Proactive Energy Management in Database Systems

Yuto Hayamizu*
The University of Tokyo
Meguro, Tokyo, Japan
haya@tkl.iis.u-tokyo.ac.jp

Masaru Kitsuregawa
The University of Tokyo
Meguro, Tokyo, Japan
kitsure@tkl.iis.u-tokyo.ac.jp

Kazuo Goda
The University of Tokyo
Meguro, Tokyo, Japan
kgoda@tkl.iis.u-tokyo.ac.jp

ABSTRACT

Reducing carbon footprint of datacenters is an unavoidable challenge for sustainable IT infrastructure. As the core software for data processing and management, energy efficiency in database systems has arisen as an important research subject. The tradeoff between power consumption and performance has been extensively studied across various workloads and query optimization techniques have been proposed to improve energy efficiency. Utilization of general-purpose power management methods such as dynamic frequency/voltage scaling and device hibernation in the context of database systems has also been studied.

However, many other components that make up database systems still remain largely untouched from the perspective of energy management. Energy-oriented behavioral changes in these components can unlock the potential for further energy management.

This paper explores the opportunity that database systems can proactively manage energy consumption during query execution. We particularly focused on the query execution and storage management layers, which largely determine the behavior of the system during query execution. We propose an initial design of the query execution model that proactively manages energy consumption coupled with the energy-aware storage layout. Our prototype using PostgreSQL demonstrated up to 29.6% energy reduction with 7.6% execution time overhead in an empirical evaluation using TPC-H benchmark. We also discuss the future directions of proactive energy management in database systems.

KEYWORDS

Database systems, proactive energy management, energy saving, query execution model, query optimization, storage management

ACM Reference Format:
Yuto Hayamizu, Masaru Kitsuregawa, and Kazuo Goda. 2024. Proactive Energy Management in Database Systems. In *Proceedings of 3rd Workshop on Sustainable Computer Systems (HotCarbon'24).* ACM, New York, NY, USA, 6 pages.

1 INTRODUCTION

Reducing carbon footprint of datacenters is an unavoidable challenge for sustainable IT infrastructure [2]. When it comes to improving the energy efficiency of datacenters, it is certainly necessary to enhance the energy efficiency of hardware components and facilities. However, re-architecting the software that actually drives these hardware components is also a crucial challenge.

As the core software for data processing and management, energy efficiency in database systems has arisen as an important research subject [5, 6]. The Claremont Report[1], which brought together leading researchers and practitioners in the database field, highlighted energy efficiency as one of the important research directions. Coinciding with the publication of the report, pioneering studies focusing on the energy efficiency of database systems were published [16, 18, 19, 22].

State-of-the-art studies on energy efficiency in database systems can be broadly categorized into the following three areas: (1) power-performance analysis of query operators s.t. full-table scans, index scans, hash joins, etc. on various types of workloads, (2) utilization of general-purpose power management techniques such as dynamic processor frequency and voltage scaling (DVFS) and idle device hibernation, (3) energy-aware cost modeling for query optimization. However, many other components that make up database systems remain largely untouched. In other words, these components can still be seen as **reactively** benefiting from the effects of the aforementioned energy management techniques.

We believe that database systems can take more **proactive** approaches to manage energy consumption. This is because database systems have a high degree of flexibility in determining the specific procedural steps to serve requested queries, which is usually described by declarative query languages such as SQL. Energy-oriented behavioral changes of database systems can unlock the potential for further energy management.

This paper explores the opportunity that database systems can proactively save energy consumption during query execution. In the case study, we particularly focus on the query execution and storage management layers, which largely determine the behavior of the system during query execution. We present the idea of incorporating power-state control commands of storage devices into the query execution model and demonstrates how to apply it to basic relational query operators. Furthermore, we discuss a tablespace layout that allows device power control commands issued by query operators to effectively reduce energy consumption.

To empirically evaluate the proposed model, we have prototyped the proposed query execution model using PostgreSQL, and built an experimental environment to measure the energy consumption of the database server. In experiments using the 100GB-scale TPC-H benchmark, our prototype consistently reduced the consumed energy compared to the original PostgreSQL for multiple queries, up to 29.6% energy reduction, anticipating further improvement with the sophistication of the proposed model in the follow-up studies.

