy, assuming a query workload which is composed of n queries $\{Query_1, Query_2, ..., Query_n\}$, the desired way of executing these queries is to determine a $plan_f$ for each query in the workload. That is the final $plan\_set$ denoted as $\{plan_{1f}, plan_{2f}, ..., plan_{nf} | 1 \leq f \leq m\}$. Moreover, once an energy-efficient plan is generated and then selected for a specific query, the plan can be cached, since for future same or similar structured queries the plan can be reused to save energy and improve performance.

Each query statement that submitted to a database will go through 2 steps: 1) compiling the query statement and 2) then executing the query according to a selected plan. Since query optimization is a CPU-intensive job and is included in the compiling stage of query processing, time and energy cost of query compilation should in no case exceed that of the actual query execution. In addition, the evaluation model, along with the power/energy model, should meet the criteria described in Section 4.2. Note that the open-source nature of DBMSs like PostgreSQL and MySQL gives the community useful tools to experiment with. Researchers can directly modify kernels (i.e., query optimizer) of these DBMSs to testify feasibility and effectiveness of the proposed models.

5.3.2 DVFS-based Energy-efficient Techniques. Modern processors have been considered as the dominant power-consuming components in typical servers. Fortunately, their wide dynamic power range and power-efficient features like DVFS (Dynamic Voltage and Frequency Scaling) can provide opportunities for researchers to improve energy efficiency of database systems [8]. Numerous efforts and proposals have been conducted on reducing energy consumption of database servers based on DVFS. Generally, DVFS represents two power-saving techniques that can be used to save power cost of various systems by lowering supply voltage and/or clock frequency of a processor and its attached peripherals. DVFS has been recognized as one of the most important and effective techniques for constructing energy-efficient systems.

The motivation of DVFS is not only the benefit of reducing power but also the need for thermal issue. Since the Dennard scaling for semiconductor technologies stops, power density increases and the thermal issue gets worse [45]. Fortunately, modern CPUs is enabled to explore different power and performance trade-offs based on DVFS.

DVFS enables processors to control their performance levels and power costs by adjusting voltage in combination with clock frequencies at runtime. Modern operating systems also provide user-space power governors that help manage DVFS based on load perceived. However, energy consumption pattern of a database server is likely to change with different workloads [99]. Moreover, due to the sensitivity nature of database systems caused by workload fluctuations and environment dynamics, it is a challenge to determine to what extend the CPU's frequency should be scaled to minimize power consumption under a performance bound.

From the aspect of software, improvement of energy efficiency is generally depending on energy-efficient algorithms and the support from the OS itself to control power-saving features of processors. However, both CPU and the OS lack important information about specific database applications. As a consequence, leaving the responsibility of controlling processor power states to the OS or the CPU itself becomes the major problem of building green DBMSs. Comparing with coarse-grained control at the OS or processor level, database software is able to use information collected within itself to guide DVFS operations, since it maintains more detailed knowledge of itself and its workload. Some researchers have reported their efforts in manipulating power modes of processors from the data management software perspective as the following overview shows.

Lang and Patel [104] propose a technique called PVC (Processor Voltage/Frequency Control) with a focus on saving energy for query processing. Modern features of processors are utilized by PVC to execute queries at a low voltage and frequency. Instead of capping multiplier, PVS is realized by using underclocking to lower FSB speed, which enables a finer granularity of CPU frequency scaling for significant energy saving with reasonable performance degeneration.

Xu et al. [207] construct an online framework called PAT with a focus on power-aware throughput control. PAT is used to dynamically change CPU frequency to save energy, as well as maintain acceptable performance in term of throughput. By defining a linear relationship between throughput and CPU frequency, along with statistical data extracted from database, PAT can provide rigorous and formal feedback control of different CPU power modes. Also, it can be used to explore the relationships among system throughput, statistics of query processing, resource consumption patterns of workloads, CPU power modes, and power consumption.

Tu et al. [184] propose a DVFS controller that can be integrated into the DBMS software. By utilizing a feedback control loop, the controller can adaptively change CPU frequency according to collection and analysis of database dynamics. For example, when the workload is more I/O intensive, a low-power mode of CPU with a performance penalty is still able to meet the performance requirement of query processing.

Yang et al. [208] explore the potential of DVFS for mobile database applications on smart phones. To utilize explicit trade-off between performance and energy, a benchmarking system based on TPC-H is designed to measure performance and energy cost. Their experimental results suggest that traditional energy-saving techniques including DVFS is not always energy-efficient and suitable for mobile database applications, which suggests a dynamic relationship between performance and energy in mobile databases.

Due to the unpredictability of DBMS workloads, Zhu [219] designs a power regulation scheme based on a closed-loop feedback controller [50]. By binding active database threads to specific cores, the controller can regulate individual core frequency to change energy consumption of certain thread. And according to priority and degree of workloads' computation intensity, the controller can be used to save energy for the entire system.

Psaroudakis et al. [147] propose to use precise hardware models to facilitate DBMS to schedule query operators at runtime with a fine granularity. The hardware models are generated by building energy efficiency curves under different combinations of DVFS levels, parallelism levels, thread scheduling strategies, and memory access patterns. And for memory-intensive operators like scan and aggregation, the influence of DVFS on their performance and energy cost are also analyzed. In addition, the authors suggest that proposed models should be used to calibrate query operators in advance to reduce the cost of on-the-fly data mining.

Targeting at a query level, Götz et al. [55] present a simple and practical technique for developers to save energy for read-only (analytical) queries by using sweet spots, i.e., energy-efficient CPU frequencies. By changing

two key variables including the number of threads and CPU frequency, an optimal configuration for each TPC-H query can be finally found and therefore the existence of sweet spot frequency is confirmed. In addition, by describing and benchmarking the relationship between the two variables and the query's time and energy cost, the proposed technique can be further used to improve energy efficiency for database workloads consisting of application-specific queries.

Korkmaz et al. [100] argue to save energy for memory-intensive workloads by adjusting frequency and voltage of processors. The authors suggest that database system should utilize its knowledge of workload characteristics. Also, the management of DVFS within a database server should quickly reflect the natural short-term fluctuations of its workloads. Since energy savings obtained from DVFS is at the cost of performance to a certain extent, their experimental results imply that heavily memory-bound workloads will result in little performance degeneration when scaling down the processors.

Embedded databases are expected to support timely data services under various constraints, for example limited energy and data freshness of sensor data and transactions. Kang and Chung [90] explore energy-saving potential of DVFS on embedded databases. By combining the dynamic QoD adaption (quality of data) with DVFS technique, and taking them as two tuning knobs, a feedback controller is developed to manage energy consumption. Effectiveness of the proposed controller is evaluated by two pre-defined performance metrics: 1) tardiness, denoted as the ratio of actual response time to target response time), and 2) QoD, denoted as the ratio of the fresh data objects to the total accessed data objects.

Ungethüm et al. [186, 187] suggest that performance and energy efficiency should be both considered when selecting appropriate hardware settings for database workloads. The authors describe a concept of work-energy-profile (WEP). WEP can be generated by changing rich tuning knobs to test every possible configuration for a repeated workload. The knobs that are used for producing WEPs, such as DVFS, sleep states, core selection, and hyper threads, are provided by a heterogeneous multiprocessor and the operating system. With the help of detailed WEPs, an optimal hardware configuration can be finally selected to offer the best energy efficiency under a specific performance constraint.

By running online transaction processing workloads generated from a cloud OLTP benchmark, the inefficiency of existing Linux frequency governors is showed in [170]. Also, their experimental results show that active state (C0) is the most occupied power mode of processors and may result in aggressive power savings. By effectively quantifying utilization, a new reactive governor on top of the original Linux governor is developed to better adjust core frequency to improve power saving of transactions across all load levels.

A general conclusion can be drawn from aforementioned works that various system settings will result in different trade-offs between energy cost and performance. Kissinger et al. [94, 95] show experimentally that hardware itself is not able to effectively configure the provided rich tuning knobs due to its lack of awareness of application-specific knowledge. Energy profiles (EP) are generated to describe possible trade-offs between energy cost and performance by mapping a set of power-related control features to hardware configurations. And to guarantee a specific requirement on query latency, an energy control loop (ECL) is integrated into large scale-up NUMA system for in-memory database systems. The ECL based on continuously maintained EPs at runtime can be used to adaptively select a most energy-saving configuration for current workload.

The short-term fluctuation nature of transactional database workload provides researchers opportunities to save energy by adjusting system processing capacity to adapt to such dynamics. Korkmaz et al. [99] suggest that DVFS controlled by processors or operating systems cannot make quick and timely reactions to workload fluctuations. The authors develop a technique called POLARIS for transactional database systems to facilitate DBMSs to directly manage DVFS to minimize power consumption. Being aware of its working units (in term of transactions) and latency requirements, a DBMS integrated with POLARIS can make better decisions on 1) when and to what extend the DVFS will be scaled, and 2) how workload will be scheduled. In addition, energy

savings obtained from POLARIS are decided by two main factors: a) slack time of transactions' deadlines and b) the average load handled by system.

Guo et al. [65, 66] propose to save energy for cloud database systems by addressing the issue of resource provisioning mainly in term of the computing resource. By predicting required throughput for each node, the DVFS technique can be used to select appropriate frequencies for nodes in clusters. And with respect to overloaded nodes and workload balancing, the authors also design a migration process to decide how and where the workload should be migrated. In addition, to reduce the large solution space introduced by various frequency options and the number of nodes, frequency combinations are coded and the redundancies between combinations are also eliminated.

As an extension of Xu et al. [207], Xu et al. [202] propose to realize PAT as a container hierarchy called Crop based on Docker for real DBMS (PostgreSQL). To better understand how system performance changes with CPU frequency, a workload classifier based on fuzzy rules is built to provide accurate identification of resource consumption pattern for queries. Compared to running on virtual machines or physical servers, Crop can achieve more energy savings as well as maintain a desired throughput of the workload with acceptable performance degeneration.

To improve energy efficiency of modern edge devices in the IoT environments, Michalke et al. [127] propose EcoJoin with a focus on reducing energy consumption of join operations for streaming queries. By considering and exploiting characteristics of the given workload, EcoJoin can take full advantage of heterogeneous computing resources provided by SoC that integrated with GPUs. Specifically, a steady stream of tuples are batched to split the data processing into two different phases, allowing the processor to switch into low power states to save energy in the idle phase. In addition, the relationship among the batch size, power consumption, and latency is also investigated to achieve fine power-latency trade-offs.

We list down a summary of energy-efficient techniques based on DVFS in Table 5. The analysis table contains the researchers' names and publication year of reviewed studies, experimental results that these approaches obtained in term of performance and energy savings, main characteristics of these approaches, and the workload that the researchers used to verify their proposed energy-efficient techniques.

5.3.3 Memory and Substorage Energy Management. With the ever-increasing need of in-memory computing, the total share of energy cost of memory in database systems has become the second largest part [36] and sometimes can be comparable to that of processors [113]. DRAM energy consumption has attracted substantial attention in recent years but most works are focused on general computing systems. A survey conducted by Mittal [128] investigates a number of DRAM power-saving techniques for main memory systems. However, power management of memory has been less studied for current database applications.

Over the last several decades, Hard Disk Drives (HDDs) have been considered as the main type of media of secondary storage. A comprehensive survey conducted by Bostoen et al. [20] investigates the existing energy management techniques for disk-based storage systems in data center. However, with the quick development of a new storage media NAND flash usually in the form of SSDs (Solid State Drives), memory hierarchies of server systems are experiencing significant changes. What is more, a study by Janukowicz et al. [85] shows that the price of SSDs is dropping much faster than that of HDDs, and SSDs are expected to take the place of dominated HDDs in the near future. It seems that the prediction of "Tape is Dead, Disk is Tape, Flash is Disk." made by Jim Gray comes true [58, 59].

In this subsection, we first investigate energy-saving techniques that focused on memory and then move on to flash-based techniques for saving energy in the context of database systems.

Memory-based Techniques. Several energy-saving approaches associated with memory have been proposed for database applications. These approaches are all targeting at main-memory database applications, in which memory itself becomes a key component that need to be further explored to achieve better energy efficiency and

Table 5. Summary of DVFS-based energy management techniques

Authors	Results	Workloads	Characteristics
Lang and Patel [104]	20% energy saving, 6% performance degeneration	TPC-H	EDPs are produced to facilitate fine-grained DVFS by lowering FSB speed
Xu et al. [207]	51.3% energy saving with performance guarantee	TPC-C	Study the relationship between energy cost and throughput for queries by DVFS
Tu et al. [184]	51.3% more energy saving than OS-level mechanism	TPC-C, TPC-H	Scaling decisions are made based on collection and analysis of database states
Yang et al. [208]	N/A	TPC-H	Study the relationship between energy cost and performance for mobile database applications running on Android system
Zhu [219]	6.53% energy saving, 3.93% performance degeneration	N/A	Fine-grained DVFS at core-level based on closed-loop feedback control
Psaroudakis et al. [147]	Up to four times of energy efficiency improvement	N/A	Energy efficiency curves are generated under different DVFS levels, parallelism levels, thread scheduling strategies, and memory access patterns
Götz et al. [55]	Up to 50% energy saving	TPC-H	The number of threads and CPU frequency are used to identify the most energy-saving configuration for each query
Korkmaz et al. [100]	Higher energy saving than OS-managed DVFS	TPC-C	Database-managed DVFS at an individual core-level based on latency-aware P-state selection (LAPS) algorithm
Kang and Chung [90]	Lower energy consumption than baseline method	N/A	Explore energy-saving potential of embedded databases by combining dynamic QoD adaption with DVFS
Ungethüm et al. [186, 187]	N/A	N/A	Database-managed DVFS, and WEPs are constructed to find an optimal hardware configuration by changing various tuning knobs
Sen and Halverson [170]	N/A	N/A	A new reactive governor with better power efficiency that build upon the original Linux governor
Kissinger et al. [94, 95]	Energy saving ranging from 2% to 40%	N/A	Database-managed DVFS, and EPs are generated by mapping power-related control features to hardware configurations
Korkmaz et al. [99]	N/A	TPC-C, TPC-E	Database-managed DVFS based on awareness of its working units and latency requirements
Guo et al. [65, 66]	21.5% energy saving	YCSB	Appropriate frequencies are chosen for nodes according to predicted workload with computing resource provisioning
Michalke et al. [127]	Up to 81% energy saving against Handshake Join	N/A	Tuples are batched to split the processing into different phases, allowing the processor to switch into low power states to save energy
Xu et al. [202]	Up to 51.3% energy saving with 6% performance loss	TPC-C	PAT is realized as a container hierarchy called Crop based on Docker for real DBMS (PostgreSQL)

performance. The existing energy-saving techniques for in-memory databases can be mainly classified into two major categories: 1) memory capacity scaling and 2) power mode switching. Both of them are proposed to cope with the provisioning problem of memory capacity and bandwidth due to the increasing cores of processor and the growing need of in-memory computing [128]. Generally, memory size scaling is used when the provided memory capacity is larger or smaller than that of actually required. The basic motivation of size scaling is to avoid paying extra power cost for unneeded memory. Similarly, power mode switching is often used to allow unneeded memory to move to low-power modes, which can be realized by using memory DVFS technique [34] or throttling memory operations with memory controller [35].

Pisharath et al. [142, 143] present two techniques to reduce energy cost of query processing in memory-resident databases. One technique is to monitor and detect idleness of memory banks, and then switch among various power modes. Another technique is to maximize the possibility that data will be clustered by restructuring and regrouping queries according to their table access patterns. On one hand access clustering can help to increase inter-access idle time among memory banks, and on the other hand it will allow a more efficient and aggressive control of memory mode switching and their energy consumption behavior.

Bae and Jamel [6] describe a heuristic-based approach to dynamically change capacity of on-line memory (i.e., buffer pool size) to save energy for OLTP workloads. To efficiently and accurately identify the amount of data

required by workload, two metrics are defined and measured: Bhit (hit ratio of buffer block per request) and Butil (the amount of currently occupied buffer block). Based on the two metrics, enough data can be cached and the size of the buffer can be determined to whether expand or shrink with little performance penalty.

