Exploring Power-Performance Tradeoffs in Database Systems

Zichen Xu, [1] Yi-Cheng Tu, [1] and Xiaorui Wang [2]

[1]*Department of Computer Science & Engineering, University of South Florida*
4202 E. Fowler Ave., ENB118, Tampa, Florida, U.S.A.
{zxu5, ytu}@cse.usf.edu

[2]*Dept. of Electrical Engineering & Computer Science, University of Tennessee*
421 Ferris Hall, Knoxville, Tennessee, U.S.A.
xwang@eecs.utk.edu

***Abstract*— With the total energy consumption of computing systems increasing in a steep rate, much attention has been paid to the design of energy-efficient computing systems and applications. So far, database system design has focused on improving performance of query processing. The objective of this study is to experimentally explore the potential of power conservation in relational database management systems. We hypothesize that, by modifying the query optimizer in a DBMS to take the power cost of query plans into consideration, we will be able to reduce the power usage of database servers and control the tradeoffs between power consumption and system performance. We also identify the sources of such savings by investigating the resource consumption features during query processing in DBMSs. To that end, we provide an in-depth anatomy and qualitatively analyze the power profile of typical queries in the TPC benchmarks. We perform extensive experiments on a physical testbed based on the PostgreSQL system using workloads generated from the TPC benchmarks. Our hypothesis is supported by such experimental results: power savings in the range of 11% - 22% can be achieved by equipping the DBMS with a query optimizer that selects query plans based on both estimated processing time and power requirements.**

I. Introduction

The main objective of computing system design has been improving performance. With the significant increase in energy consumption of computers, power management has become a critical issue in system design and implementation [26]. Various sources [20], [18], [19] suggest an annual electricity bill of 2.7-14 billion US dollars for powering the servers in the United States. In a typical data center, electricity consumed by servers and cooling systems (needed to remove the heat generated by servers) contributes to around 20% of the total ownership cost, equivalent to one-third of the total maintenance cost [11]. This makes energy the second largest item in a typical IT department's monthly bill (while labour being the number one). It is generally believed that these numbers will continue to increase in the next few years [20], [18], [19], [11]. From an environmental point of view, computing systems contributes to 2% of the world's carbon footprint [24]. Data centers alone contributes to 0.5% (1.2% in U.S. [20]), with a projected four-fold increase by 2020.

While the above costs are calculated directly from energy consumption, power (i.e., energy consumption per unit time) savings are of more practical importance than energy savings in system design. Power capping has become a serious challenge for data centers in recent years. Controlling power consumption is an essential way to avoid system failures caused by power capacity overload or overheating due to increasing high server density (e.g., blade servers) [17], [34]. Modern high-density servers face an increasing probability of thermal failures, due to their continuously decreasing size and increasing demand for computational capabilities. For example, recent studies show that 50% of all electronics failures are related to overheating [37]. In addition, a 15°C increase in temperature could double the failure rate of a disk drive [4]. Therefore, it is important to reduce the database power consumption so that the total power consumption of an entire system can be kept below a given power budget. Furthermore, reduced power consumption can lead to lower system temperature and thus cooling costs [28]. This can contribute to significant energy savings in the long run as the cooling system could account for up to 50% of total energy consumption of a data center [33], [20].

Due to the above facts, power conservation in computing systems has attracted much attention from government agencies, industrial sectors, and the research communities. Standards and benchmarks are quickly adapting to the consideration of power consumption [27], [1], [31]. In the research part, early innovations on power-aware computer servers have concentrated on hardware and system software such as compilers and operating systems [22], [38], [23], [14], [7]. Those build a foundation for a computing environment where: 1) hardware can operate on various power-saving modes in which different power/performance tradeoffs can be achieved; 2) system software can control the operating modes of hardware and reschedule resource requests from applications to save energy. One common theme of system-level research on power-aware computing is that they treat the high-level applications as passive resource consumers (abstracted as workload) that are not aware of the low-level efforts of power reduction. In the past few years, the research community has shifted much interest to *power-aware applications* [6], [36], [10], which provide significant synergistic values to current research on the hardware and system levels. Our research falls into this

category.