The contributions of this paper are summarized as follows:

*He was affiliated with the University of Tokyo at the time of this research and is currently affiliated with Justice Technologies Inc.

HotCarbon'24, July 9, 2024, Santa Cruz, CA

- This paper introduces *proactive energy management* for database systems, an idea of creating opportunities for energy management by redesigning the behavior of the database engine.
- This paper presents an initial design of the query execution model that proactively manages power-state of storage devices and demonstrates the effectiveness of the proposed model through an empirical evaluation using PostgreSQL and TPC-H benchmark.
- This paper explores the future directions of proactive energy management in database systems.

The remainder of this paper is organized as follows. Section 2 summarizes the related work. Section 3 describes the concept of proactive energy management in database systems. Section 4 presents the case study of proactive energy management in the query execution model and experimental evaluation. Section 5 discusses the future directions of proactive energy management in database systems. Finally, Section 6 concludes the paper.

2 RELATED WORK

In the late 2000s, the energy efficiency of datacenters started to gain attention. The U.S. government agency reported that power consumption of datacenters doubled from 2000 to 2006[2], and the Claremont Report[1] highlighted energy efficiency as one of the important research directions in the database field.

As fundamental studies for energy efficiency, characteristics of power-performance relationship were analyzed in various aspects such as operator-level analysis[24], or workload level analysis such as transactional workloads[18], analytical workloads[12, 16, 19, 20, 22], etc.

Based on these analyses, Xu et al. pointed out the importance of energy-aware query optimization [25], and proposed a query optimization under the tradeoff annotated by database administrators [26]. In the follow-up study, they also provide dynamic correction of cost models based on control theory[27]. Luo et al. modeled energy costs of analytical queries with processor speed scaling[15].

Another approach of energy management in database systems is to utilize general-purpose power management techniques. DVFS has been studied in the context of database systems[8, 10, 11, 13, 23]. Hibernation of hardware components is also a common technique to save energy, and several approaches have been made such as memory module hibernation[9], storage device hibernation[17], cluster node hibernation[21], and MapReduce node hibernation[14].

3 PROACTIVE ENERGY MANAGEMENT

The first possibility of proactive energy management is to treat power control commands to hardware devices as primitive operations in query processing. Database systems inherently know the exact storage areas it is going to access and the computational resources that are going to be required for upcoming operations. Database systems have all the necessary information to plan the optimal timing for switching the power-state of hardware devices, which cannot be directly obtained from other peripheral programs or the operating system.

Thus, it is natural that database systems can take the initiative in managing the power-state of hardware devices. Treating power control commands as primitives on par with operations such as storage access and record processing appears to be an effective fundamental approach for applying the idea of proactive energy management.

The second possibility of proactive energy management is to fully leverage the design flexibility in query processing.

After the relational model[3] was invented, database systems typically receive requests in declarative query languages such as SQL. This allowed the decoupling of the logical requirements and the physical implementation of the query processing, and enables database systems to have a high degree of design flexibility in determining the specific procedural steps for processing queries.

Database systems typically modularize query processing as a *query execution model*, and many of today's database system implementations adopt a derivative of Volcano-style iterator model[4] as their foundation. In Volcano-style iterator model, common operations such as table scan, join, sort, and aggregation are formalized as *logical operators*, and implementations can provide multiple *physical operators* for each logical operator. The combination of logical operators describes the logical steps (tables to be scanned, join orderings, requirements of sorting, etc.), and each physical operator chosen for a logical operator describes the procedural steps (table access methods, join algorithms, sorting algorithms, etc.).

Query optimization is a process of selecting the optimal combination of logical operators and corresponding physical operators for each logical operator based on predefined cost functions of physical operators. Incorporating energy-related metrics into the cost function is a solid step forward to improve the energy efficiency of database systems. On the other hand, it is not possible to switch behaviors beyond the scope of the query execution model that serves as the basis for query optimization.