Similarly, Korkmaz et al. [100] propose to save energy for memory-resident database systems by dynamically adjusting memory size to reflect fluctuations of workload. A rank-aware allocation technique is proposed to arrange memory allocation to the underlying memory system in database. Also, power consumption of memory is related to the required capacity of memory by the database. Two challenges are discussed by the authors for future memory power management. First, algorithms used by the original memory controller need to be well documented to enable future improvements. Also, memory power estimation based on DRAM voltage sensor is not accurate. In addition, high sampling frequency and the overhead of transferring sampled data will add complexity to efficient memory power management.

Appuswamy et al. [4] explore potential of switching memory power modes to save energy for main-memory databases. Compared with traditional disk-based databases that CPUs are the dominating power consumer, their experimental results show that power consumption of DRAM will soon overshade that of CPUs in main memory databases. Trade-offs between performance and power are also investigated by using two mechanisms that are related with memory power management, including memory frequency scaling and power-down modes.

Generally, DRAM has two main technology limitations: 1) inefficient power management and 2) limited physical scalability. The near-zero static power (i.e., low leakage power) and high-density feature of NVM (Non-volatile Memory) are expected to provide opportunities for better energy efficiency and scalability. With the help of NVM, memory capacity can be expanded while extra static power of DRAM is saved. However, the latency and dynamic power of NVM is much larger than that of DRAM. To overcome such disadvantages, Hassan et al. [75] propose a hybrid memory architecture using both NVM and DRAM. Data placement strategies at the application level are also proposed to determine where to place the data in term of objects rather than pages. By identifying all data objects in an application, e.g., global variables, heap- and stack-allocated data, a profiler is generated to measure the memory access type and frequency of each object. Also, latency, static and dynamic energy of each object are estimated to provide decisive information for data placement and size scaling of DRAM and NVM. In a subsequent work of Hassan et al. [74], the authors further develop a strategy to find the optimal data placement in a hybrid memory for OLAP workloads. To this end, with the help of a tool [76], execution plans of queries and various statistics of their memory accesses are analyzed to decide where the data sets should be put. Since this strategy can produce high accuracy placement decisions, data migration due to wrong placement is avoided to improve performance and save energy.

A recent study by Korkmaz et al. [100] has illustrated that power cost of memory can be broken up into two parts: active power and background power. Active power is the power consumed for actual operations, and it is related to the frequency of operations performed on memory. Background power is consumed when memory is not used, and it will change among different memory power states. Also, each type of operation has a fixed unit of power cost. Karyakin and Salem [92] present an empirical study on how power cost of memory changes with database workload and database size in main-memory databases. The experimental results demonstrate that power cost of memory is depend on workload while is not in proportion to the amount of workload. Additionally, compared with active power, background power is the major consumer of the overall memory power cost, which indicates more energy-saving opportunities of memory can be provided by reducing background power.

In a following work, Karyakin and Salem [93] propose DimmStore that can be served as a testbed to identify energy-saving opportunities for transactional workloads in in-memory database systems. The motivation of developing DimmStore is that significant energy is expected to save by taking full advantages of the power-saving features provided by DIMMs. Specifically, by dividing virtual address space into two regions with individual memory allocators, memory load can be shifted to system region and hence access rates of data region can be reduced as much as possible to save the background power.

Dreseler et al. [44] propose to accelerate memory operations for in-memory databases on large-scale NUMA (Non-uniform Memory Access) systems. An in-depth and detailed overview of SGI UV architecture and functionality provided by GRU (Global Reference Unit) API is presented to facilitate understanding and future result analysis. API is provided by SGI to interact with and explicitly instruct GRU. Also, GRU provides API to accelerate offloading of memory operations, as well as provides instructions to execute atomic memory operations. By offloading distant memory access and using explicit access instructions, throughput is improved. Their experimental results reveal that performance of full table scans can be improved by 30% using the gru_bcopy operation. Transaction sequencing and latches can be improved by 10x and 8x respectively using GRU's atomic memory operations.

Memory swapping is an expensive process that requires moving data to/from disk [57]. Therefore, it not only has a negative impact on performance but also has been considered as a major consumer to energy cost of multicore-based database systems [48]. Zhou et al. [217] provide an approach to enhance energy efficiency for database operations by alleviating the memory swapping issue. To estimate energy efficiency of database operations, EDOM, a multicore manager with a benchmark tool is developed. And the most important component of EDOM is a memory cost model which is able to predict memory utilization based on four factors including the number of tables, the types of queries, the number of records, and the size of the record. With the help of the cost model, a proper number of cores can be decided by the multicore manager to avoid uncalled-for memory swapping.

Dominico et al. [43] design a core allocation mechanism to improve performance and save energy for NUMA systems by alleviating data movement. Traditional thread mapping for modern database system among NUMA nodes may result in inefficient memory activity (i.e., remote memory access). A priority queue is used to keep the history of threads' PID and their resource usage information such as accessed memory address space and execution core. Based on the priority queue their proposed mechanism can decide an optimal allocation of cores and distribution among NUMA nodes to reduce remote memory accesses.

Similarly, to alleviate the bottleneck of moving data between memory and cores, Imani et al. [83] propose a configurable architecture NVQuery based on content addressable memory (CAM) to accelerate basic query operations. By utilizing the non-volatile characteristic of CAM, data can be processed locally in memory, thus reducing the cost of data movement as well as speeding up query processing. In addition, configurable approximation in NVQuery is realized by 1) adaptively using voltage scaling for selective blocks in memory and 2) mapping operations to needed voltage levels according to their nature.

Chandrasekharan and Gniady [28] introduce a technique called QAMEM to save energy for memory by utilizing application-level information provided by DBMS software. Based on extracted information from executed queries and the system, memory requirements for running expected queries are estimated. And then, memory that are not frequently accessed can be switched into lower-power states for energy conservation. However, inappropriate power states for memory may result in longer response time of memory-intensive queries, and finally total energy cost is increased. By monitoring memory accesses and CPU time cost, the correlation between memory behavior and specific query is identified. Based on obtained correlations, their experimental results illustrate that appropriate selection of memory power states can save significant energy with small performance degeneration.

Other than memory capacity scaling and power mode switching, several efforts are dedicated to develop energy-efficient buffer cache algorithms. Being a bridge between I/O and CPU, the influence of memory on performance and energy cost of I/O operations cannot be ignored. What is more, whether to improve performance or save energy, controlling buffer pool has always been the primary and obvious target since it has the largest occupation of the whole memory capacity [100].

Most of the existing buffer algorithms, for example the well-known LRU and CLOCK, are traditional disk-oriented algorithms designed for improving performance. With the emergence of low-power flash-based SSDs, several buffer algorithms including CCF-LRU [112], CFDC [138], and AD-LRU [86] have been proposed to

improve performance for SSDs-based databases. These algorithms have better efficiency and are all aimed at the asymmetric read/write characteristics of flash memory. However, although energy savings may be a side benefit of significant performance improvement, these algorithms are not designed for energy management, thus cannot be directly used to improve energy efficiency.

Different from general-purpose databases, a real-time database in embedded systems has two major constraints including power cost and deadline miss ratio. Traditional buffer management is designed for high hit ratio to reduce I/O operations and improve performance, which is not suitable for real-time databases in power-constraint embedded systems. Though, power savings can be achieved by reducing I/O operations, the asymmetric read/write characteristics of flash memory may balance out the profits gained from high cache hit ratio. A power-efficient buffer pool management strategy is proposed by Kang et al. [91] to save energy for flash-based real-time databases. To adapt to different performance and power features of read/write operations of flash-based storage, the original buffer pool is logically divided into updated and non-updated buffers. Subsequently, data are also grouped into two categories: temporal and non-temporal data. Update transactions are used to update temporal data periodically while non-temporal data are modified by user transactions. Finally, a feedback control loop is developed to schedule read/write workloads to meet the requirements of miss ratio and I/O power conservation.

Existing page replacement policies in flash-based databases are facing the challenge of reducing unnecessary energy consumption caused by evicting dirty pages, because, compared with read operations, write operations cost more energy. And each dirty page eviction will lead to a write operation on the flash memory. In addition, loading an evicted page with more log pages is more expensive than that of an evicted page with fewer log pages. To solve this problem, Cesana and He [27] present a multi-buffer manager with a buffer replacement policy based on the fact that energy cost of evicting a data page is determined by the number of its related log pages. Firstly, the global buffer pool is divided into a number of local buffer pools with various sizes. This allows dirty pages with different log pages to be assigned and put in separate buffer pools. Also, these pools store data at a granularity of page. Being aware of loading cost of different pages, the decision of which page to evict can be made by the proposed policy. And then the manager can calculate an optimal buffer size to adapt to workload dynamics and satisfy energy conservation.

Ou et al. [139] explore performance and energy efficiency of several existing buffer algorithms in database systems using both SSDs and HDDs. Several important principles of using flash memory to achieved better energy efficiency for hybrid databases are discussed. In contrast to the notion that performance and energy are two independent optimization goals, their experimental results indicate a strong relationship between performance and energy-saving opportunities in flash-augmented databases.

Opportunities of saving energy based on designing energy-efficient buffer algorithms are provided by low-power SSDs. However, above-mentioned algorithms are proposed with an assumption that the secondary storage is only composed of flash memory, and therefore cannot adapt to changes in storage system such as economical hybrid databases. We suggest that future explorations on energy-efficient algorithms are required for hybrid-storage databases.

Now that we have reviewed a number of memory-related energy-saving techniques, we present a summary of them in Table 6. The analysis table includes the article year, authors, benchmark used, main experimental results and characteristics of their proposed methods.

Flash-based Techniques. Solid-state drives (i.e., SSDs) are a type of storage device that consists of NAND-based flash memory [20]. Due to its special characteristics, such as low power consumption, high I/O speed, non-volatile, and fast random reads, SSD has become a strong candidate for traditional disk [59, 107, 183]. Also, the availability of SSD increases the probability of building energy-proportional storage [60]. However, with the development of SSD, new challenges regarding architecture and algorithms design have been posed to traditional disk-oriented database management systems [40]. Moreover, it can be challenging to optimize performance of database systems based on SSDs because of their special characteristics, for example the asymmetry among

Table 6. Summary of techniques for memory energy management in database systems

Authors	Results	Workloads	Characteristics
Pisharath et al. [142, 143]	Up to 90% energy saving and 45% performance improvement	TPC-H	Restructure and regroup queries to increase the idle times of memory based on their table access patterns
Kang et al. [91]	N/A	N/A	Power-aware buffer pool management based on divided buffer pool and classified data
Cesana and He [27]	Up to 40% energy saving compared with the CFLRU policy	TPC-C	A set of buffer pools with various sizes, and data stored at a granularity of page
Ou et al. [139]	N/A	TPC-C	Explore performance and energy consumption of several existing buffer algorithms for flash-based DBMS
Bae and Jamel [6]	4%~8% energy saving without performance penalty	TPCC-UVa	Memory size scaling based on two metrics (Bhit and Butil) that are defined to quantify the amount of required data
Korkmaz et al. [100]	Up to 40% power saving with 7% performance degradation	TPC-C	Dynamically arrange memory allocation to reflect workload fluctuations
Appuswamy et al. [4]	N/A	TPC-C, TPC-H	Explore memory frequency scaling and power-down modes to save energy for main-memory databases
Hassan et al. [75]	Energy saving with less than 3.81% performance degradation	TPC-H, Twitter, YCSB	Hybrid memory architecture using both DRAM and NVM, and data placement are decided by identifying data objects
Karyakin and Salem [92]	N/A	TPC-C, TPC-H	Study how memory power consumption changes with different database workloads and database sizes
Dreseler et al. [44]	30%, 10x, 8x performance improvements on table scan, transaction sequencing, and latch	TPC-H	Accelerate memory operations on NUMA system by offloading distant memory access and utilizing explicit access instructions
Dominico et al. [43]	Up to 1.53x speedup and 26.5% energy saving	TPC-H	Decide an optimal allocation of cores and its distribution on NUMA nodes to reduce remote memory access
Chandrasekharan and Gniady [28]	25% energy saving	TPC-H	Switch among memory power modes using application-level information provided by the DBMS software
Karyakin and Salem [93]	Up to 50% power saving with minor performance degradation	YCSB, TPC-C	Shift as much as memory load to system region to reduce background power of DIMMs of data region
Imani et al. [83]	49.3x speedup and 32.9x energy saving	N/A	Data are processed locally in memory and voltage overscaling are used to realize approximation with acceptable error rates
Hassan et al. [74]	Up to 81.5% energy saving and 3.3% performance degradation	TPC-H, TPC-DS	Query plans and various statistics of their memory accesses are analyzed to decide the best data placement strategy
Zhou et al. [217]	N/A	TPC-W	Decide an optimal number of cores to alleviate memory swapping issue based on a memory cost model

read, write, and erase operations. Note that although the price per gigabyte of SSD has been falling, it is still comparable higher than that of HDD. Therefore, one of the major challenges is to decide appropriate capacity trade-off between HDD and SSD to obtain desirable performance and price cost for individual applications.

Meng et al. [125] present a detailed report on building flash-based database systems. However, this report is focused on techniques that tried to make the most of SSDs to improve performance. With the increasing awareness of energy efficiency, a number of afterward works have shared their insights on improving energy efficiency for hybrid or flash-only database systems.

Härder et al. [72] present a systematic comparation of SSDs and HDDs on their I/O performance (i.e., sequential/random read and write operations), power cost per I/O operation, and throughput. The experimental results suggest that "flash beats disk" in read-only scenarios. However, due to the relatively slow random write operation and its considerable power cost, traditional disk-orientated DBMSs are not able to take full advantage of flash characteristics. Therefore, specific designs are required for adaptive cache management and logging [40].

A following work of Härder et al. [71] addresses several important issues verifies that whether important features of SSDs are as expected, for example, the read and write asymmetry, the performance of overwriting blocks on full disks compared with that of writing on empty disks, and the influence of queue depth on performance.

Their experimental results demonstrate that power profiles of various SSDs are different. Also, power cost of SSDs no matter in idle mode or under peak load, is much smaller than that of disks.

Similarly, Schall et al. [168] provide a more detailed cross-comparation of SSDs and disks (including three types of SSD and HDD) to validate performance features claimed by hardware manufactures. Their findings suggest that SSDs show better energy efficiency than disks, and the newer SSD the more efficient. Additionally, the standby mode of SSDs almost consumes zero energy, and SSDs are able to switch quickly among different power modes. Therefore, the authors believe that SSDs have become an ideal candidate for building energy-proportional storage systems [162].

Cost model has been playing a vital role in query optimization and processing. Given the different performance features of SSDs and HDDs, Bausch et al. [10] and Park [140] both argue to modify cost model of traditional disk-oriented databases to adapt to flash-based systems. Bausch et al. [10] suggest that it is necessary to take hardware-specific features into account when it comes to cost estimation and plan selection in query optimizer. An asymmetry-aware cost model is proposed by Bausch et al. [10] to speed up query processing in flash-based databases. The proposed model is based on four parameters that are related to I/O operations, which allows query optimizer to distinguish sequential and random operations as well as read and write operations. Similarly, Park [140] builds a flash-aware cost model for join queries including join algorithms and scan methods. The model is built on top of the original disk-oriented cost model by distinguishing read, write, and erase operations. Since flash translation layer (FTL) will introduce additional overhead of read and write operations on SSDs, two parameters are also defined to calculate addition cost of each operation separately.

Pelley et al. [141] suggest that it is not necessary to modify the original cost model, since SSD-oblivious optimizer can make effective choices for plan selection in most cases. However, only a limited number of query operators are considered and analyzed by the authors. In addition, existing works are all concentrated on the possible performance improvement while ignoring the potential energy savings by building asymmetry-aware cost model. We suggest that future investigations are required to make a more comprehensive conclusion.

Do et al. [39] implement a prototype of relational DBMS that enhanced with a Samsung Smart SSD. Different from common SSDs, Smart SSDs have memory and simple processor inside the device. By offloading database operations, including simple selection and aggregation operators, to a query processing framework that augmented by Smart SSDS, significant performance improvement and energy reduction is observed compared with regular SSDs.

Schall and Härder [162] explore the possibility of building an energy-proportional distributed database cluster using SSD-based storage system. Based on benchmark tests, detailed comparisons of performance, power consumption, energy efficiency, and energy delay product (EDP) between SSD-based cluster and HDD-based cluster are provided. Each comparison is made under three different configurations including a small cluster and a big cluster with fixed sizes, and a dynamic cluster with flexible sizes. The experimental results show that energy proportionality can be approximated by dynamically reconfiguring the storage to adapt to current workload.