In this paper, we study power consumption patterns and identify power-saving opportunities in database management systems (DBMS). Our ultimate goal is to build power-aware DBMSs that can achieve significant cost savings while maintaining reasonable performance in query processing. Such vision is motivated by the following observations.

First, power reduction in DBMSs are of high economical significance: DBMS is an important type of software in the three-tier computing architecture adopted by most of today's business computing environments. In a typical data center, a majority of the computing resources are dedicated to database servers, making DBMS the largest consumers of power in all the software applications deployed. It is generally true that the processing capacity of the back-end database servers determines that of the front-end web and application servers. In [29], a back-end to front-end power consumption ratio of $11 : 9.9$ is reported. Most database servers are configured with capacity sufficient to handle peak load. This implies great opportunities for power savings.

Second, a DBMS possesses salient features that can facilitate effective power control, making it a desirable platform for validating research ideas in general application-level power conservation. Specifically, a DBMS: 1) maintains extensive statistics about database states for the purpose of efficient query processing. These information can be directly used to derive the power profile of the database workload and help power-related decision making; 2) can adjust its behavior (e.g., selecting query execution plan, materializing views, building indexes) towards a performance goal. Its query optimizer can generate a large number of execution plans for the same query, allowing various tradeoffs between performance and power cost; and 3) is much like an OS itself by requesting large chunks of resources (e.g., memory buffer and disk space) from the OS and managing the distribution of such resources among its users. This provides opportunities to optimize resource management towards high power-efficiency inside the DBMS. It is also for the above reasons that *power-aware DBMS can be complementary to OS-level power management solutions and yield greater power savings*.

The main objective of this study is to validate our vision on power-aware DBMSs (PDBMS) and provide solutions to some of the main problems related to building such systems. Specifically, we want to answer the following questions:

1) *Are there really opportunities to save power with PDBMS? In other words, are existing DBMSs inefficient in terms of power consumption?*
2) *If so, what are the reasons for such inefficiency?*
3) *What can we do in PDBMSs to remedy the above problems?* and
4) *How much power do we expect to save in a PDBMS?*

This paper reports empirical results to address the above issues. Specifically, we profile the power consumption patterns of individual queries in popular TPC benchmarks by both modeling-based and experimental approaches. From these profiles, we identify significant potential for saving power and performing power-performance tradeoffs. We also propose a strategy to make DBMSs power-efficient: **redesign the query optimizer to take the power cost of query plans into consideration**. Via extensive experiments under different workloads, we show that a DBMS equipped with such a query optimizer can achieve up to 22% reduction of active power. Note that the main goal of this study is to validate this strategy by an initial design of the optimizer and we leave more sophisticated PDBMS design and implementation as future work.

Paper organization. The paper is organized as follows: we summarize related work in Section II; we state our hypothesis and motivating examples in Section III; Section IV describes the methodology and environment for our experiments; we present and interpret the experimental results in Section V and conclude this paper with discussions on future research plans in Section VI.

II. Related Work

System-level power-related research has created an ocean of literature. As mentioned earlier, much of the prior work has either attempted to reduce power consumption by improving the power-efficiency of individual hardware components [22], or focused on system-level power and thermal management [38], [23], [14], [7], [15], [30], [35]. The common philosophy behind these is: *power consumption should be proportional to the load put on the system*. In order to save energy, we need to turn the hardware to power-saving modes when the workload on the system is light. Among these work, Zeng et al. [39] manages power in operating system which provides us with an initial power model for power cost estimation of query plans. In this paper, we will show that, given the same load of user queries, the DBMS can choose to handle the load in a power-saving manner.

Application-level power-aware computing has become an active research field for the following reasons: 1) Low-level power optimization depends heavily on accurate estimation of the power profiles of applications. However, much information and statistics needed for such estimations are only available inside the applications. Workload abstraction is often found to be an oversimplified description of application behaviors; 2) Many applications have different execution paths to accomplish the same computational task. Power-aware applications, which adaptively adjust its behaviors according to the power-related states of underlying systems, can provide additional opportunities for power saving. Many research projects in this area are on web services [6], [16], [40], [32], [9], [8], [12]. Recent target application include MapReduce [10]. In [36], a general-purpose programming framework was developed to allow applications to be executed with different plans according to their power costs.