In the context of energy management, the iterator model encounters the issue of logical operators losing their logical nature. In the iterator model, physical operators for each logical operator are assumed to be interchangeable, and the physical operators are designed to be independent of each other. However, in the context of energy management, physical operators affect each other because of the statefulness of the power-state of hardware devices. Hardware devices necessitate a certain amount of time overhead for power-state transitions, and there may be additional energy costs for state transitions. This means that possible power-states may change depending on the sequence in which physical operators are executed and the combination of physical operators that are executed concurrently in a pipelined manner.

Considering these factors, to fully leverage the intrinsic design flexibility of database systems, it is time to rethink the query execution model. As an initial step toward proactive energy management, we will discuss an initial design of the query execution model that proactively manages power-state of storage devices.

4 CASE STUDY: QUERY EXECUTION MODEL

4.1 Extension of query execution model

In the conventional iterator model, a logical operator l serves as an iterator that produces a tuple per each call of GETNEXT interface, and typically corresponds to a relational operator: e.g. selecting tuples from table R (denoted as $\sigma(R)$), joining two tables R, S (denoted

as $R \bowtie S$). Each call of GETNEXT may return a tuple, or NULL to indicate the end of processing. A logical operator can take up to two inputs, which are also logical operators or tables. A logical operator may call GETNEXT interface of these inputs to obtain necessary tuples for processing. Input-output relationship of logical operators forms a binary-tree structure, which is called a *query execution plan*. Based on this structure, the database system can complete the processing of a query by repeatedly calling GETNEXT of the root logical operator until it returns NULL.

As discussed in Section 3, the iterator model has a limitation in that it does not consider the statefulness of hardware devices. So, we propose to introduce a concept of a *quasi-logical operator* to explicitly handle power-state control commands and statefulness of underlying devices in the query execution model.

A quasi-logical operator q is a sequence of unit operators u_k denoted as $q = \langle u_1, u_2, \cdots \rangle$, where a unit operator u_k may be a logical operator or a power-state control command. We define the semantics of GETNEXT interface for quasi-logical operators as follows:

- A quasi-logical operator q keeps track of the *current* unit operator. Initially the current unit operator is the first unit operator u_1.
- When GETNEXT interface of q is called, it calls GETNEXT interface of the current unit operator and returns its result.
- If the current unit operator returns NULL, q advances the current unit operator to the next unit operator and calls GETNEXT interface of the new current unit operator.
- If there is no more unit operator, q keeps returning NULL.

Next, we introduce power-state control commands $\uparrow d$ and $\downarrow d$ for wake-up and stand-by of a set of storage device d. Where d_R denotes a set of storage devices that store table R, we can define the following quasi-logical operator for example:

$$q = \langle \uparrow d_R, \sigma(R), \downarrow d_R \rangle$$

This is a simple alternative to the logical operator $\sigma(R)$ with energy management. It wakes up the storage devices that store table R before accessing the table, and puts them into standby state after accessing the table.

We move on to a bit more complex cases. Consider a join operation $R \bowtie S$. When hash join (HJ) is employed as the physical operator, the database system first builds a hash table from R, and then probes the hash table with tuples from S. So the operation on R and S are executed in a serial manner, and a quasi-logical operator can naturally describe these operations for example:

$$\langle \uparrow d_R, \textsc{Build}(\sigma(R)), \downarrow d_R, \uparrow d_S, \textsc{Probe}(\sigma(S)), \downarrow d_S \rangle$$

Apparently, a single logical-operator corresponds to a number of quasi-logical operators, and theoretical design space is more complex. Since this paper aims to initiate the discussion of proactive energy management in database systems, complete design space exploration is left for future work. In this paper, we focus on scan and join operations, storage device power-state control, and the basic pattern of quasi-logical operators $\langle \uparrow d_R, u \text{ on } R, \downarrow d_S \rangle$.