Based on approximate hardware designs, He [77] constructs a hybrid-storage database system named ApproxiDB. The basic idea of approximate hardware is to use the trade-off between declined result accuracy and increased performance and/or reduced energy cost. Also, solid-state memories are used to develop such approximate database system, in which approximate query processing (AQP) is more tolerant and users do not need accurate answers. Additionally, several challenges for building ApproxiDB are identified and discussed, ranging from physical design to multi-criteria optimization, which suggests ApproxiDB is a promising direction for faster and greener databases.

Zhang et al. [212] propose a hybrid-DB using both HDDs and SSDs. Instead of replacing all the HDDs with SSDs, they suggest that SSDs should be used to enhance performance of the original storage engine. Therefore, the placement of SSDs is required to provide a reasonable balance between performance and price cost. The original storage system is redesigned by constructing a two-level caching hierarchy. Level one is severed as a

Table 7. Summary of flash-based energy-efficient techniques

Authors	Results	Workloads	Characteristics
Härder et al. [71, 72]	N/A	N/A	Systematic comparation on I/O performance, power cost per I/O operation, and throughput test of SSDs and disks
Guerra et al. [60]	40%∼75% energy saving	N/A	Explore the potential and challenge of building energy-propositional storage system
Schall et al. [168]	N/A	N/A	Detailed cross-comparation of SSDs and disks to validate the features claimed by hardware manufactures
Pelley et al. [141]	N/A	Wisconsin, TPC-H	Explore the effectiveness of plan selection in SSD-oblivious optimizer for flash-based databases
Do et al. [40]	Up to 9.4X speedup	TPC-C, -E, and -H	Improve performance of buffer manager by dealing with evicted dirty pages
Bausch et al. [10]	48% performance improvement	TPC-H	Build an asymmetry-aware cost model that allows the optimizer to distinguish different read/write operations
Park [140]	N/A	SysBench, TPC-C and TPC-H	Build a flash-aware cost model for join queries by quantifying additional overheads of read and write operations on SSDs
Do et al. [39]	2.7x speedup, 3.0x energy saving	TPC-H	Offload database operations to an extended query processing framework based on Smart SSDs
Schall and Härder [162]	N/A	N/A	Detailed comparisons of performance, power, energy efficiency, and EDP between SSD- and HDD-based clusters
He [77]	N/A	N/A	A hybrid-storage database based on AQP that trades off results accuracy for increased performance and reduced energy cost
Zhang et al. [212]	N/A	TPC-H	Semantic information of I/O operations are identified and passed to storage manager through a direct communication channel
Yun et al. [210]	Up to 58% performance improvement and 51% energy saving	YCSB	Similar and nearby data are clustered to find the most likely referenced data for future reuse, and hence mitigate access latency of the flash

caching device and is composed of SSDs, while level two is composed of HDDs. To further utilize SSDs, critical semantic information of I/O operations are distinguished and then transmitted to the underlying storage manager through a direct communication channel. In addition, a number of effective rules are designed to facilitate the coordination between SSDs and HDDs since they have fundamentally different working mechanisms.

To lower the price cost and improve energy efficiency for in-memory database systems, Yun et al. [210] present a hybrid memory architecture integrating both DRAM and NAND flash. To alleviate the performance degeneration introduced by using the flash, a migration engine based on spatial locality is established to cluster similar and nearby data since these data are more likely to be reusable. And to manage the irregular access patterns of memory, based on linear regression analysis, a prefetching mechanism with a buffer is also proposed to store and load data by using historically requested pages to predict future requested ones.

We investigate and analyze existing works that based on flash memory to save energy for database systems up to this point. We present a summary of them as shown in Table 7. The analysis table contains the authors, publication year, experimental results, benchmark utilized, and the main characteristics of the proposed approaches.

5.3.4 Accelerators and Tailor-made Hardware Designs. Over recent years, the evolving application requirements have strengthened the widespread believe that "one size does not fit all", and accelerated the trend of designing specialized hardware for specific database applications [2]. Also, to tackle the new challenges introduced by power and thermal constraints, as well as the increasingly complexity of database applications, a number of researchers have proposed to use hardware accelerators, such as FPGAs, GPUs, and ASICs, to accelerate database operations as well as improve energy efficiency [129].

Mueller and Teubner [132] describe an intuition that the data-intensive nature of database systems will make them a good fit for FPGA-based query processing. A FPGA is a semi-custom integrated circuit that allows itself to be programmed and configured with flexibility. The authors discuss potential problems and challenges that

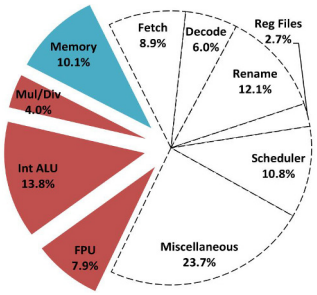

Fig. 4. Energy breakdown of the original pipeline [31].

in front of FPGA-accelerated databases to be widely used. Major challenges include design techniques that can be used to develop programmed FPGA chips for database workload, and how FPGA chips and general-purpose processors could be integrated together to speed up database operations.

Casper and Olukotun [25] argue to use dedicated hardware to accelerate in-memory database operations, especially join operations for online analytic processing and data mining workloads. Generally, workloads that analyze massive datasets are irregular and hardly indexed, which makes traditional joins more expensive. A prototype system based on a FPGA platform is implemented to execute database operations including selection, merge join, and sorting. Also, an equi-join of two tables is performed to show the utilization of memory bandwidth can be significantly improved for perspective energy savings.

Similarly, Kocberber et al. [97] propose to investigate analytic workloads on current in-memory databases. Their experimental results highlight that hash index lookups are the largest time consumer of the overall execution, since they require ALU-intensive key hashing together with memory-intensive node list traversals. The authors also present an on-chip accelerator called Widx for speeding up hash table lookups that 1) walk multiple hash buckets parallelly and 2) decouple key hashing from list traversals. By using a custom RISC core, flexibility is ensured to various schemes and data types. And by tightly integrating with a conventional core, the complexity and overhead of designing such specialized TLB and cache is avoided.

Becher et al. [11] present a query-specific FPGA-based hardware accelerator to speed up query processing as well as reduce energy consumption. Dynamic partial reconfiguration is used by the proposed query accelerator to generate query plans, which also significantly extend the scope of database operators. Compared with benchmarks that run on a x86-based server, their approach can save energy up to 5% at an equal throughput level.

To future illustrate energy-saving opportunities provided by customized hardware, and better understand energy inefficiency of general-purpose processors, one detailed example is presented as shown in Figure 4 [31]. Figure 4 shows a detailed energy breakdown of a general-purpose processor. The energy consumption is drawn from the SPEC benchmarks based on McPAT framework. As well known each step of the computation task will execute an instruction. From the picture we can see that, fetching an instruction will cost 9% energy, decoding an instruction will cost 6% energy. Besides, there are other energy-consuming stages before the actual computing units (i.e., Int ALU, FPU, and Mul/Div) and memory. Note that memory and computing units only account for 26% and 10% of the total energy cost, respectively. To be more specific, the majority of the total energy is consumed to facilitate the instruction-oriented model of general-purpose cores, while not for performing the actual computation task. Therefore, if all the assisted components that do not need actual computation are reduced, then chip area can be significantly extended to highly speed up computing and improve energy efficiency. What is more, with the help of customized hardware, bit wide can be reduced. For instance, different from the typical 32-bit or 64-bit processor, for graph analyzing and computing the bit wide can be reduced to maybe 8.

Note that the majority of the total power cost consumed by processors is not for actual computation. This observation motivated Cong et al. [31] to design domain-specific customization architectures to improve performance per watt for specific application workloads. By paying more energy to actual computation, the improved ASIC-based accelerators can be used to achieve better performance and energy efficiency. Several important issues that are related to accelerator-centric design are also discussed: design space exploration, promising ways to improve energy efficiency, and performance limitation introduced by the interface between ABB island (accelerator building blocks) and NoC (network-on-chip).

Arnold et al. [5] propose to build application-specific processors to speed up and save energy for data-intensive tasks. The authors discuss the feasibility of building such a DB-processor by showing how to create an extended instruction set that integrated within a low-power processor. The potential and limitations of two techniques, including element-wise instructions and intra-element wise instructions, are also discussed. And combining the two techniques, a new instruction set is generated by merging existing instructions and building their SIDM versions. However, the proposed instruction set is only used to speed up queries that limited to sorting and operations performed on sorted sets. Therefore, more instructions that cover various basic functions of query processing need to be further explored and developed.

Wu et al. [197, 198] present a database processing unit (DPU) called Q100 that can support basic operators of query processing, such as select, aggregate, boolgen, colfilter, partition, join, and sort. Q100 is an energy-efficient data analysis accelerator that operates on relational tables and columns. Pipeline parallelism and data streaming are also used to speed up query processing, improve throughput, decide relationships between operators, and identify data dependency between instructions. Based on TPC-H queries, the authors also manually perform a detailed design space of Q100 with 150 different configurations using 11 types of hardware tiles. Finally, three representatives of them are selected for future exploration: a low-power configuration, a high-performance configuration, and a configuration with maximum performance per watt (i.e., energy efficiency). Particularly, queries are represented as graphs with nodes denoting operators and edges denoting data dependency between instructions.

In a following work of Ungethüm et al. [185], a low-energy processor that contains a large number of heterogeneous cores (i.e., a heterogeneous MPSoC) is constructed for database applications. Each core is customized with an extension of domain-specific database instruction set. To build such a complex architecture, generally, two important challenges need to be tackled: 1) efficient data transfer among different cores and 2) a fine-grained data partitioning for query processing. The first can be solved by a core manager that provides data locality information and delegates data transfer. And the second can be solved by using intra-operator parallelism mode of operators at a task-level parallelism.

Compared with CPUs, parallelization of string-matching algorithms for query processing on GPUs is difficult. Sitaridi and Ross [176] propose to implement string matching operators for queries on two different GPUs. Memory and thread divergence on GPUs for both single and multiple pattern string matching are investigated. And by determining the appropriate parallelism granularity and string layout, string searching of different algorithms can be optimized. Their experimental results imply that the KMP algorithm would be preferable for GPUs since its regular memory access pattern.

Based on MPSoC, Haas et al. [67] develop a hardware accelerator (Tomahawk3) to speedup selected operators and queries in databases. Performance is improved by using RISC processors that extended with a specific instruction set and a task scheduling unit. Also, to minimize power consumption, ultra-fast per-core DVFS and AVS (adaptive voltage scaling) are used to enable hierarchical power management. A following work of Haas et al. [69] presents two application-specific architectures to accelerate and improve energy efficiency of basic database operations. One is an extended instruction set developed on Cadence Tensilica processor (ASIP) to measure performance and power consumption. Another is an application-specific integrated circuit (ASIC) to enhance the ASIP approach, and to further quantify the results. Additionally, Haas and Fettweis [68] propose to implement

hash-join algorithms on Tomahawk4 MPSoCs based on the ASIP approach. Four database-specific cores are included in the chip. And each of the accelerator cores has its own local SRAM and an extended instruction set that tailored for hash algorithms. Also, an off-chip DRAM is shared by these cores as the main memory. Their experimental results suggest that current hash-join algorithms can be well fitted in MPSoC architectures. Note that data partition will be needed to better exploit system parallelism, and main memory access is also required to adapt to the size of the input relation and hash table.

Salami et al. [158] develop an AxleDB on a FPGA-based platform to enable fast and energy-efficient query processing. By using modern interfaces, the database-specific accelerators are integrated with flash-based storage. AxleDB is served as a coordinated infrastructure between the host and data storage, and it directly supports blocks of data columns. To manage data movement between accelerators and storage units, as well as minimize overhead of SSD I/O operations, it supports FPGA-based database indexing operations to provide quick scans of large tables.

Balkesen et al. [7] propose a relational, columnar, in-memory query processing engine named RAPID. RAPID is based on hardware and software codesign to provide architecture-conscious performance and better performance per watt. RAPID is designed and implemented with a simple and low-power processor specialized for SQL. And the data processing unit (DPU) of RAPID is composed of 32 low power cores. Also, the new DPU abandoned several complex components such as large caches, cache coherence, advanced instructions, and sophisticated branch predictors. Since it is not cache-coherent, the data movement between DRAM and cores is managed by an on-chip programmable data movement engine.

Bitmap index can effectively support parallel processing and multi-dimensional query while its creation is a time- and energy-consuming job. Nguyen et al. [136] provide two hardware accelerators named BIC64K8 and BIC32K16 to speed up bitmap index creation and maximize indexing throughput. BIC64K8 and BIC32K16 can parallelly index as many as 65 536 8-bit and 32 768 16-bit words for each clock cycle. Both accelerators consist of 1) a content-addressable memory that uses the bit-sliced technique, 2) a query logic array module that contains a set of logic gates and multiplexer.

To accelerate data-intensive applications for database systems in the cloud, Sun et al. [179] propose a storage engine embedded with a novel data filter based on FPGA. Mainly benefiting from the inherent parallelism provided by FPGA, the data filter can be used to parse data blocks into rows, check qualified row units, and finally alleviate the bottleneck of filtering data during query processing. Moreover, in order to avoid the performance degeneration caused by high selectivity, an adaptable software switch is employed to decide whether to turn the filter on or off.

Due to the limited amount of reconfigurable resources, FPGA cannot efficiently and simultaneously support complex and resource-hungry database operators including sorting, aggregation, and join. Targeting at this issue, Moghaddamfar et al. [130] propose morphing sort-merge to support as well as accelerate these pipeline-breaking operators by utilizing run-time configurability of FPGA modules to reuse their dedicated resources at fine granularity. The key feature of morphing sort-merge is that a typical sort phase can be transformed into a partial aggregation with higher data reduction. Leaving the subsequent merge phase less resource-demanding, morphing sort-merge improves resource utilization as well as system performance.

We describe existing works that focused on building energy-efficient DBMSs by accelerators and tailor-made hardware up to this point. All the proposals discussed above are summarized in Table 8. The analysis table contains the author names, publication year, major experimental results, benchmarks, and the characteristics of above proposed approaches.

5.3.5 Enhanced Resource Sharing and Reusability. Different from commonly used approaches that based on trade-off between performance and energy, a number of works are concentrated on how to share and reuse various database resources for queries to optimize both performance and energy consumption. Generally, resource

Table 8. Summary of energy-efficient techniques that based on accelerators and tailor-made hardware

Authors	Results	Workloads	Characteristics
Mueller and Teubner [132]	N/A	N/A	Explore the potential and challenges of building DBMS based on FPGAs
Kocberber et al. [97]	3.1x performance improvement and 83% energy saving	TPC-H, TPC-DS	An on-chip accelerator that tightly integrated with a conventional core for hash index lookups
Cong et al. [31]	97% energy saving of the compute units	N/A	Explore domain-specific customization architecture by paying more energy to actual computation in ASIC-based accelerators
Arnold et al. [5]	A 960x better energy efficiency than a high-end x86	N/A	Build application-specific processors by creating an extended instruction set that integrated with a low-power processor
Becher et al. [11]	5% energy saving with same throughput	TPC-DS	A FPGA-based query accelerator that extends the scope of supported operators, and use dynamic partial reconfiguration to generate execution plans
Casper and Olukotun [25]	More than 1.4x utilization improvement	N/A	A prototype based on four FPGAs to accelerate database operations including selection, sorting, and joining
Wu et al. [197, 198]	3x energy saving and 70x performance improvement	TPC-H	Explore the design space of database processing units that operate on relational tables and columns as well as support various operators
Ungethüm et al. [185]	N/A	N/A	A low-energy processor based on a heterogeneous MPSoC with individual customized cores
Sitaridi and Ross [176]	N/A	TPC-H	Investigate memory and thread divergence on two different GPUs for single- and multi-pattern string matching algorithms
Haas et al. [67]	96x improvement of energy efficiency	N/A	A heterogeneous hardware accelerator that uses ultra-fast per-core DVFS and AVS to enable hierarchical power management
Haas et al. [69]; Haas and Fettweis [68]	Up to 5x speedup and 200mW less power consumption	N/A	Explore application-specific architectures to accelerate database operations based on ASIP and ASIC
Salami et al. [158]	1.8x∼34.2x speedup, 2.8x∼62.1x energy efficiency	TPC-H	Database-specific accelerators that integrated with flash-based storage on FPGA-based platform
Nguyan et al. [136]	1.43GB/s∼1.46GB/s throughput, 6.76%∼3.28% energy saving	TPC-H	Two accelerators to speed up bitmap index creation and maximize indexing throughput
Balkesen et al. [7]	N/A	TPC-H	A simple DPU without cache coherence, and data movement between DRAM and cores is managed by an on-chip programmable data movement engine
Sun et al. [179]	Up to 2.8x performance improvement and 2.87x energy saving	TPC-H	A data filter based on inherent parallelism provided by FPGA and can be turned on/off according to selectivity
Moghaddamfar et al. [130]	An average of 5x speedup against MonetDB	N/A	The sort phase is transformed into partial aggregation with higher data reduction to improve RAM bandwidth utilization and performance

reusability refers to the ability to use previously computed or stored resources as required. Resource sharing is often associated with multiple queries that have similar selection predicates, common table access, and join tasks. The basic idea behind both techniques is to avoid resource wasting and increase resource utilization, i.e., maximizing the value of a given amount of resource.