The power consumption in databases has just started drawing attention from the research community. The Claremont report on database research [2] explicitly states the importance of "designing power-aware DBMSs that limit energy costs

without sacrificing scalability ...". Two recent projects address power consumption issues in data processing applications and data centers: [31] presents a benchmark for evaluating the energy efficiency of implementations of various sorting algorithms. Due to the large volume of data, such algorithms are I/O intensive and therefore the I/O access patterns have profound effects on the energy efficiency of the algorithms. [29] reports extensive experimental results on the power consumption patterns in typical commercial database servers. Based on these results, it provides suggestions on how to make the system more power efficient. However, these suggestions focus on utilizing new hardware rather than modifying the DBMS software. To the best of our knowledge, our project is the first one that takes the path of redesigning the DBMS kernel for power-saving purposes.

While limiting power consumption is obviously necessary in mobile computing systems (e.g., PDAs, laptops) in which battery power is the most stringent resource, there are very few efforts [3] devoted to power-aware database systems on servers connected to power grids.

TABLE I

PEAK POWER CONSUMPTION OF VARIOUS AMD CPUS. DATA SOURCE: HTTP://WWW.AMD.COM .

Processor	TDP(W)
Athlon 64 3500+	67
Athlon 64 3700+	89
Athlon 64 FX-55/57	104
Opteron 2.8GHz	93
Opteron Dual Core 2.2GHz S	93
Opteron Dual Core 1MB Cache 64 2.4GHz	95
Opteron Dual Core 1MB Cache 64 3GHz	98

TABLE II

POWER CONSUMPTION OF VARIOUS SEAGATE MOMENTUS HARD DISKS. DATA SOURCE: HTTP://WWW.SEAGATE.COM .

Disk	Peak Power(W)	Idle Power(W)
5400.5, 320GB, SATA	2.45	0.92
7200.5, 320GB, SATA	3.03	0.95
7200.5, 100GB, ATA	3.03	0.95
7200.5, 160GB, SATA	3.83	1.10
7200.5, 100GB, SATA	4.03	1.74

III. HYPOTHESIS AND MOTIVATING EXAMPLES

In response to the questions list in Section I (especially questions 1 and 3), our main hypothesis in this study is:

There exist power saving opportunities in current DBMSs, and we can grab those opportunities by designing a query optimizer that takes power cost into consideration in query evaluation.

The above hypothesis is intuitive due to the well-known fact that the current query optimization mechanisms consider performance as the sole goal. Therefore, the lack of power considerations makes them unsuitable for saving power. On the other hand, it could also be counterintuitive to many since it is possible that query optimization towards shortest processing time coincides with that towards lowest power cost, and therefore no power savings can be achieved by modifying the behavior of the database engine. In fact, this has been a frequent argument we encounter in our communications with fellow researchers. The argument is based on the understanding of query processing time being a measure of some "load" to be finished by the DBMS, and more "load" the system faces, more energy will be consumed. Most of the power-aware computing research follow this line of reasoning. However, more careful analysis of the different types of resources associated with such "load" to be handled in the DBMS will lead to a different conclusion.

A. Rationale behind the hypothesis

In traditional performance-driven query optimizers, the plan cost is the summation of the holding times of all system resources including CPU, disks, and communication channels (in case of distributed databases). However, the increase of the processing speed of modern CPUs outpaces that of the I/O speed by orders of magnitude, making CPU time negligible as compared to I/O time in most cases. Therefore, query optimization becomes essentially a problem of finding the plan with the smallest number of I/O operations instead of "total" processing time.[1] In the paradigm of energy-aware computing, however, the weight of CPUs in terms of power consumption is greater than (or at least comparable to) that of the storage systems [29]. For example, according to Tables I and II (in Table I, TDP stands for thermal design power), power consumption of one Opteron Dual Core 3GHz CPU equals that of 24 SATA disks with 100GB volume each. Given this, we can easily see that the existing query optimization mechanisms in DBMSs are ill-suited for saving power. Consider the following scenario: for two execution plans A and B of the same query, if the I/O cost of plan A is slightly higher than that of plan B but it uses much less CPU time than B, the current query optimizer will choose B since I/O is always the dominant factor in total processing time. However, a query optimizer that considers both power and performance could choose A instead. This is because plan B's high CPU requirement translates into a (much) higher power consumption while it is only marginally better than A in (estimated) processing time. In this study, we explored the power and performance costs of a large number of plans for the TPC-H queries and found many cases that match the above scenario. One representative case will be demonstrated in Section III-B.