4.2 Storage layout consideration

With the introduction of quasi-logical operators, we can explicitly handle power-state control commands in the query execution model. However, the effectiveness of proactive energy management depends on the layout of storage devices.

$$\langle \uparrow d_R, \textsc{Build}(\sigma(R)), \downarrow d_R, \uparrow d_S, \textsc{Probe}(\sigma(S)), \downarrow d_R \rangle$$

Let us consider the case of hash join shown above. When R, S resides on the same storage device, $\downarrow d_R, \uparrow d_S$ cancel each other out, and the whole storage needs to be active during the join operation. On the other hand, when R, S resides on different storage devices, each device can be partially stand-by during the join operation and there should be a potential for energy reduction. Thus, independent storage allocation for each table, especially for large tables such as fact tables in analytical workloads, is one effective approach to proactive energy management.

Another possibility is the partitioning for proactive energy management. Suppose that R and S are partitioned into $R_k, S_k (k = 1, 2, \cdots)$ with the join key, and R_k, S_k resides on the storage d_k.

$$\langle \cdots, \uparrow d_k, \textsc{Build}(\sigma(R_k)), \textsc{Probe}(\sigma(S_k)), \downarrow d_k, \cdots \rangle$$

This quasi-logical operator enables energy reduction by activating only the necessary storage devices for the partition being processed.

The potential design space of storage layout for proactive energy management is also quite large and it is difficult to cover completely in this paper. Again, since this paper aims to initiate the discussion of proactive energy management, we focus on the following policy in prototyping and evaluation: allocate an independent storage region for each large table such as fact tables, and another region for small tables such as dimension tables.

4.3 Prototype implementation

To evaluate the effectiveness of the proposed query execution model, we have implemented a prototype using PostgreSQL, a widely used open source database system. The architecture of the query executor and the structure of the query execution plan in PostgreSQL adhere to the Volcano-style iterator model, and it is relatively easy to extend the query execution model by introducing quasi-logical operators. Since the counterpart of GETNEXT interface in PostgreSQL is the `ExecProcNode` function, and we have inserted codes to spin up and down storage devices [1] around `ExecProcNode` function calls based on the annotated mapping between the tablespaces and storage devices. Based on PostgreSQL 14, we have modified 1175 lines of code to implement the proposed query execution model for proactive energy management.

4.4 Experimental setup

We prepared the database server equipped with a Xeon E3-1240 v5 (3.5GHz, 4c/8t) processor, 16GB DRAM, and eight 4TB 7.2krpm SATA HDDs connected via the host bus adapter for database storage. Figure. 1 shows the connection diagram of the database server, disk drives, and the power measurement circuit. To measure the power consumption of the database server and the disk drives, we designed a custom power outlet box that allows us to probe the current and voltage on the AC 100V power line. We used a Yokogawa WT1800 power meter to measure the power consumption of two power lines in the custom power outlet box.

[1] `hdparm` command was invoked to spin up and down devices.

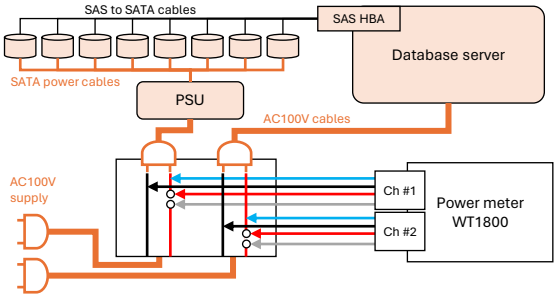


Figure 1: Connection diagram of the database server, disk drives and the power measurement circuit.

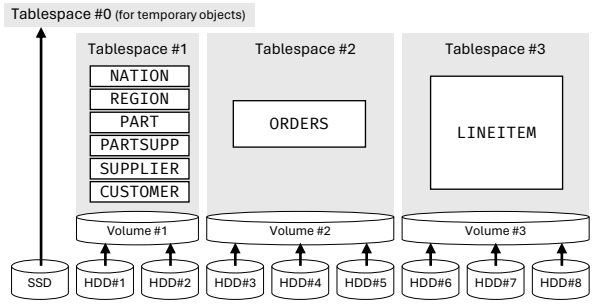


Figure 2: Storage organization and tablespace layout for our prototype. For normal PostgreSQL, all hard disk drives were organized as a single volume.