Query optimization is a CPU-intensive process and a necessary precondition for query execution. One of its most important principles is that the time cost of query optimization should not exceed that of query execution. To speed up query processing in commercial DBMSs, functions are generally provided to cache previous used plans for incoming queries. And to reuse cached plans, the optimizer needs to do textual matching among query statements. However, due to the declarative nature of SQL, different textual query statements may result in the same result, which adds to the difficulty of identifying plans with reusability. In addition, traditional query matching for reusing plans is extremely rigorous: a cached plan will be reused only if the prospective queries are textually similar to the stored queries.

To further enhance the plan reusing facility, a tool named PLASTIC is designed to extend the scope of plans for reusability by selecting plans though incremental clustering [54, 172]. Queries are grouped into clusters and each cluster has a persistently stored template that represents an execution plan generated by query optimizer. PLASTIC can reuse plan for incoming queries that share a common operator tree even if these queries are different in projection, selection and join predicates. By identifying whether incoming queries have identical plan templates, the possibility of reusing a cached plan is improved by PLASTIC with a 90% accuracy of plan selection.

In a following work of Sarda and Haritsa [160], a further augmented PLASTIC is introduced by integrating with new characteristics such as an augmented query feature vector, sized clusters, and a query classifier based on decision tree. To better evaluate the similarity among queries, query feature vector is used to capture query characteristics such as table access paths, index types, query structures, and query statistics. The tree-based classifier is used to fasten query clustering. And instead of fixed-size clusters, variable-sized clustering is used to reduce redundant plan space and adapt to dynamics of plan selection.

During query optimization, a query is generally divided into its sub-expressions. Galindo-Legaria and Waas [51] propose an approach to detect frequently used sub-expressions in a query workload to facilitate resource reusing. By storing encodings of sub-expressions in spite of no matching materialization is found, the usage statistics counters will be updated and can be used to found the most frequently occurring sub-expressions for subsequently queries.

Lang and Patel [104] describe a technique called QED to improve energy efficiency for query execution. QED is realized by introducing explicit delays at a workload level. QED can be utilized to different queries that share common components, for example, two queries in a batch job that both need to extract data from a same table. Also, QED is validated by a batch of simple select-only queries that read data from a same table while with different ranges of selection predicates. However, when queries are complex, a variety of operations will be included in a batch, which makes it more difficult to find an optimal way to regroup and rearrange operations and queries. Additionally, QED is restricted to scenarios where users are more patient and do need to be answered immediately, for example in non-interactive cases.

Meza et al. [126] present an approach to save energy for decision support systems. First, a power breakdown of the system is provided. The authors found that the storage subsystem, responsible for over a half of the system's total power consumption, is the largest power-consuming component. By repartitioning the database across fewer disks, unnecessary disks (i.e., over-provisioning storage resource) can be removed or turned off to save energy. Their experimental results show that 45% power savings can be achieved with only 5% peak performance degradation.

By merging and aggregating queries according to their query region, attribute, time duration, and frequency, Nan and Li [134] propose an energy-efficient query management scheme. The scheme can be used to reduce the energy cost for executing common tasks shared by a group of queries. And by executing common subexpression only once at the gateway side, communication overhead can also be minimized to further reduce energy consumption. Similarly, Bestgen et al. [15] present an energy-efficient technique by aggregating and queuing queries according to storage devices required for executing queries.

Researchers have studied the problem of multiple query optimization (MQO) in database systems for many years. One of its main objectives is to identify common tasks among queries [169]. Dokeroglu et al. [41, 42] provide a set of heuristic algorithms with adapted MQO to construct an energy-efficient global plan for concurrent queries in a batch to enhance resource sharing and improve resource utilization. Also, a cost model including communication overhead is built to facilitate decision making in cloud databases.

The idea of batching queries that access the same data or having common operations may not be practical in real-time databases. Kang [88, 89] propose a heuristic-based approach to deal with the constraints of deadline miss ratio and limited energy for real-time databases. To avoid the risk of jeopardizing timeliness, the proposed real-time aggregation approach can queue read operations of transactions and queries according to their priority

Table 9. Summary of energy-efficient techniques that based on reusing and sharing resources

Authors	Results	Workloads	Characteristics
Ghosh et al. [54]; Sengar and Haritsa [172]	90% accuracy of plan selection	TPC-H	Reuse plans for incoming queries even if queries are different, and queries are grouped into individual clusters that has persistently stored templates for plans
Sarda and Haritsa [160]	N/A	TPC-H	Extend the scope, usability, and efficiency of earlier proposed PLASTIC
Galindo-Legaria and Waas [51]	N/A	N/A	Detect frequently used sub-queries in a workload to facilitate resource reusing
Lang and Patel [104]	54% energy saving and 43% performance degeneration	TPC-H	Improve energy efficiency for a batch of queries by introducing explicit delays
Meza et al. [126]	save 45% power with 5% performance degradation	TPC-H	Repartition the database using less storage resource and turn off unused ones
Nan and Li [134]	26%∼42% energy saving	N/A	Reduce common tasks for a group of queries by merging and aggregating queries and executing common tasks only once
Bestgen et al. [15]	N/A	N/A	Aggregate queries according to storage devices that required for certain query processing
Dokeroglu et al. [41, 42]	N/A	TPC-H	A set of heuristic algorithms with adapted MQO to facilitate the construction of a global plan for concurrent queries in a batch
Kang [88, 89]	38% and 52% improvement of deadline miss ratio and power	N/A	A heuristic-based approach that queues read operations of transactions and queries according to their priority in a non-ascending way

(i.e., deadline) in a non-ascending way. Therefore, queues that are ready can be executed as much as possible without duplicating data access, which in turn reduce energy consumption as well as guarantee a decent deadline.

Here, we present a summary of reviewed techniques that aimed at reusing and sharing various database resources. Table 9 contains author names, publication year, experimental results, benchmarks, and main characteristics of above reviewed approaches.

5.4 Cluster Level Energy Management

There is no universal definition of the term database cluster. Generally, a database cluster is a group of individual database servers that are connected together by high speed networks. In Section 5.3, we have discussed energy-saving techniques that were proposed for standalone servers. However, the energy usage pattern of server clusters, especially those that are used to handle large data volumes and massive access requests in large-scale Internet services, has very different characteristics [8, 36]. Most of the time, servers in clusters are operating at relatively low utilization levels (i.e., between 10% and 50%), which represents a non-proportional relationship between utilization and energy consumption. This underutilization has motivated a number of researchers to provide techniques for cluster-level energy management in large-scale deployments. Valentini et al. [188] and Orgerie et al. [137] both present studies on low-power techniques for general server clusters, while this section serves as a complement with a focus on energy-efficient database clusters.

Generally, techniques for energy management can be generally divided into two major categories: 1) static energy management (SEM) and 2) dynamic energy management (DEM). SEM refers to using low-power hardware techniques to reduce peak power and/or the total energy cost of clusters, as well as keep an acceptable performance that usually specified by SLAs. Note that for a typical database cluster, processor and memory are currently considered as two major optimization objectives of which the range of dynamic power is much higher than that of the remaining components, and thus provide more energy-saving opportunities. DEM mainly refers to two approaches: 1) using online-adaptation techniques to dynamically change hardware power states and 2) utilizing techniques, for example load balancing, to save energy while taking into consideration of the current resource demands as well as the dynamic feature of cluster states. A classification and summary of different energy

management techniques for database clusters is given in Table 10. Also, each of the cluster-level approaches will be discussed in detail below.

5.4.1 Static Energy Management Techniques. Caulfield et al. [26] propose an energy-efficient cluster architecture called Gordon. Gordon is a combination of low-power processors and flash memory, which can be applied to parallel data-centric applications. And to fully utilize efficient characteristics of flash-based devices, Gordon uses a novel flash translation layer that closely links processors and the flash array over a simple interconnect. Compared with disk-based clusters, the experimental results reveal that Gordon systems can deliver 1.5x performance and 2.5x energy efficiency.

To provide better I/O throughput and energy efficiency, Szalay et al. [180] also explore the potential of building low-power architecture using energy-efficient processors and solid-state disks. Their prototypes are developed on Amdahl blades and can be used as a guideline to decide a smallest CPU throughput for data-intensive workloads that dominated by sequential reads.

Redesigning cluster architecture with wimpy nodes is introduced by Andersen et al. [3]. These nodes are often more energy-efficient due to the low-cost and low-power features of hardware components (i.e., low-end processors and small-size flash storage). This new architecture tends to be a competitive replacement of traditional high energy-consuming server nodes. However, compared with traditional nodes, these low-end nodes are far less competitive in performance [78, 106]. As a consequence, a small cluster of traditional nodes might need to be replaced by a large cluster of wimpy nodes.

Lang et al. [106] present an investigation on performance variations of increasingly scale-out wimpy-node cluster when handling complex database workload. Also, detailed comparisons are made between traditional nodes and wimpy nodes from two aspects: 1) price/performance of individual node for processing partitioned data and 2) diminishing returns as a result of startup-interference-skew factors when scaling up parallelly. The experimental results suggest that scale-out wimpy-node clusters are probably not cost-efficient alternatives with equivalent performance when running complex workloads, since the overhead introduced by parallel data processing is sufficient to vanish the benefit of low-end nodes.

Vasudevan et al. [190] further explore the performance of FAWN (i.e., fast array of wimpy nodes) with various workloads. The potential scaling capacity and speed for both memory and the second storage are evaluated. The experimental results reveal that wimpy-node cluster is more efficient than traditional cluster on I/O-intensive workload. However, exceptions lie in parallel data processing, which suggests a same conclusion as Lang et al. [106]. Similarly, Schall and Härder [164] investigate performance and energy efficiency of OLTP and OLAP queries on a cluster of wimpy nodes against a single brawny server. Their experimental results show that for low- and middle-intensive workloads significant power reduction can be achieved with small performance degeneration.

Loghin et al. [119] provide insights of future improvement for cluster designs by studying the big data workloads on small nodes in comparison with conventional big nodes. ARM and Xeon nodes are evaluated to compare their execution time as well as energy cost. I/O-intensive workloads consume less energy on Xeon-node clusters while CPU-intensive workloads show better energy efficiency on ARM-node clusters. In-addition, ARM-node clusters always show better energy efficiency when running database query workloads. Their experimental results draw a conclusion that the one-size-fits-all rule does not apply in small-node clusters or big-node clusters when running workloads with the objective of achieving better energy efficiency.

Wimpy-node cluster can be used to provide power-efficient workload processing while its performance and energy efficiency has been less optimized. Mühlbauer et al. [133] present Hyper, an in-memory database system that is suitable for processing OLAP and OLTP workloads with adaption to both brawny- and wimpy-node clusters. Through a hardware-supported snapshotting mechanism that based on POSIX system, Hyper can decouple mission-critical OLTP into time-consuming OLAP. HyPer is independent of the underlying platforms in term of wimpy cluster or brawny cluster. And to overcome the shortcomings of traditional iterator model, it

uses a different query compilation strategy based on LLVM compiler framework [135]. Since HyPer has a small memory footprint, it is applicable to mobile and embedded database systems.

Although wimpy-node cluster can be used to alleviate power-hunger server nodes, they are less efficient in complex and non-parallelizable workloads [106]. Sirin et al. [175] propose an alternative by using a commercial implementation of ARM Cortex-A57 to support server-grade applications. A high-performance Intel Xeon processor is compared with a recent ARM processor in term of their power cost, throughput, and latency when running OLTP workloads. The experimental results indicate that, compared with Xeon, ARM is 4x~5x more power-efficient while with 1.7x~3x lower throughput, and finally resulting in up to 9x higher energy efficiency than Xeon. In addition, the large power overhead of Xeon makes it far from energy-proportional while ARM obtained approximate energy proportionality due to the almost linear relationship between its utilization and power consumption.

István et al. [84] present a distributed storage cluster named Caribou with specialized hardware based on FPGAs. Caribou can support access to DRAM-NVRAM storage using a simple interface through traditional TCP/IP connectivity. Using internal dataflow parallelism that integrated within FPGA platform, each storage is able to support high-bandwidth processing with low latency. Fault tolerance is also provided by transparent data replication and partition among Caribou nodes for large-scale deployments. To provide near the data processing, Caribou is designed to support previously selection predicate operations before sending it to the processing nodes. Therefore, part of the computation can be offloaded to storage nodes and executed parallelly on FPGAs with little performance degeneration.

The ever-increasing size of blockchain data as well as the inefficient caching structure have led to poor query performance of blockchain applications, especially for the edge devices in IoT scenarios. Sanka et al. [159] present a hybrid and distributed caching architecture based on FPGA and NoSQL database (Redis) to improve scalability and reduce power consumption for blockchain servers. The NoSQL database is treated as the second caching layer to improve performance when cache miss happens on the first caching layer of FPGA due to their limited memory. Finally, FPGA and the NoSQL database can work cooperatively to improve performance as well as reduce the high cost of power and system resources introduced by the NoSQL caching layer.

5.4.2 Dynamic Energy Management Techniques. Wang and Chen [193] describe a cluster-level power controlling loop to shift and share power among individual servers in the cluster according to their prospective performance requirements. And hence, a given power budget of the entire cluster can be guaranteed without performance degeneration. Instead of heuristic approaches, a rigorous feedback control theory is used by the controlling loop to provide precisely control of power states of servers and to dynamically change frequency levels of processors. In addition, to better adapt to cluster-level environment, a multiple-input and multiple-output controlling algorithm is used to simultaneously manage servers' power cost in a small-scale cluster.

Individual and a small set of nodes are unlikely to be completely idle even during low-service demand periods, since both data and computation load are distributed among nodes. Fortunately, data replication provides an opportunity to power down nodes without losing data access [19]. However, well-designed replication and power-down strategy are both needed to avoid load imbalance on remaining active nodes. Moreover, data replication should work together with energy-efficient load balancing and task scheduling techniques to improve resource utilization, while minimizing energy consumption and maximizing throughput [53, 148].

Lang et al. [105] propose to investigate the interactions among load balancing, data replication, and energy management strategies at a cluster level. Given massive works that dedicated to designing replication schemes, the Chained Declustering (CD) technique [82] is found to be suitable for ensuring availability as well as fitting the need of generating a balanced and energy-efficient cluster. The CD allows multiple faults along the chain on nonadjacent nodes. Two CD-based schemes that have different power down/up schemes are presented and evaluated: 1) the dissolving chain (DC) that supports binary cuts and powers down node sequentially; 2) the

blinking chain (BC) that supports general cuts and provides better load balancing with energy efficiency, while its state-transition cost might be significant when system utilization changes frequently and greatly.

Precedence-constrained parallel tasks may have slack time for their execution. Wang et al. [192] provide a green parallel task scheduling technique for homogeneous clusters by taking advantages of the DVFS technique and green SLAs. Generally, SLAs are designed to meet the peak demand, and thus often provide slack time that permits additional response time penalty. After negotiating with users, the slack time within an affordable limit for non-critical tasks can be identified to help set the DVFS. Even though scaling down processors will increase the execution time of these tasks, significant energy savings can be achieved.