We prepared TPC-H benchmark dataset with ScaleFactor=100 (raw input data is approximately 100GB). When we conducted the measurement with normal PostgreSQL, all hard disk drives were organized as a single RAID0 volume and a single tablespace was created on it. On the other hand, when we conducted the measurement with our prototype, we created 3 RAID0 volumes and 3 tablespaces on the hard disk drives, placed dimension tables (NATION, REGION, PART, PARTSUPP, SUPPLIER, CUSTOMER) on tablespace #1, and placed fact tables (LINEITEM, ORDERS) on tablespace #2 and #3 respectively as shown in Figure. 2. Any temporary tables were created on tablespace #0 on system storage.

4.5 Experimental results

We have measured the query execution time and the power consumption of TPC-H queries from Q.1 to Q.10 except Q.2[2]. Since the query optimizer of PostgreSQL is not aware of our prototyped query execution model, we hinted the ordering of joins to the optimizer in SQL statements. We used the same pattern of the query execution plan for each query in both normal PostgreSQL and our prototype.

Figure. 3 shows the comparison of power consumption of the database server during the execution of TPC-H Q.7 between *Normal PG* (normal PostgreSQL) and *Proactive PG* (our prototype). For this query, both case employed the same pattern of query execution plan shown in the following steps: (1) $T_1 \leftarrow$ NATION $\bowtie$ CUSTOMER,

[2]Q.2 was omitted because our prototype did not support semi-join operators.

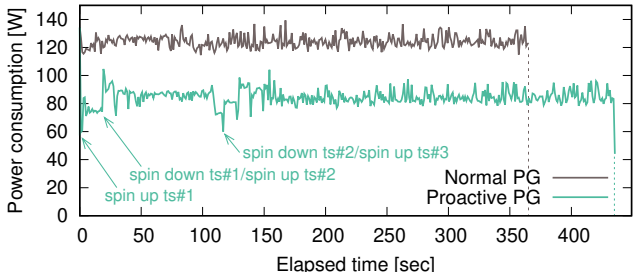


Figure 3: Power consumption of TPC-H Q.7: Normal PG (PostgreSQL) and Proactive PG (our prototype).

(2) $T_2 \leftarrow$ NATION $\bowtie$ SUPPLIER, (3) $T_3 \leftarrow T_1 \bowtie$ ORDERS, and (4) $T_4 \leftarrow T_3 \bowtie (T_2 \bowtie$ LINEITEM) as final output. Note that all joins were hash joins. In the case of Proactive PG, all disk drives were in stand-by state before query execution, and the power-state of disk drives were switched to active state only when the corresponding tablespace was accessed. Step (1) and (2) accessed only tablespace #1 (NATION, CUSTOMER, SUPPLIER. From $t = 0$ to 18), step (3) accessed only tablespace #2 (ORDERS. From $t = 18$ to 118), and step (4) accessed only tablespace #3 (LINEITEM. From $t = 118$ to the end).

As shown in Figure. 3, wait time of disk power-state transition was observed at the beginning of steps (1), (3), and (4) in Proactive PG and the query execution time was 68.5 seconds longer than that of Normal PG. On the other hand, the power consumption of Proactive PG was lower than that of Normal PG and reduced 39.6W in average because only one tablespace was active at a time. The enegy consumption of Normal PG and Proactive PG were 45.4kJ and 36.7kJ, respectively. This observation confirmed that Proactive PG reduced the average power consumption by 31.9% and the energy consumption by 19.1% despite 18.7% execution time overhead.

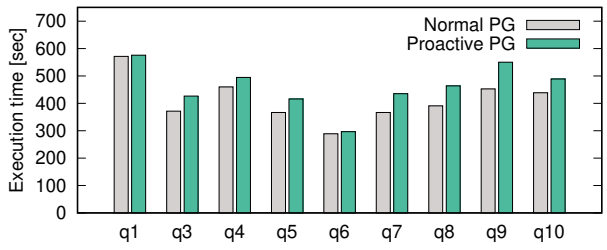
Figure 4 shows the comparison of query execution time and energy consumption between Normal PG and Proactive PG for all measured TPC-H queries. For all queries, a similar trend to Q.7 described earlier was observed. For each query, although there was an overhead in execution time due to power-state transitions, the energy consumption was consistently reduced, up to 29.6% reduction with 7.6% execution time overhead in Q.4.

In summary, our prototype based the idea of proactive energy management successfully reduced the energy consumption of the database server by up to 29.6% even with the execution time overhead in the empirical evaluation using TPC-H benchmark.

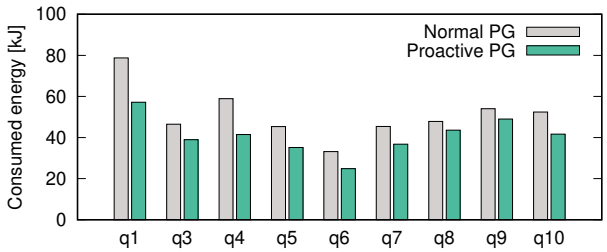
5 FUTURE DIRECTIONS

5.1 Query execution model, storage management and query optimization

We have introduced the concept of quasi-logical operators and storage layout to explicitly handle power-state control commands for proactive energy management. As described in Section 4, we have just defined the basic framework and provided several cases corresponding to basic relational operations. Compared to the conventional logical operators, quasi-logical operators involve potentially more complex design space and there exists a large room for



(a) Query execution time



(b) Energy consumption

Figure 4: Query execution time and energy consumption between Normal PG (PostgreSQL) and Proactive PG (out prototype) for TPC-H queries. *Although power-state control commands incurred execution time overhead, the energy consumption is consistently lower in Proactive PG, up to 29.6% reduction.*

further exploration including, but is not limited to, the following aspects: (1) coverage of common operations in database queries such as sorting, aggregation, subqueries, user-defined procedures, etc, (2) scheduling of unit operators in quasi-logical operators, (3) equivalent transformation of quasi-logical operators, (4) coupling with storage layout, and (5) quasi-logical operator-based query optimization.

5.2 Scheduling of multiple queries

In addition to intra-query scheduling of power-state control commands, we can further consider the inter-query scheduling of power-state control commands. Consolidating usage of specific resources such as storage devices among multiple queries is expected to have a great impact on energy efficiency. Approaches studied in work sharing among multiple queries[7] can be a good starting point for this direction. However, energy-oriented scheduling is not necessarily the best in terms of performance, as shown in our empirical evaluation, so there would be another opportunity of multiple query scheduling for proactive energy management.

5.3 Hardware heterogeneity

The heterogeneity of hardware components is one of the prominent trends in modern datacenters. Even in a single processor, memory access is not uniform anymore in the era of NUMA architecture, and multicore processors with different types of cores are becoming more popular. Purpose-specific accelerators such as GPUs and FPGAs are also becoming quite popular in datacenters. Proactive energy management in database systems can be a good fit for these heterogeneous hardware components, and there are many opportunities to explore in this direction.

5.4 Cluster and cloud orchestration

In production environments, it is rare that a single server serves all of the tasks of database systems. Database servers are often deployed in a cluster, and sometimes multiple clusters cooperate to satisfy the application requirements. In addition, modern IT infrastructure usually deploys a number of application servers and cache servers around the database servers, sometimes in virtualized cloud environments. For proactive energy management, power control of these servers and clusters can be also an another unit of primitive operations.

5.5 Facility integration

The energy consumption of datacenter facility is the largest and the most challenging opportunity for proactive energy management. Cooling and power distribution facilities are the most energy-consuming components in datacenters. For most software, these facilities are not considered as targets of any form of control in the first place. However, database systems that subordinate numerous servers and storage resources have the potential to cooperate with these facilities. In large scale deployment of database systems such that spans multiple racks in datacenters, scheduling of operations could create changes in power consumption at a level that affects the operation modes of these facilities.