In the absence of new hardware with energy proportionality, WattDB provides a possibility of constructing approximate energy-proportional database cluster with off-the-shelf hardware [161, 163, 167]. WattDB is a distributed cluster coordinated by a dedicated master node. Each node in WattDB cluster can provide feedback about its real-time resource utilization to a coordinator. And then this aggregated information is used by the coordinator to dynamically switch nodes on and off for adapting to current workload. However, dynamic cluster reconfiguration (i.e., to scale in/out a cluster) requires balanced data redistribution among active nodes to avoid hotspots or bottlenecks on individual nodes. A following work of Schall and Härder [165] propose an energy-efficient data repartitioning and redistribution scheme by adapting the physiological partitioning (PLP [181]) method to a WattDB cluster.

Lang et al. [102] is the first work that aimed at building a green database cluster by exploring the architectural design space of parallel database clusters. By varying cluster size as well as cluster design, the interaction between the inherent non-linear scalability and energy efficiency of processing parallel data is investigated. And to study tradeoffs between performance and energy among various cluster designs, a metric called EDP (energy delay product, i.e., energy delay) is defined. Also, to further understand key factors of designing energy-efficient DBMS clusters, a model called P-store is developed to predict performance and energy efficiency for a database cluster with different node configurations.

Xie [199] describes an energy-efficient query distribution strategy for a heterogeneous database cluster based on DVFS. Firstly, servers of the cluster are divided into a few sets with different CPU frequency, and then queries are also regrouped according to their resource consumption patterns like CPU-intensive or I/O-intensive. Finally, concurrent queries can be allocated to different nodes with distinct CPU frequency to reduce energy consumption.

Zhang et al. [213] present a green distributed database cluster called Mega-KV for in-memory key-value store. Based on their observations that the homogeneous multicore CPUs are failed to support massive data parallelism and high memory bandwidth, Mega-KV cluster is developed and implemented on a heterogeneous CPU–GPU architecture. To alleviate the memory access overhead, GPUs is used by each node to offload and accelerate index operations, since index is identified as one of the main overheads for query processing. By utilizing GPUs, high memory bandwidth, lower latency, higher throughput and scalability can be achieved. What is more, by dynamically scaling the frequency of CPU and GPU nodes, higher energy efficiency in term of 299 thousand operations per Watt can be obtained.

Molka and Casale [131] propose an energy-efficient in-memory database cluster that aimed at two issues including 1) server assignment and 2) resource allocation. The proposed solution is based on a hybrid algorithm combining the best-fit decreasing [87] and genetic algorithms [211].

By comparing algorithms described in [23] with two energy-aware heuristics and two energy-blind heuristics, Casalicchio et al. [24] provide several insights that can be used to build auto-scaling energy-efficient Cassandra clusters. Their main findings are that virtual cluster allocation focused on energy saving and better resource utilization can significantly affect cluster reliability. Scaling out cluster is very slow and therefore cannot adapt to throughput surge, while scaling up can be an effective alternative.

To address the problem that horizontal scaling cannot timely respond to workload variations, Lombardi et al. [120] propose PASCAL a proactive architecture that can be used for Cassandra clusters to automatically as well

Table 10. Taxonomy of energy management techniques at the cluster-level

Category	Subcategory	Works
Static Energy Management	Low-power Processor and Flash Memory	Caulfield et al. [26]; Andersen et al. [3]; Szalay et al. [180]]
	Wimpy Node vs. Brawny Node	Vasudevan et al. [190]; Lang et al. [106]; Schall and Härder [164]; Loghin et al. [119]; Sirin et al. [175]; Mühlbauer et al. [133]
	FPGAs and DRAM-NVRAM Storage	István et al. [84]; Sanka et al. [159]
Dynamic Energy Management	DVFS-based	Wang and Chen [193]; Xie [199]; Zhang et al. [213]; Wang et al. [192]
	Design Space of Database Cluster	Lang et al. [102]
	Power-mode Switching	Zhou et al. [216, 218]
	Node Turning On/Off of Cluster Scaling	Lang et al. [105]; Schall and Hudlet [167]; Schall and Härder [161, 163]; Casalicchio et al. [24]; Lombardi et al. [120]
	Server Assignment and Resource Allocation	Molka and Casale [131]
	Data Allocation	Schall and Härder [165]

as timely scale in/out. PASCAL is mainly composed of a system performance estimator, a workload forecaster, a decision module, and a configuration manager. To avoid the issue of over-provisioning, the decision module can compute a minimum amount of system resources to satisfy an incoming workload and an expected performance that derived respectively from the workload forecaster and performance estimator. Finally, the configuration manager receives scaling actions from the decision module and applies the right configuration to the cluster in time.

Zhou et al. [216, 218] propose to use data prefetching and caching strategies to save energy for database clusters. The authors provide a skewed scheme for running workload in a cluster composed of a set of hot and cold nodes. Firstly, popular data and unpopular data are fetched and kept in hot and cold nodes separately. Then cold nodes can be switched to lower-power modes in longer time periods to reduce energy cost, and queries can be processed faster by assigning to hot nodes. In addition, the number of and the overhead of transitions among various power modes can be further reduced for energy conservation.

5.5 Open Issues and Challenges in Energy Management

In this section, we present multiple future directions for energy conservation in database systems. As we have observed from surveyed literature that researchers tried to improve energy efficiency from several aspects, but a number of issues are still unaddressed or underestimated. We have tried to separately address those open issues and future directions as following.

5.5.1 Multi-objective Query Optimization. Query optimization is one of the most studied problems in database systems, and therefore has drawn a lot of attention as a promising direction for energy management in databases. To satisfy the increasingly demanding requirement of energy conservation, other than performance, energy efficiency has already become an important optimization goal. Consequently, for typical database queries and requests, performance in term of response time and throughput is not the only criterion for evaluation. Most researchers have treated this new query optimization problem as tradeoffs between performance and energy consumption. Some researchers have explored possible tradeoffs between result accuracy and energy consumption. Trade-offs between various resources and metrics have begun to be considered as an interesting direction in

realizing green database systems. Note that not all of these trade-offs discussed in this article are new, however, until now, they have tended to be studied between only two metrics.

With the increasing diversification demand from both customers and service providers, there is a need for databases to support multi-objective query optimization. Also, the goal of query optimization will be shifted to select correct trade-offs among various metrics to satisfy multi-criteria quality of service that specified by users. We believe that the possibility of trade-offs among different metrics and resources should be investigated, as well as the complex correlation among these metrics, and their combined impact on query processing.

Since different applications may have different metrics to evaluate performance and cost, and different scenarios may have diverse requirements, a designer or a customer should be able to decide trade-offs on a case by case basis. Moreover, there is a need to create a standard to define and describe important information in a comprehensive and understandable way to facilitate delicate trade-offs. Additionally, user interface is required to provide interactions and to allow users to describe their desired optimization criteria for a given workload. Note that this multi-objective optimization problem, though can be intuitive for queries, also need to be explored in physical design of databases [56, 73, 108, 154, 155].

5.5.2 In-memory Databases. In the age of big data, today's database systems are required to efficiently handle mass data to meet performance requirements of various applications. Compared with traditional disk-oriented databases, main-memory databases are able to provide significant performance improvement since data can be stored entirely in DRAM to avoid costly disk I/O of queries. Over the past few years, with the increasing densities of DRAM and its falling cost, combined with the growing appetite of systems with large-memory, in-memory databases are becoming common.

Memory should be considered as an important energy-consuming component and optimization objective in current database systems. It has been evaluated that up to 40% energy were consumed by memory systems, and in larger configurations, memory power consumption overshaded that of the CPUs [9, 109, 194]. Furthermore, for main-memory databases, memory itself will accounts for a larger share of system's total energy consumption [92]. However, energy management techniques that aimed at memory systems have been less studied. This may be partly due to the lack of memory power model and the functionality that supports memory power mode transition. Consequently, we suggest that more research should be focused on energy conservation of main-memory databases as well as better memory power management.

5.5.3 NoSQL Databases. The development of the Internet has pushed database research into a new era. Traditional relational databases are facing challenges brought by big data, which refers to data volume (from terabyte to petabyte), data variety (structured, semi- and un-structured), and the high growing speed. Generally, non-relational databases, usually called NoSQL databases, have been emerging along with some leading enterprises such as Google and Amazon. Complementing a number of limitations of RDBMSs, NoSQL are now widely used in many Internet companies [177]. Since NoSQL databases are designed for large-scale data storing and high concurrent data processing, the scalability of NoSQL also indicates massive energy consumption and thus pose higher demand for energy efficiency and energy proportionality [111, 178]. However, many, although not all, energy-efficient techniques discussed in this article are proposed and developed in the context of relational database systems. We believe that techniques that focused on RDBMSs should be reconsidered and re-examined for NoSQL and more efforts need to be paid on building green NoSQL databases [123]. Note that graph databases have become the highest concern in recent years and a number of researchers have made their efforts on energy-efficient graph processing [46, 52, 70, 214, 215].

5.5.4 Adaptive Energy Management by DBMSs. Most existing energy-saving policies and techniques for database systems are application-specific. This is partially due to the different resource usage pattern among various

database applications and workloads. From the trend observed through this survey, we believe that more energy savings will be achieved if these techniques can be controlled and managed by the DBMS software.

There are three main reasons for why the DBMSs are more suitable to do energy management for themselves. Firstly, database systems, as a basic component of almost every service, have been one of the main energy consumers, as well as an important target for energy management. Secondly, a DBMS can provide rich information about the system and its workload. Some information is generated before workload execution, for example, a number of query plans will be generated to fasten query processing. While certain information is available at run time, indicating execution behaviors of each transaction/query. This feature of DBMSs is important since being aware of and understanding energy usage of a system and its workload is a prerequisite for application-level energy management. In addition, this feature also indicates that the information required for making the right decision on energy management can be found within database systems, while cannot be captured by low-level hardware or the operating system.

The last but not least, DBMSs support extensive run-time tracing facilities and most of them are user-controllable, which means various types of information can be produced, collected, and maintained by DBMSs whether for run-time or after-execution analysis. This feature can further help to predict future workload based on historical data and to better support highly dynamic environments [121]. Furthermore, various information that obtained from DBMSs can be combined in a meaningful way to gain insights on application behaviors to facilitate decision making and fine-grained energy control. These features of DBMSs will also make them desirable platforms for developing and evaluating general or specific energy-efficient techniques. To conclude, since DBMSs have more knowledge, we suggest that both hardware and software knobs that may influence energy consumption should be managed by database software in the future research.

5.5.5 DBMSs and New Hardware. The next stage of database development will be tightly linked with new hardware features. Memory with larger capacity, NVM technology, and hardware accelerators like GPUs and FPGAs are changing the landscape of database systems. The evolving of hardware technology will push the databases toward customized service with higher energy efficiency based on combined efforts of hardware-software co-design and collaboration. However, only a limited number of works have made their efforts in this direction. Most existing energy-related techniques are either focused on the software level or the hardware level. And currently, power-efficient hardware features are generally controlled by low-level components implemented in the OS or the hardware, for example the Powersave governor and scaling driver that supported by the CpuFreq Architecture. However, they all lack the capability of controlling these features appropriately.

Compared with the OS and the hardware itself, the database software has more advantages as we discussed in Section 5.5.4. Consequently, DBMSs are expected to make the most of hardware features and result in better energy efficiency. Furthermore, with the availability of power-manageable hardware and their more sophisticated features, there exists substantial opportunities for building greener databases.

Note that performance of certain database applications may be sensitive to resource allocation and power scaling of certain hardware [171], along with the increasing complexity of database applications, fine-grained and dynamic controlling is needed to decide whether, when as well as to what extend individual hardware or node should be turning off/on and scaling down/up. Moreover, standardized hardware/software interface is required to enable software-controlled energy management of hardware components. We believe that energy-related hardware knobs controlled by DBMSs can lead to better energy-efficient outcome and we suggest future researchers investigate more on hardware and software combined approaches.

5.5.6 Energy-aware Data Management for Fog/Edge Computing Environments. Compared with traditional server systems in cloud datacenters, fog/edge computing platforms, closing to the ground, can be used to meet the requirements of location awareness as well as low latency for time-critical and real-time IoT applications [18]. Nevertheless, edge/mobile devices have inherent resource constraints, such as energy consumption, networking,

computing and storage, contradicting with the ever-increasing popularity of fog/edge computing. Recently, researchers have been actively investigating the energy-efficient strategies that are aimed at offloading data/tasks from energy-constrained edge devices to nearby or remote resource-rich platforms, such as powerful servers, a cluster or the cloud, for execution [32]. However, when it comes to improving energy efficiency of edge devices in fog/edge computing, we argue that, as indispensable and critical components, database systems will play a vital role in the future. The first step taken by recent studies starts from exploring the impact of relational and NoSQL database systems on the performance and energy consumption of various edge devices when processing different workload patterns [115, 124]. This interesting research area calls for more efforts to gain insightful observations on how to provide energy-aware data management for fog/edge computing environments.

6 CONCLUSION

During the last several decades, in the research field of database technology, the objective of performance had been widely studied. However, with the increasing awareness of going green in IT systems, the database community is also facing new challenges raised by energy constraints. It has been ten years since the proposal of constructing green database systems, and this research area is attracting more and more researchers to join and make a contribution. Many researchers have made their efforts and contributions to this area. And after a decade of research, it is a time to review the past and look into the future. This survey paper gave a comprehensive and in-depth study on existing works that explicitly address energy management issues in database systems. Firstly, a level-wise decomposition of energy-saving efforts for green databases was performed. And hence, the main body of this survey was naturally divided into two parts: energy modeling and energy management. Then, we described, compared, and discussed existing works that fall into the two levels separately. We classified each of the reviewed paper into the taxonomy that was presented. We also provided and discussed open issues and challenges to be future explored on realizing energy-efficient database systems. Based on these insights observed through our study, we hope that this survey will attract more attention and motivate significant growth in energy modeling and management for database systems in the near future.

REFERENCES

[1] Rakesh Agrawal, Anastasia Ailamaki, Philip A Bernstein, Eric A Brewer, Michael J Carey, Surajit Chaudhuri, AnHai Doan, Daniela Florescu, Michael J Franklin, Hector Garcia-Molina, et al. 2008. The Claremont report on database research. *ACM Sigmod Record* 37, 3 (2008), 9–19.

[2] Anastasia Ailamaki. 2015. Databases and hardware: The beginning and sequel of a beautiful friendship. *Proceedings of the VLDB Endowment* 8, 12 (2015), 2058–2061.

[3] David G Andersen, Jason Franklin, Michael Kaminsky, Amar Phanishayee, Lawrence Tan, and Vijay Vasudevan. 2009. FAWN: A fast array of wimpy nodes. In *Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles*. 1–14.

[4] Raja Appuswamy, Matthaios Olma, and Anastasia Ailamaki. 2015. Scaling the memory power wall with dram-aware data management. In *Proceedings of the 11th International Workshop on Data Management on New Hardware*. 1–9.

[5] Oliver Arnold, Sebastian Haas, Gerhard Fettweis, Benjamin Schlegel, Thomas Kissinger, and Wolfgang Lehner. 2014. An application-specific instruction set for accelerating set-oriented database primitives. In *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*. 767–778.

[6] Chang S Bae and Tayeb Jamel. 2011. Energy-aware memory management through database buffer control. In *Proc. Workshop on Energy-Efficient Design*. Citeseer.

[7] Cagri Balkesen, Nitin Kunal, Georgios Giannikis, Pit Fender, Seema Sundara, Felix Schmidt, Jarod Wen, Sandeep Agrawal, Arun Raghavan, Venkatanathan Varadarajan, et al. 2018. Rapid: In-memory analytical query processing engine with extreme performance per watt. In *Proceedings of the 2018 International Conference on Management of Data*. 1407–1419.

[8] Luiz André Barroso and Urs Hölzle. 2007. The case for energy-proportional computing. *Computer* 40, 12 (2007), 33–37.

[9] Luiz André Barroso and Urs Hölzle. 2009. The datacenter as a computer: An introduction to the design of warehouse-scale machines. *Synthesis lectures on computer architecture* 4, 1 (2009), 1–108.

[10] Daniel Bausch, Ilia Petrov, and Alejandro Buchmann. 2012. Making cost-based query optimization asymmetry-aware. In *Proceedings of the Eighth International Workshop on Data Management on New Hardware*. 24–32.