6 CONCLUSION

This paper introduced the idea of proactive energy management in database systems, which creates opportunities for energy management by redesigning the behavior of the database system itself. As an initial step toward proactive energy management, we presented a design of the query execution model that proactively manages power-state of storage devices and demonstrated the effectiveness of the proposed model through an empirical evaluation using PostgreSQL and TPC-H benchmark. Experimental evaluation with our prototype demonstrated that the proposed query execution model consistently reduces the energy consumption for tested TPC-H queries, up to 29.6% reduction with 7.6% execution time overhead. Although the proposed model is a simple and naïve design, the results showed its effectiveness in reducing energy consumption. Since there exist many opportunities for the sophistication of the proposed model, this evaluation result indicated that proactive energy management can be a promising approach to improving energy management in database systems. We also envisioned the future possibilities of proactive energy management in hopes of stimulating research communities.

ACKNOWLEDGMENTS

This work has been partially supported by Cross-cutting Technology Development for IoT Promotion program of NEDO and Big Data Value Co-creation Platform Engineering social cooperation program at UTokyo-IIS with Hitachi.

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, Johannes Gehrke, Le Gruenwald, Laura M. Haas, Alon Y. Halevy, Joseph M. Hellerstein, Yannis E. Ioannidis, Hank F. Korth, Donald Kossmann, Samuel Madden, Roger Magoulas, Beng Chin Ooi, Tim O'Reilly, Raghu Ramakrishnan, Sunita Sarawagi, Michael Stonebraker, Alexander S. Szalay, and Gerhard Weikum. 2008. The Claremont report on database research. *SIGMOD Rec.* 37, 3 (sep 2008), 9–19. https://doi.org/10.1145/1462571.1462573

[2] Richard Brown, Eric Masanet, Bruce Nordman, Bill Tschudi, Arman Shehabi, John Stanley, Jonathan Koomey, Dale Sartor, Peter Chan, Joe Loper, Steve Capana, and Bruce Hedman. 2007. Report to Congress on Server and Data Center Energy Efficiency: Public Law 109-431. (01 2007). https://doi.org/10.2172/929723

[3] E. F. Codd. 1970. A relational model of data for large shared data banks. *Commun. ACM* 13, 6 (jun 1970), 377–387. https://doi.org/10.1145/362384.362685

[4] G. Graefe. 1994. Volcano: An Extensible and Parallel Query Evaluation System. *IEEE Trans. on Knowl. and Data Eng.* 6, 1 (feb 1994), 120–135. https://doi.org/10.1109/69.273032

[5] 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* (Nantes, France) *(SETMDM '08)*. Association for Computing Machinery, New York, NY, USA, 24–28. https://doi.org/10.1145/1385486.1385494

[6] Stavros Harizopoulos, Mehul A. Shah, Justin Meza, and Parthasarathy Ranganathan. 2009. Energy Efficiency: The New Holy Grail of Data Management Systems Research. In *Fourth Biennial Conference on Innovative Data Systems Research, CIDR 2009*. Asilomar, CA, USA. http://www-db.cs.wisc.edu/cidr/cidr2009/Paper_112.pdf

[7] Stavros Harizopoulos, Vladislav Shkapenyuk, and Anastassia Ailamaki. 2005. QPipe: a simultaneously pipelined relational query engine. In *Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data* (Baltimore, Maryland) *(SIGMOD '05)*. Association for Computing Machinery, New York, NY, USA, 383–394. https://doi.org/10.1145/1066157.1066201

[8] Yuto Hayamizu, Kazuo Goda, Miyuki Nakano, and Masaru Kitsuregawa. 2011. Application-Aware Power Saving for Online Transaction Processing Using Dynamic Voltage and Frequency Scaling in a Multicore Environment. In *Architecture of Computing Systems - ARCS 2011*, Mladen Berekovic, William Fornaciari, Uwe Brinkschulte, and Cristina Silvano (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 50–61.

[9] Alexey Karyakin and Kenneth Salem. 2019. DimmStore: memory power optimization for database systems. *Proc. VLDB Endow.* 12, 11 (jul 2019), 1499–1512. https://doi.org/10.14778/3342263.33422629

[10] 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* (Houston, TX, USA) *(SIGMOD '18)*. Association for Computing Machinery, New York, NY, USA, 1717–1720. https://doi.org/10.1145/3183713.3193554

[11] Mustafa Korkmaz, Alexey Karyakin, Martin Karsten, and Kenneth Salem. 2015. Towards Dynamic Green-Sizing for Database Servers.. In *ADMS@ VLDB*. 25–36.