[11] Andreas Becher, Florian Bauer, Daniel Ziener, and Jürgen Teich. 2014. Energy-aware SQL query acceleration through FPGA-based dynamic partial reconfiguration. In *2014 24th International Conference on Field Programmable Logic and Applications (FPL)*. IEEE, 1–8.
[12] Anton Beloglazov, Rajkumar Buyya, Young Choon Lee, and Albert Zomaya. 2011. A taxonomy and survey of energy-efficient data centers and cloud computing systems. In *Advances in computers*. Vol. 82. Elsevier, 47–111.
[13] Luca Benini, Alessandro Bogliolo, and Giovanni De Micheli. 2000. A survey of design techniques for system-level dynamic power management. *IEEE transactions on very large scale integration (VLSI) systems* 8, 3 (2000), 299–316.
[14] Andreas Berl, Erol Gelenbe, Marco Di Girolamo, Giovanni Giuliani, Hermann De Meer, Minh Quan Dang, and Kostas Pentikousis. 2010. Energy-efficient cloud computing. *The computer journal* 53, 7 (2010), 1045–1051.
[15] Robert Joseph Bestgen, Wei Hu, Shantan Kethireddy, Andrew Peter Passe, and Ulrich Thiemann. 2011. Aggregating database queries. US Patent 7,958,158.
[16] Robert J Bestgen, Wei Hu, Shantan Kethireddy, Andrew P Passe, and Ulrich Thiemann. 2012. Generating database query plans. US Patent 8,312,007.
[17] Robert J Bestgen, Wei Hu, Shantan Kethireddy, Andrew P Passe, and Ulrich Thiemann. 2015. Organizing databases for energy efficiency. US Patent 9,189,047.
[18] Flavio Bonomi, Rodolfo Milito, Jiang Zhu, and Sateesh Addepalli. 2012. Fog computing and its role in the internet of things. In *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*. 13–16.
[19] Dejene Boru, Dzmitry Kliazovich, Fabrizio Granelli, Pascal Bouvry, and Albert Y Zomaya. 2015. Energy-efficient data replication in cloud computing datacenters. *Cluster computing* 18, 1 (2015), 385–402.
[20] Tom Bostoen, Sape Mullender, and Yolande Berbers. 2013. Power-reduction techniques for data-center storage systems. *ACM Computing Surveys (CSUR)* 45, 3 (2013), 1–38.
[21] Richard E Brown, Richard Brown, Eric Masanet, Bruce Nordman, Bill Tschudi, Arman Shehabi, John Stanley, Jonathan Koomey, Dale Sartor, Peter Chan, et al. 2007. *Report to congress on server and data center energy efficiency: Public law 109-431*. Technical Report. Lawrence Berkeley National Lab.(LBNL), Berkeley, CA (United States).
[22] Christian Bunse, Hagen Höpfner, Essam Mansour, and Suman Roychoudhury. 2009. Exploring the energy consumption of data sorting algorithms in embedded and mobile environments. In *2009 Tenth International Conference on Mobile Data Management: Systems, Services and Middleware*. IEEE, 600–607.
[23] Emiliano Casalicchio, Lars Lundberg, and Sogand Shirinbab. 2016. Energy-aware adaptation in managed cassandra datacenters. In *2016 International Conference on Cloud and Autonomic Computing (ICCAC)*. IEEE, 60–71.
[24] Emiliano Casalicchio, Lars Lundberg, and Sogand Shirinbab. 2017. Energy-aware auto-scaling algorithms for Cassandra virtual data centers. *Cluster Computing* 20, 3 (2017), 2065–2082.
[25] Jared Casper and Kunle Olukotun. 2014. Hardware acceleration of database operations. In *Proceedings of the 2014 ACM/SIGDA international symposium on Field-programmable gate arrays*. 151–160.
[26] Adrian M Caulfield, Laura M Grupp, and Steven Swanson. 2009. Gordon: using flash memory to build fast, power-efficient clusters for data-intensive applications. *ACM Sigplan Notices* 44, 3 (2009), 217–228.
[27] Ulpian Cesana and Zhen He. 2010. Multi-buffer manager: Energy-efficient buffer manager for databases on flash memory. *ACM Transactions on Embedded Computing Systems (TECS)* 9, 3 (2010), 1–36.
[28] Srinivasan Chandrasekharan and Chris Gniady. 2018. Qamem: Query aware memory energy management. In *2018 18th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*. IEEE, 412–421.
[29] Surajit Chaudhuri. 1998. An overview of query optimization in relational systems. In *Proceedings of the seventeenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems*. 34–43.
[30] Maxime Colmant, Pascal Felber, Romain Rouvoy, and Lionel Seinturier. 2017. Wattskit: Software-defined power monitoring of distributed systems. In *2017 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*. IEEE, 514–523.
[31] Jason Cong, Mohammad Ali Ghodrat, Michael Gill, Beayna Grigorian, Karthik Gururaj, and Glenn Reinman. 2014. Accelerator-rich architectures: Opportunities and progresses. In *2014 51st ACM/EDAC/IEEE Design Automation Conference (DAC)*. IEEE, 1–6.
[32] Peijin Cong, Junlong Zhou, Liying Li, Kun Cao, Tongquan Wei, and Keqin Li. 2020. A survey of hierarchical energy optimization for mobile edge computing: A perspective from end devices to the cloud. *ACM Computing Surveys (CSUR)* 53, 2 (2020), 1–44.
[33] Fahimeh Dabaghi, Zeinab Movahedi, and Rami Langar. 2017. A survey on green routing protocols using sleep-scheduling in wired networks. *Journal of Network and Computer Applications* 77 (2017), 106–122.
[34] Howard David, Chris Fallin, Eugene Gorbatov, Ulf R Hanebutte, and Onur Mutlu. 2011. Memory power management via dynamic voltage/frequency scaling. In *Proceedings of the 8th ACM international conference on Autonomic computing*. 31–40.
[35] Howard David, Eugene Gorbatov, Ulf R Hanebutte, Rahul Khanna, and Christian Le. 2010. RAPL: Memory power estimation and capping. In *2010 ACM/IEEE International Symposium on Low-Power Electronics and Design (ISLPED)*. IEEE, 189–194.
[36] Miyuru Dayarathna, Yonggang Wen, and Rui Fan. 2015. Data center energy consumption modeling: A survey. *IEEE Communications Surveys & Tutorials* 18, 1 (2015), 732–794.

[37] Simon Pierre Dembele, Ladjel Bellatreche, and Carlos Ordonez. 2020. Towards Green Query Processing-Auditing Power Before Deploying. In *2020 IEEE International Conference on Big Data (Big Data)*. IEEE, 2492–2501.
[38] Simon Pierre Dembele, Ladjel Bellatreche, Carlos Ordonez, and Amine Roukh. 2020. Think big, start small: a good initiative to design green query optimizers. *Cluster Computing* 23, 3 (2020), 2323–2345.
[39] Jaeyoung Do, Yang-Suk Kee, Jignesh M Patel, Chanik Park, Kwanghyun Park, and David J DeWitt. 2013. Query processing on smart ssds: Opportunities and challenges. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*. 1221–1230.
[40] Jaeyoung Do, Donghui Zhang, Jignesh M Patel, David J DeWitt, Jeffrey F Naughton, and Alan Halverson. 2011. Turbocharging DBMS buffer pool using SSDs. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of data*. 1113–1124.
[41] Tansel Dokeroglu, Murat Ali Bayir, and Ahmet Cosar. 2015. Robust heuristic algorithms for exploiting the common tasks of relational cloud database queries. *Applied Soft Computing* 30 (2015), 72–82.
[42] Tansel Dokeroglu, Serkan Ozal, Murat Ali Bayir, Muhammet Serkan Cinar, and Ahmet Cosar. 2014. Improving the performance of Hadoop Hive by sharing scan and computation tasks. *Journal of Cloud Computing* 3, 1 (2014), 1–11.
[43] Simone Dominico, Eduardo C de Almeida, Jorge A Meira, and Marco AZ Alves. 2018. An elastic multi-core allocation mechanism for database systems. In *2018 IEEE 34th International Conference on Data Engineering (ICDE)*. IEEE, 473–484.
[44] Markus Dreseler, Thomas Kissinger, Timo Djürken, Eric Lübke, Matthias Uflacker, Dirk Habich, Hasso Plattner, and Wolfgang Lehner. 2017. Hardware-Accelerated Memory Operations on Large-Scale NUMA Systems.. In *ADMS@ VLDB*. 34–41.
[45] Hadi Esmaeilzadeh, Emily Blem, Renee St Amant, Karthikeyan Sankaralingam, and Doug Burger. 2011. Dark silicon and the end of multicore scaling. In *2011 38th Annual international symposium on computer architecture (ISCA)*. IEEE, 365–376.
[46] SM Faisal, Georgios Tziantzioulis, AM Gok, Nikolaos Hardavellas, S Ogrenci-Memik, and Srinivasan Parthasarathy. 2015. Edge importance identification for energy efficient graph processing. In *2015 IEEE International Conference on Big Data (Big Data)*. IEEE, 347–354.
[47] Xiaobo Fan, Wolf-Dietrich Weber, and Luiz Andre Barroso. 2007. Power provisioning for a warehouse-sized computer. *ACM SIGARCH computer architecture news* 35, 2 (2007), 13–23.
[48] Franz Färber, Sang Kyun Cha, Jürgen Primsch, Christof Bornhövd, Stefan Sigg, and Wolfgang Lehner. 2012. SAP HANA database: data management for modern business applications. *ACM Sigmod Record* 40, 4 (2012), 45–51.
[49] Tobias Flach. 2010. *Optimizing query execution to improve the energy efficiency of database management systems*. Technical Report. Citeseer.
[50] Gene F Franklin, J David Powell, Michael L Workman, et al. 1998. *Digital control of dynamic systems*. Vol. 3. Addison-wesley Menlo Park.
[51] Cesar Galindo-Legaria and Florian Waas. 2005. Automatic detection of frequently used query patterns in a query workload. US Patent App. 10/873,529.
[52] Abdullah Gharaibeh, Elizeu Santos-Neto, Lauro Beltrão Costa, and Matei Ripeanu. 2013. The energy case for graph processing on hybrid cpu and gpu systems. In *Proceedings of the 3rd Workshop on Irregular Applications: Architectures and Algorithms*. 1–8.
[53] Einollah Jafarnejad Ghomi, Amir Masoud Rahmani, and Nooruldeen Nasih Qader. 2017. Load-balancing algorithms in cloud computing: A survey. *Journal of Network and Computer Applications* 88 (2017), 50–71.
[54] Antara Ghosh, Jignashu Parikh, Vibhuti S Sengar, and Jayant R Haritsa. 2002. Plan selection based on query clustering. In *VLDB'02: Proceedings of the 28th international conference on very large databases*. Elsevier, 179–190.
[55] Sebastian Götz, Thomas Ilsche, Jorge Cardoso, Josef Spillner, Thomas Kissinger, Uwe Aßmann, Wolfgang Lehner, Wolfgang E Nagel, and Alexander Schill. 2014. Energy-efficient databases using sweet spot frequencies. In *2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing*. IEEE, 871–876.
[56] Goetz Graefe. 2008. Database servers tailored to improve energy efficiency. In *Proceedings of the 2008 EDBT workshop on Software engineering for tailor-made data management*. 24–28.
[57] Goetz Graefe, Haris Volos, Hideaki Kimura, Harumi Kuno, Joseph Tucek, Mark Lillibridge, and Alistair Veitch. 2014. In-Memory Performance for Big Data. *Proceedings of the VLDB Endowment* 8, 1 (2014).
[58] Jim Gray. 2007. Tape is dead, disk is tape, flash is disk, RAM locality is king. *Gong Show Presentation at CIDR* 15 (2007), 231–242.
[59] Jim Gray and Bob Fitzgerald. 2008. Flash Disk Opportunity for Server Applications: Future flash-based disks could provide breakthroughs in IOPS, power, reliability, and volumetric capacity when compared with conventional disks. *Queue* 6, 4 (2008), 18–23.
[60] Jorge Guerra, Wendy Belluomini, Joseph Glider, Karan Gupta, and Himabindu Pucha. 2010. Energy proportionality for storage: Impact and feasibility. *ACM SIGOPS Operating Systems Review* 44, 1 (2010), 35–39.
[61] Binglei Guo, Jiong Yu, Bin Liao, and Dexian Yang. 2015. SQL energy consumption modeling and optimization research. *J. Comput. Scie* 42, 10 (2015), 202–207.
[62] Binglei Guo, Jiong Yu, Bin Liao, and Dexian Yang. 2015. SQL execution power profiling and modeling. *J. Comput. Appl* 12, 33 (2015), 3362–3367.
[63] Binglei Guo, Jiong Yu, Bin Liao, and Dexian Yang. 2017. SQL energy consumption forecasting model based on database load status. *J. Comput. Scie* 44, 1 (2017), 208–213.

[64] Binglei Guo, Jiong Yu, Bin Liao, Dexian Yang, and Liang Lu. 2017. A green framework for DBMS based on energy-aware query optimization and energy-efficient query processing. *Journal of Network and Computer Applications* 84 (2017), 118–130.
[65] Chaopeng Guo, Jean-Marc Pierson, Hui Liu, and Jie Song. 2018. Frequency selection approach for energy aware cloud database. *IEEE access* 7 (2018), 1927–1942.
[66] Chaopeng Guo, Jean-Marc Pierson, Jie Song, and Christina Herzog. 2019. Hot-N-Cold model for energy aware cloud databases. *J. Parallel and Distrib. Comput.* 123 (2019), 130–144.
[67] Sebastian Haas, Oliver Arnold, Benedikt Nöthen, Stefan Scholze, Georg Ellguth, Andreas Dixius, Sebastian Höppner, Stefan Schiefer, Stephan Hartmann, Stephan Henker, et al. 2016. An MPSoC for energy-efficient database query processing. In *Proceedings of the 53rd Annual Design Automation Conference*. 1–6.
[68] Sebastian Haas and Gerhard P Fettweis. 2017. Energy-Efficient Hash Join Implementations in Hardware-Accelerated MPSoCs.. In *ADMS@ VLDB*. 26–33.
[69] Sebastian Haas, Stefan Scholze, Sebastian Höppner, Annett Ungethüm, Christian Mayr, René Schüffny, Wolfgang Lehner, and Gerhard Fettweis. 2017. Application-specific architectures for energy-efficient database query processing and optimization. *Microprocessors and Microsystems* 55 (2017), 119–130.
[70] Tae Jun Ham, Lisa Wu, Narayanan Sundaram, Nadathur Satish, and Margaret Martonosi. 2016. Graphicionado: A high-performance and energy-efficient accelerator for graph analytics. In *2016 49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. IEEE, 1–13.
[71] Theo Härder, Volker Hudlet, Yi Ou, and Daniel Schall. 2011. Energy efficiency is not enough, energy proportionality is needed!. In *International Conference on Database Systems for Advanced Applications*. Springer, 226–239.
[72] Theo Härder, Karsten Schmidt, Yi Ou, and Sebastian Bächle. 2009. Towards flash disk use in databases–Keeping performance while saving energy? *Datenbanksysteme in Business, Technologie und Web (BTW)–13. Fachtagung des GI-Fachbereichs" Datenbanken und Informationssysteme"(DBIS)* (2009).
[73] Stavros Harizopoulos, Mehul A Shah, Justin Meza, and Parthasarathy Ranganathan. 2009. Energy Efficiency: The New Holy Grail of Data Management Systems Research. *CIDR* 96, 3 (2009), 81–90.
[74] Ahmad Hassan, Dimitrios S Nikolopoulos, and Hans Vandierendonck. 2019. Fast and Energy-Efficient OLAP Data Management on Hybrid Main Memory Systems. *IEEE Trans. Comput.* 68, 11 (2019), 1597–1611.
[75] Ahmad Hassan, Hans Vandierendonck, and Dimitrios S Nikolopoulos. 2015. Energy-efficient in-memory data stores on hybrid memory hierarchies. In *Proceedings of the 11th International Workshop on Data Management on New Hardware*. 1–8.
[76] Ahmad Hassan, Hans Vandierendonck, and Dimitrios S Nikolopoulos. 2015. Software-managed energy-efficient hybrid DRAM/NVM main memory. In *Proceedings of the 12th ACM International Conference on Computing Frontiers*. 1–8.
[77] Bingsheng He. 2014. When data management systems meet approximate hardware: Challenges and opportunities. *Proceedings of the VLDB Endowment* 7, 10 (2014), 877–880.
[78] Urs Hölzle. 2010. Brawny cores still beat wimpy cores, most of the time. *IEEE Micro* 30, 4 (2010).
[79] Andy Hooper. 2008. Green computing. *Communication of the ACM* 51, 10 (2008), 11–13.
[80] Hagen Höpfner and Christian Bunse. 2009. Towards an Energy Aware DBMS-Energy Consumptions of Sorting and Join Algorithms.. In *Grundlagen von Datenbanken*. 69–73.
[81] Mohammad Ashraful Hoque, Matti Siekkinen, Kashif Nizam Khan, Yu Xiao, and Sasu Tarkoma. 2015. Modeling, profiling, and debugging the energy consumption of mobile devices. *ACM Computing Surveys (CSUR)* 48, 3 (2015), 1–40.
[82] Hui-I Hsiao and David J DeWitt. 1989. *Chained declustering: A new availability strategy for multiprocessor database machines*. Technical Report. University of Wisconsin-Madison Department of Computer Sciences.
[83] Mohsen Imani, Saransh Gupta, Sahil Sharma, and Tajana Simunic Rosing. 2018. NVQuery: Efficient query processing in nonvolatile memory. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 38, 4 (2018), 628–639.
[84] Zsolt István, David Sidler, and Gustavo Alonso. 2017. Caribou: Intelligent distributed storage. *Proceedings of the VLDB Endowment* 10, 11 (2017), 1202–1213.
[85] Jeff Janukowicz, David Reinsel, and John Rydning. 2008. *Worldwide solid state drive 2008-2012 forecast and analysis*. Technical Report. Technical Report 212736, IDC.
[86] Peiquan Jin, Yi Ou, Theo Härder, and Zhi Li. 2012. AD-LRU: An efficient buffer replacement algorithm for flash-based databases. *Data & Knowledge Engineering* 72 (2012), 83–102.
[87] David S Johnson. 1973. *Near-optimal bin packing algorithms*. Ph. D. Dissertation. Massachusetts Institute of Technology.
[88] Kyoung-Don Kang. 2016. Reducing deadline misses and power consumption in real-time databases. In *2016 IEEE Real-Time Systems Symposium (RTSS)*. IEEE, 257–268.
[89] Kyoung-Don Kang. 2018. Enhancing timeliness and saving power in real-time databases. *Real-Time Systems* 54, 2 (2018), 484–513.
[90] Woochul Kang and Jaeyong Chung. 2017. Energy-efficient response time management for embedded databases. *Real-Time Systems* 53, 2 (2017), 228–253.

[91] Woochul Kang, Sang H Son, and John A Stankovic. 2008. Power-aware data buffer cache management in real-time embedded databases. In *2008 14th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications*. IEEE, 35–44.
[92] Alexey Karyakin and Kenneth Salem. 2017. An analysis of memory power consumption in database systems. In *Proceedings of the 13th International Workshop on Data Management on New Hardware*. 1–9.
[93] Alexey Karyakin and Kenneth Salem. 2019. DimmStore: memory power optimization for database systems. *Proceedings of the VLDB Endowment* 12, 11 (2019), 1499–1512.
[94] Thomas Kissinger, Dirk Habich, and Wolfgang Lehner. 2018. Adaptive energy-control for in-memory database systems. In *Proceedings of the 2018 International Conference on Management of Data*. 351–364.
[95] Thomas Kissinger, Marcus Hähnel, Till Smejkal, Dirk Habich, Hermann Härtig, and Wolfgang Lehner. 2018. Energy-utility function-based resource control for in-memory database systems live. In *Proceedings of the 2018 International Conference on Management of Data*. 1717–1720.
[96] Barbara Kitchenham, O Pearl Brereton, David Budgen, Mark Turner, John Bailey, and Stephen Linkman. 2009. Systematic literature reviews in software engineering–a systematic literature review. *Information and software technology* 51, 1 (2009), 7–15.
[97] Onur Kocberber, Boris Grot, Javier Picorel, Babak Falsafi, Kevin Lim, and Parthasarathy Ranganathan. 2013. Meet the walkers accelerating index traversals for in-memory databases. In *2013 46th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. IEEE, 468–479.
[98] Fanxin Kong and Xue Liu. 2014. A survey on green-energy-aware power management for datacenters. *ACM Computing Surveys (CSUR)* 47, 2 (2014), 1–38.
[99] Mustafa Korkmaz, Martin Karsten, Kenneth Salem, and Semih Salihoglu. 2018. Workload-aware CPU performance scaling for transactional database systems. In *Proceedings of the 2018 International Conference on Management of Data*. 291–306.
[100] Mustafa Korkmaz, Alexey Karyakin, Martin Karsten, and Kenneth Salem. 2015. Towards Dynamic Green-Sizing for Database Servers.. In *ADMS@ VLDB*. 25–36.
[101] Mayuresh Kunjir, Puneet K Birwa, and Jayant R Haritsa. 2012. Peak power plays in database engines. In *Proceedings of the 15th International Conference on Extending Database Technology*. 444–455.
[102] Willis Lang, Stavros Harizopoulos, Jignesh M Patel, Mehul A Shah, and Dimitris Tsirogiannis. 2012. Towards energy-efficient database cluster design. *arXiv preprint arXiv:1208.1933* (2012).
[103] Willis Lang, Ramakrishnan Kandhan, and Jignesh M Patel. 2011. Rethinking query processing for energy efficiency: Slowing down to win the race. *IEEE Data Eng. Bull.* 34, 1 (2011), 12–23.
[104] Willis Lang and Jignesh Patel. 2009. Towards eco-friendly database management systems. *arXiv preprint arXiv:0909.1767* (2009).
[105] Willis Lang, Jignesh M Patel, and Jeffrey F Naughton. 2010. On energy management, load balancing and replication. *Acm Sigmod Record* 38, 4 (2010), 35–42.
[106] Willis Lang, Jignesh M Patel, and Srinath Shankar. 2010. Wimpy node clusters: What about non-wimpy workloads?. In *Proceedings of the Sixth International Workshop on Data Management on New Hardware*. 47–55.
[107] Sang-Won Lee, Bongki Moon, Chanik Park, Jae-Myung Kim, and Sang-Woo Kim. 2008. A case for flash memory SSD in enterprise database applications. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*. 1075–1086.
[108] Jeff LeFevre. 2014. *Physical design tuning methods for emerging system architectures*. University of California, Santa Cruz.
[109] Charles Lefurgy, Karthick Rajamani, Freeman Rawson, Wes Felter, Michael Kistler, and Tom W Keller. 2003. Energy management for commercial servers. *Computer* 36, 12 (2003), 39–48.
[110] Ding Li, Shuai Hao, Jiaping Gui, and William GJ Halfond. 2014. An empirical study of the energy consumption of android applications. In *2014 IEEE International Conference on Software Maintenance and Evolution*. IEEE, 121–130.
[111] Tiantian Li, Ge Yu, Xuebing Liu, and Jie Song. 2014. Analyzing the waiting energy consumption of nosql databases. In *2014 IEEE 12th International Conference on Dependable, Autonomic and Secure Computing*. IEEE, 277–282.
[112] Zhi Li, Peiquan Jin, Xuan Su, Kai Cui, and Lihua Yue. 2009. CCF-LRU: A new buffer replacement algorithm for flash memory. *IEEE Transactions on Consumer Electronics* 55, 3 (2009), 1351–1359.
[113] Jiang Lin, Hongzhong Zheng, Zhichun Zhu, Eugene Gorbatov, Howard David, and Zhao Zhang. 2008. Software thermal management of DRAM memory for multicore systems. *ACM SIGMETRICS Performance Evaluation Review* 36, 1 (2008), 337–348.
[114] Weiwei Lin, Fang Shi, Wentai Wu, Keqin Li, Guangxin Wu, and Al-Alas Mohammed. 2020. A taxonomy and survey of power models and power modeling for cloud servers. *ACM Computing Surveys (CSUR)* 53, 5 (2020), 1–41.
[115] Jian Liu, Kefei Wang, and Feng Chen. 2021. Understanding Energy Efficiency of Databases on Single Board Computers for Edge Computing. In *2021 29th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*. IEEE, 1–8.
[116] Xiaowei Liu, Jinbao Wang, Haijie Wang, and Hong Gao. 2013. Generating power-efficient query execution plan. In *2nd International Conference on Advances in Computer Science and Engineering (CSE 2013)*. Atlantis Press, 284–288.
[117] Xue Liu, Xiaoyun Zhu, Pradeep Padala, Zhikui Wang, and Sharad Singhal. 2007. Optimal multivariate control for differentiated services on a shared hosting platform. In *2007 46th IEEE Conference on Decision and Control*. IEEE, 3792–3799.

[118] Eric Lo, Carsten Binnig, Donald Kossmann, M Tamer Özsu, and Wing-Kai Hon. 2010. A framework for testing DBMS features. *The VLDB Journal* 19, 2 (2010), 203–230.
[119] Dumitrel Loghin, Bogdan Marius Tudor, Hao Zhang, Beng Chin Ooi, and Yong Meng Teo. 2015. A performance study of big data on small nodes. *Proceedings of the VLDB Endowment* 8, 7 (2015), 762–773.
[120] Federico Lombardi, Andrea Muti, Leonardo Aniello, Roberto Baldoni, Silvia Bonomi, and Leonardo Querzoni. 2019. Pascal: An architecture for proactive auto-scaling of distributed services. *Future Generation Computer Systems* 98 (2019), 342–361.
[121] Lin Ma, Dana Van Aken, Ahmed Hefny, Gustavo Mezerhane, Andrew Pavlo, and Geoffrey J Gordon. 2018. Query-based workload forecasting for self-driving database management systems. In *Proceedings of the 2018 International Conference on Management of Data*. 631–645.
[122] Lothar F Mackert and Guy M Lohman. 1986. R* optimizer validation and performance evaluation for local queries. In *Proceedings of the 1986 ACM SIGMOD international conference on Management of data*. 84–95.
[123] Divya Mahajan, Cody Blakeney, and Ziliang Zong. 2019. Improving the energy efficiency of relational and NoSQL databases via query optimizations. *Sustainable Computing: Informatics and Systems* 22 (2019), 120–133.
[124] Yaser Mansouri, Victor Prokhorenko, Faheem Ullah, and M Ali Babar. 2021. Evaluation of Distributed Databases in Hybrid Clouds and Edge Computing: Energy, Bandwidth, and Storage Consumption. *arXiv preprint arXiv:2109.07260* (2021).
[125] Xiaofeng Meng, Peiquan Jin, Wei Cao, and Lihua Yue. 2011. Report on the first international workshop on flash-based database systems (FlashDB 2011). *ACM Sigmod Record* 40, 2 (2011), 40–44.
[126] Justin Meza, Mehul A Shah, Parthasarathy Ranganathan, Mike Fitzner, and Judson Veazey. 2009. Tracking the power in an enterprise decision support system. In *Proceedings of the 2009 ACM/IEEE international symposium on Low power electronics and design*. 261–266.
[127] Adrian Michalke, Philipp M Grulich, Clemens Lutz, Steffen Zeuch, and Volker Markl. 2021. An Energy-Efficient Stream Join for the Internet of Things. In *Proceedings of the 17th International Workshop on Data Management on New Hardware (DaMoN 2021)*. 1–6.
[128] Sparsh Mittal. 2012. A survey of architectural techniques for DRAM power management. *International Journal of High Performance Systems Architecture* 4, 2 (2012), 110–119.
[129] Sparsh Mittal. 2014. A survey of techniques for improving energy efficiency in embedded computing systems. *International Journal of Computer Aided Engineering and Technology* 6, 4 (2014), 440–459.
[130] Mehdi Moghaddamfar, Christian Färber, Wolfgang Lehner, Norman May, and Akash Kumar. 2021. Resource-Efficient Database Query Processing on FPGAs. In *Proceedings of the 17th International Workshop on Data Management on New Hardware (DaMoN 2021)*. 1–8.
[131] Karsten Molka and Giuliano Casale. 2017. Energy-efficient resource allocation and provisioning for in-memory database clusters. In *2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*. IEEE, 19–27.
[132] Rene Mueller and Jens Teubner. 2010. FPGAs: a new point in the database design space. In *Proceedings of the 13th International Conference on Extending Database Technology*. 721–723.
[133] Tobias Mühlbauer, Wolf Rödiger, Robert Seilbeck, Angelika Reiser, Alfons Kemper, and Thomas Neumann. 2014. One DBMS for all: the Brawny Few and the Wimpy Crowd. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*. 697–700.
[134] Guofang Nan and Minqiang Li. 2010. Energy-efficient query management scheme for a wireless sensor database system. *EURASIP Journal on Wireless Communications and Networking* 2010 (2010), 1–18.
[135] Thomas Neumann. 2011. Efficiently compiling efficient query plans for modern hardware. *Proceedings of the VLDB Endowment* 4, 9 (2011), 539–550.
[136] Xuan-Thuan Nguyen, Trong-Thuc Hoang, Hong-Thu Nguyen, Katsumi Inoue, and Cong-Kha Pham. 2018. An FPGA-based hardware accelerator for energy-efficient bitmap index creation. *IEEE Access* 6 (2018), 16046–16059.
[137] Anne-Cecile Orgerie, Marcos Dias de Assuncao, and Laurent Lefevre. 2014. A survey on techniques for improving the energy efficiency of large-scale distributed systems. *ACM Computing Surveys (CSUR)* 46, 4 (2014), 1–31.
[138] Yi Ou, Theo Härder, and Peiquan Jin. 2009. CFDC: a flash-aware replacement policy for database buffer management. In *Proceedings of the fifth international workshop on data management on new hardware*. 15–20.
[139] Yi Ou, Theo Härder, and Daniel Schall. 2010. Performance and power evaluation of flash-aware buffer algorithms. In *International Conference on Database and Expert Systems Applications*. Springer, 183–197.
[140] Sangwon Park. 2013. Flash-Aware Cost Model for Embedded Database Query Optimizer. *J. Inf. Sci. Eng.* 29, 5 (2013), 947–967.
[141] Steven Pelley, Kristen LeFevre, and Thomas F Wenisch. 2011. Do query optimizers need to be SSD-aware?. In *ADMS@ VLDB*. Citeseer, 44–51.
[142] Jayaprakash Pisharath, Alok Choudhary, and Mahmut Kandemir. 2004. Energy management schemes for memory-resident database systems. In *Proceedings of the thirteenth ACM international conference on Information and knowledge management*. 218–227.
[143] Jayaprakash Pisharath, Alok Choudhary, and Mahmut Kandemir. 2004. Reducing energy consumption of queries in memory-resident database systems. In *Proceedings of the 2004 international conference on Compilers, architecture, and synthesis for embedded systems*. 35–45.
[144] Meikel Poess and Raghunath Othayoth Nambiar. 2008. Energy cost, the key challenge of today's data centers: a power consumption analysis of TPC-C results. *Proceedings of the VLDB Endowment* 1, 2 (2008), 1229–1240.