[12] Willis Lang, Stavros Harizopoulos, Jignesh M. Patel, Mehul A. Shah, and Dimitris Tsirogiannis. 2012. Towards energy-efficient database cluster design. *Proc. VLDB Endow.* 5, 11 (jul 2012), 1684–1695. https://doi.org/10.14778/2350229.2350280

[13] Willis Lang and Jignesh M. Patel. 2009. Towards Eco-friendly Database Management Systems. In *Fourth Biennial Conference on Innovative Data Systems Research, CIDR 2009, Asilomar, CA, USA, January 4-7, 2009, Online Proceedings*. www.cidrdb.org. http://www-db.cs.wisc.edu/cidr/cidr2009/Paper_51.pdf

[14] Willis Lang and Jignesh M. Patel. 2010. Energy management for MapReduce clusters. *Proc. VLDB Endow.* 3, 1–2 (sep 2010), 129–139. https://doi.org/10.14778/1920841.1920862

[15] Boming Luo, Yuto Hayamizu, Kazuo Goda, and Masaru Kitsuregawa. 2018. Modeling Query Energy Costs in Analytical Database Systems with Processor Speed Scaling. In *Database and Expert Systems Applications: 29th International Conference, DEXA 2018, Regensburg, Germany, September 3–6, 2018, Proceedings, Part II* (Regensburg, Germany). Springer-Verlag, Berlin, Heidelberg, 310–317. https://doi.org/10.1007/978-3-319-98812-2_27

[16] 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* (San Fancisco, CA, USA) *(ISLPED '09)*. Association for Computing Machinery, New York, NY, USA, 261–266. https://doi.org/10.1145/1594233.1594295

[17] Norifumi Nishikawa, Miyuki Nakano, and Masaru Kitsuregawa. 2012. Energy Efficient Storage Management Cooperated with Large Data Intensive Applications. In *2012 IEEE 28th International Conference on Data Engineering*. 126–137. https://doi.org/10.1109/ICDE.2012.47

[18] 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. *Proc. VLDB Endow.* 1, 2 (aug 2008), 1229–1240. https://doi.org/10.14778/1454159.1454162

[19] Meikel Poess and Raghunath Othayoth Nambiar. 2010. Tuning servers, storage and database for energy efficient data warehouses. In *2010 IEEE 26th International Conference on Data Engineering (ICDE 2010)*. 1006–1017. https://doi.org/10.1109/ICDE.2010.5447806

[20] Meikel Poess and Raghunath Othayoth Nambiar. 2010. A power consumption analysis of decision support systems. In *Proceedings of the First Joint WOSP/SIPEW International Conference on Performance Engineering* (San Jose, California, USA) *(WOSP/SIPEW '10)*. Association for Computing Machinery, New York, NY, USA, 147–152. https://doi.org/10.1145/1712605.1712629

[21] 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* (Athens, Greece) *(SIGMOD '11)*. Association for Computing Machinery, New York, NY, USA, 1229–1232. https://doi.org/10.1145/1989323.1989461

[22] 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* (Indianapolis, Indiana, USA) *(SIGMOD '10)*. Association for Computing Machinery, New York, NY, USA, 231–242. https://doi.org/10.1145/1807167.1807194

[23] Yi-Cheng Tu, Xiaorui Wang, Bo Zeng, and Zichen Xu. 2014. A system for energy-efficient data management. *SIGMOD Rec.* 43, 1 (may 2014), 21–26. https://doi.org/10.1145/2627692.2627696

[24] 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*. 155–160. https://doi.org/10.1109/ICDEW.2015.7129569

[25] 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)*. 485–496. https://doi.org/10.1109/ICDE.2010.5447840

[26] Zichen Xu, Yi-Cheng Tu, and Xiaorui Wang. 2012. PET: reducing database energy cost via query optimization. *Proc. VLDB Endow.* 5, 12 (aug 2012), 1954–1957. https://doi.org/10.14778/2367502.2367546

[27] 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*. 83–92. https://doi.org/10.1109/ICDCS.2013.21