[145] Meikel Poess, Raghunath Othayoth Nambiar, Kushagra Vaid, John M Stephens Jr, Karl Huppler, and Evan Haines. 2010. Energy benchmarks: a detailed analysis. In *Proceedings of the 1st International Conference on Energy-Efficient Computing and Networking*. 131–140.
[146] Meikel Poess, Da Qi Ren, Tilmann Rabl, and Hans-Arno Jacobsen. 2018. Methods for Quantifying Energy Consumption in TPC-H. In *Proceedings of the 2018 ACM/SPEC International Conference on Performance Engineering*. 293–304.
[147] Iraklis Psaroudakis, Thomas Kissinger, Danica Porobic, Thomas Ilsche, Erietta Liarou, Pınar Tözün, Anastasia Ailamaki, and Wolfgang Lehner. 2014. Dynamic fine-grained scheduling for energy-efficient main-memory queries. In *Proceedings of the Tenth International Workshop on Data Management on New Hardware*. 1–7.
[148] Tilmann Rabl and Hans-Arno Jacobsen. 2017. Query centric partitioning and allocation for partially replicated database systems. In *Proceedings of the 2017 ACM International Conference on Management of Data*. 315–330.
[149] Suzanne Rivoire, Mehul A Shah, Parthasarathy Ranganathan, and Christos Kozyrakis. 2007. JouleSort: a balanced energy-efficiency benchmark. In *Proceedings of the 2007 ACM SIGMOD international conference on Management of data*. 365–376.
[150] Suzanne Rivoire, Mehul A Shah, Parthasarathy Ranganathan, Christos Kozyrakis, and Justin Meza. 2007. Models and metrics to enable energy-efficiency optimizations. *Computer* 40, 12 (2007), 39–48.
[151] Manuel Rodriguez-Martinez, Harold Valdivia, Jaime Seguel, and Melvin Greer. 2011. Estimating power/energy consumption in database servers. *Procedia Computer Science* 6 (2011), 112–117.
[152] Amine Roukh. 2015. Estimating power consumption of batch query workloads. In *Model and Data Engineering*. Springer, 198–212.
[153] Amine Roukh and Ladjel Bellatreche. 2015. Eco-processing of OLAP complex queries. In *International Conference on Big Data Analytics and Knowledge Discovery*. Springer, 229–242.
[154] Amine Roukh, Ladjel Bellatreche, Selma Bouarar, and Ahcene Boukorca. 2017. Eco-physic: Eco-physical design initiative for very large databases. *Information Systems* 68 (2017), 44–63.
[155] Amine Roukh, Ladjel Bellatreche, Ahcène Boukorca, and Selma Bouarar. 2015. Eco-dmw: Eco-design methodology for data warehouses. In *Proceedings of the ACM Eighteenth International Workshop on Data Warehousing and OLAP*. 1–10.
[156] Amine Roukh, Ladjel Bellatreche, and Carlos Ordonez. 2016. Enerquery: energy-aware query processing. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*. 2465–2468.
[157] Amine Roukh, Ladjel Bellatreche, Nikos Tziritas, and Carlos Ordonez. 2016. Energy-aware query processing on a parallel database cluster node. In *International Conference on Algorithms and Architectures for Parallel Processing*. Springer, 260–269.
[158] Behzad Salami, Gorker Alp Malazgirt, Oriol Arcas-Abella, Arda Yurdakul, and Nehir Sonmez. 2017. AxleDB: A novel programmable query processing platform on FPGA. *Microprocessors and Microsystems* 51 (2017), 142–164.
[159] Abdurrashid Ibrahim Sanka, Mehdi Hasan Chowdhury, and Ray CC Cheung. 2021. Efficient High-Performance FPGA-Redis Hybrid NoSQL Caching System for Blockchain Scalability. *Computer Communications* 169 (2021), 81–91.
[160] Parag Sarda and Jayant R Haritsa. 2004. Green query optimization: Taming query optimization overheads through plan recycling. In *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30*. 1333–1336.
[161] Daniel Schall and Theo Härder. 2013. Energy-proportional query execution using a cluster of wimpy nodes. In *Proceedings of the Ninth International Workshop on Data Management on New Hardware*. 1–6.
[162] Daniel Schall and Theo Härder. 2013. Towards an energy-proportional storage system using a cluster of wimpy nodes. *Datenbanksysteme für Business, Technologie und Web (BTW) 2032* (2013).
[163] Daniel Schall and Theo Härder. 2014. Approximating an energy-proportional dbms by a dynamic cluster of nodes. In *International Conference on Database Systems for Advanced Applications*. Springer, 297–311.
[164] Daniel Schall and Theo Härder. 2014. Energy and Performance-Can a Wimpy-Node Cluster Challenge a Brawny Server? *arXiv preprint arXiv:1407.0386* (2014).
[165] Daniel Schall and Theo Härder. 2015. Dynamic physiological partitioning on a shared-nothing database cluster. In *2015 IEEE 31st International Conference on Data Engineering*. IEEE, 1095–1106.
[166] Daniel Schall, Volker Höfner, and Manuel Kern. 2011. Towards an enhanced benchmark advocating energy-efficient systems. In *Technology Conference on Performance Evaluation and Benchmarking*. Springer, 31–45.
[167] Daniel Schall and Volker Hudlet. 2011. Wattdb: an energy-proportional cluster of wimpy nodes. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of data*. 1229–1232.
[168] Daniel Schall, Volker Hudlet, and Theo Härder. 2010. Enhancing energy efficiency of database applications using SSDs. In *Proceedings of the Third C* Conference on Computer Science and Software Engineering*. 1–9.
[169] Timos K Sellis. 1988. Multiple-query optimization. *ACM Transactions on Database Systems (TODS)* 13, 1 (1988), 23–52.
[170] Rathijit Sen and Alan Halverson. 2017. Frequency governors for cloud database OLTP workloads. In *2017 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*. IEEE, 1–6.
[171] Rathijit Sen and Karthik Ramachandra. 2018. Characterizing resource sensitivity of database workloads. In *2018 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. IEEE, 657–669.

[172] Vibhuti S Sengar and Jayant R Haritsa. 2003. PLASTIC: Reducing query optimization overheads through plan recycling. In *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*. 676–676.
[173] Craig Shallahamer, Jody Alkema, Tim Gorman, and Jared Still. 2007. *Forecasting Oracle Performance*. Springer.
[174] Junaid Shuja, Sajjad A Madani, Kashif Bilal, Khizar Hayat, Samee U Khan, and Shahzad Sarwar. 2012. Energy-efficient data centers. *Computing* 94, 12 (2012), 973–994.
[175] Utku Sirin, Raja Appuswamy, and Anastasia Ailamaki. 2016. OLTP on a server-grade ARM: power, throughput and latency comparison. In *Proceedings of the 12th International Workshop on Data Management on New Hardware*. 1–7.
[176] Evangelia A Sitaridi and Kenneth A Ross. 2016. GPU-accelerated string matching for database applications. *The VLDB Journal* 25, 5 (2016), 719–740.
[177] Michael Stonebraker. 2010. SQL databases v. NoSQL databases. *Commun. ACM* 53, 4 (2010), 10–11.
[178] Balaji Subramaniam and Wu-chun Feng. 2014. On the Energy Proportionality of Distributed NoSQL Data Stores. In *International Workshop on Performance Modeling, Benchmarking and Simulation of High Performance Computer Systems*. Springer, 264–274.
[179] Xuan Sun, Chun Jason Xue, Jinghuan Yu, Tei-Wei Kuo, and Xue Liu. 2021. Accelerating data filtering for database using FPGA. *Journal of Systems Architecture* 114 (2021), 101908.
[180] Alexander S Szalay, Gordon C Bell, H Howie Huang, Andreas Terzis, and Alainna White. 2010. Low-power amdahl-balanced blades for data intensive computing. *ACM SIGOPS Operating Systems Review* 44, 1 (2010), 71–75.
[181] Pınar Tözün, Ippokratis Pandis, Ryan Johnson, and Anastasia Ailamaki. 2013. Scalable and dynamically balanced shared-everything OLTP with physiological partitioning. *The VLDB Journal* 22, 2 (2013), 151–175.
[182] Dimitris Tsirogiannis, Stavros Harizopoulos, and Mehul A Shah. 2010. Analyzing the energy efficiency of a database server. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*. 231–242.
[183] Dimitris Tsirogiannis, Stavros Harizopoulos, Mehul A Shah, Janet L Wiener, and Goetz Graefe. 2009. Query processing techniques for solid state drives. In *Proceedings of the 2009 ACM SIGMOD International Conference on Management of data*. 59–72.
[184] Yi-Cheng Tu, Xiaorui Wang, Bo Zeng, and Zichen Xu. 2014. A system for energy-efficient data management. *ACM SIGMOD Record* 43, 1 (2014), 21–26.
[185] Annett Ungethüm, Dirk Habich, Tomas Karnagel, Wolfgang Lehner, Nils Asmussen, Marcus Völp, Benedikt Nöthen, and Gerhard Fettweis. 2015. Query processing on low-energy many-core processors. In *2015 31st IEEE International Conference on Data Engineering Workshops*. IEEE, 155–160.
[186] Annett Ungethüm, Thomas Kissinger, Dirk Habich, and Wolfgang Lehner. 2016. Work-energy profiles: General approach and in-memory database application. In *Technology Conference on Performance Evaluation and Benchmarking*. Springer, 142–158.
[187] Annett Ungethüm, Thomas Kissinger, Willi-Wolfram Mentzel, Dirk Habich, and Wolfgang Lehner. 2016. Energy elasticity on heterogeneous hardware using adaptive resource reconfiguration LIVE. In *Proceedings of the 2016 International Conference on Management of Data*. 2173–2176.
[188] Giorgio Luigi Valentini, Walter Lassonde, Samee Ullah Khan, Nasro Min-Allah, Sajjad A Madani, Juan Li, Limin Zhang, Lizhe Wang, Nasir Ghani, Joanna Kolodziej, et al. 2013. An overview of energy efficiency techniques in cluster computing systems. *Cluster Computing* 16, 1 (2013), 3–15.
[189] Narseo Vallina-Rodriguez and Jon Crowcroft. 2012. Energy management techniques in modern mobile handsets. *IEEE Communications Surveys & Tutorials* 15, 1 (2012), 179–198.
[190] Vijay Vasudevan, David Andersen, Michael Kaminsky, Lawrence Tan, Jason Franklin, and Iulian Moraru. 2010. Energy-efficient cluster computing with FAWN: Workloads and implications. In *Proceedings of the 1st International Conference on Energy-Efficient Computing and Networking*. 195–204.
[191] Jun Wang, Ling Feng, Wenwei Xue, and Zhanjiang Song. 2011. A survey on energy-efficient data management. *ACM SIGMOD Record* 40, 2 (2011), 17–23.
[192] Lizhe Wang, Gregor Von Laszewski, Jay Dayal, and Fugang Wang. 2010. Towards energy aware scheduling for precedence constrained parallel tasks in a cluster with DVFS. In *2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing*. IEEE, 368–377.
[193] Xiaorui Wang and Ming Chen. 2008. Cluster-level feedback power control for performance optimization. In *2008 IEEE 14th International Symposium on High Performance Computer Architecture*. IEEE, 101–110.
[194] Malcolm Ware, Karthick Rajamani, Michael Floyd, Bishop Brock, Juan C Rubio, Freeman Rawson, and John B Carter. 2010. Architecting for power management: The IBM® POWER7™ approach. In *HPCA-16 2010 The Sixteenth International Symposium on High-Performance Computer Architecture*. IEEE, 1–11.
[195] Molly Webb et al. 2008. Smart 2020: Enabling the low carbon economy in the information age. *The Climate Group. London* 1, 1 (2008), 1–1.
[196] Rafael Weingärtner, Gabriel Beims Bräscher, and Carlos Becker Westphall. 2015. Cloud resource management: A survey on forecasting and profiling models. *Journal of Network and Computer Applications* 47 (2015), 99–106.

[197] Lisa Wu, Andrea Lottarini, Timothy K Paine, Martha A Kim, and Kenneth A Ross. 2014. Q100: The architecture and design of a database processing unit. *ACM SIGARCH Computer Architecture News* 42, 1 (2014), 255–268.
[198] Lisa Wu, Andrea Lottarini, Timothy K Paine, Martha A Kim, and Kenneth A Ross. 2015. The Q100 database processing unit. *IEEE Micro* 35, 3 (2015), 34–46.
[199] Jiazhuang Xie. 2015. *Research on Energy-saving Query Processing Technologies for Database Clusters.* Master's thesis. University of Science and Technology of China.
[200] Baoping Xing. 2014. *Research on Energy Efficient Query Processing Technologies for Database Systems.* Master's thesis. University of Science and Technology of China.
[201] Zichen Xu. 2010. Building a power-aware database management system. In *Proceedings of the Fourth SIGMOD PhD Workshop on Innovative Database Research*. 1–6.
[202] Zichen Xu, Gele Bai, Ao Cui, and Shasha Wang. 2021. Power-aware throughput control for containerized relational operation. *CCF Transactions on High Performance Computing* 3, 1 (2021), 70–84.
[203] Zichen Xu, Yi-Cheng Tu, and Xiaorui Wang. 2010. Exploring power-performance tradeoffs in database systems. In *2010 IEEE 26th International Conference on Data Engineering (ICDE 2010)*. IEEE, 485–496.
[204] Zichen Xu, Yi-Cheng Tu, and Xiaorui Wang. 2012. PET: reducing database energy cost via query optimization. *Proceedings of the VLDB Endowment* 5, 12 (2012), 1954–1957.
[205] Zichen Xu, Yi-Cheng Tu, and Xiaorui Wang. 2013. Dynamic energy estimation of query plans in database systems. In *2013 IEEE 33rd International Conference on Distributed Computing Systems*. IEEE, 83–92.
[206] Zichen Xu, Yi-Cheng Tu, and Xiaorui Wang. 2015. Online energy estimation of relational operations in database systems. *IEEE transactions on computers* 64, 11 (2015), 3223–3236.
[207] Zichen Xu, Xiaorui Wang, and Yi-cheng Tu. 2013. Power-Aware Throughput Control for Database Management Systems. In *10th International Conference on Autonomic Computing (ICAC 13)*. 315–324.
[208] Puyuan Yang, Peiquan Jin, and Lihua Yue. 2014. Exploiting the Performance-Energy Tradeoffs for Mobile Database Applications. *J. Univers. Comput. Sci.* 20, 10 (2014), 1488–1498.
[209] Xindong You, Xueqiang Lv, Zhikai Zhao, Junmei Han, and Xueping Ren. 2020. A survey and taxonomy on energy-aware data management strategies in cloud environment. *IEEE Access* 8 (2020), 94279–94293.
[210] Ji-Tae Yun, Su-Kyung Yoon, Jeong-Geun Kim, and Shin-Dug Kim. 2020. Effective data prediction method for in-memory database applications. *The Journal of Supercomputing* 76, 1 (2020), 580–601.
[211] Hao Zhang, Gang Chen, Beng Chin Ooi, Kian-Lee Tan, and Meihui Zhang. 2015. In-memory big data management and processing: A survey. *IEEE Transactions on Knowledge and Data Engineering* 27, 7 (2015), 1920–1948.
[212] Kai Zhang, Feng Chen, Xiaoning Ding, Yin Huai, Rubao Lee, Tian Luo, Kaibo Wang, Yuan Yuan, and Xiaodong Zhang. 2015. Hetero-db: next generation high-performance database systems by best utilizing heterogeneous computing and storage resources. *Journal of Computer Science and Technology* 30, 4 (2015), 657–678.
[213] Kai Zhang, Kaibo Wang, Yuan Yuan, Lei Guo, Rubao Li, Xiaodong Zhang, Bingsheng He, Jiayu Hu, and Bei Hua. 2017. A distributed in-memory key-value store system on heterogeneous CPU–GPU cluster. *The VLDB Journal* 26, 5 (2017), 729–750.
[214] Jinhong Zhou, Shaoli Liu, Qi Guo, Xuda Zhou, Tian Zhi, Daofu Liu, Chao Wang, Xuehai Zhou, Yunji Chen, and Tianshi Chen. 2017. Tunao: A high-performance and energy-efficient reconfigurable accelerator for graph processing. In *2017 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*. IEEE, 731–734.
[215] Shijie Zhou, Charalampos Chelmis, and Viktor K Prasanna. 2016. High-throughput and energy-efficient graph processing on FPGA. In *2016 IEEE 24th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*. IEEE, 103–110.
[216] Yi Zhou, Shubbhi Taneja, Mohammed Alghamdi, and Xiao Qin. 2018. Improving energy efficiency of database clusters through prefetching and caching. In *2018 18th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*. IEEE, 388–391.
[217] Yi Zhou, Shubbhi Taneja, Xiao Qin, Wei-Shinn Ku, and Jifu Zhang. 2020. EDOM: Improving energy efficiency of database operations on multicore servers. *Future Generation Computer Systems* 105 (2020), 1002–1015.
[218] Yi Zhou, Shubbhi Taneja, Chaowei Zhang, and Xiao Qin. 2018. GreenDB: Energy-efficient prefetching and caching in database clusters. *IEEE Transactions on Parallel and Distributed Systems* 30, 5 (2018), 1091–1104.
[219] Yanhong Zhu. 2014. *Real-time Power Modeling and Control in the Power-aware DBMS.* Master's thesis. Zhejiang University of Technology.