To explore the search plans generated by PostgreSQL, we carefully modify its plan generation process such that we can see all possible plans of a query. In the regular PostgreSQL system, a heuristic search algorithm that drops intermediate plans greedily is used, in order to achieve polynomial time traverse of the space that grows exponentially with the number of tables involved. However, it also keeps a trigger to allow

[1] In the case of distributed databases, communication bandwidth plays such a dominant role (or at least it is as important as the I/O time). However, we focus on a single server setup in this paper and leave the study on distributed databases as future work.

```
SELECT s_acctbal, s_name, n_name, p_partykey,
       p_mfgr, s_address, s_phone, s_comment
FROM
       part, supplier, partsupp,
       nation, region
WHERE
       p_partkey = ps_partkey
       AND s_suppkey = ps_suppkey
       AND p_size = $15$
       AND p_type like '\%BRASS\%'
       AND s_nationkey = n_nationkey
       AND n_regionkey = r_regionkey
       AND r_name = 'EUROPE'
ORDER BY
       s_acctbal desc, n_name,
       s_name, p_partkey
LIMIT 100;
```

Fig. 1. A sample query from TPC-H.

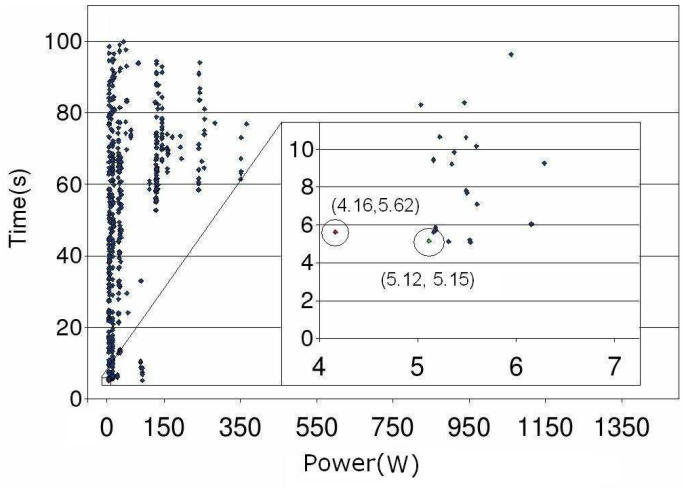

Fig. 2. The estimated processing time and power cost of plans evaluated by the modified PostgreSQL query optimizer for the query shown in Fig. 1. The small graph is a magnification of the bottom left portion of the large graph. Even the big graph contains only $1/5$ of the plans visited by the query optimizer. The other $4/5$ is ignored due to their inferiority (in both performance and energy).

exhaustive search, and we pull the trigger to collect data for all the possible plans. Tens of thousands of plans could be generated in the backend PostgreSQL server when a query is in preprocessing stage, along with their estimated time cost and power cost. The latter is rendered by our power model that will be discussed in Section IV-A.

B. A motivating example

Here we show one example with a query extracted from the TPC-H benchmark.[2] As shown in Fig. 1, this is a query that retrieves eight attributes with seven conjunctive conditions upon joining five tables. There are about 30000 plans generated before the final execution plan is selected. Most of the 30000 plans are inferior outliners that most likely would have been dropped in the heuristic search process (if the trigger was not pulled). The estimated costs of some of the more interesting plan nodes are presented in Fig. 2. In this graph, if a plan A dominates (i.e., bears lower energy and time costs) another plan B, we can be sure that B should not be considered. However, any plan in the non-dominant frontier of the graph can be chosen. Which one to choose will depend on the level of energy-performance tradeoff (i.e., how much performance to sacrifice to save 1% power) we are willing to make (more details can be found in Section IV-B).

From Fig. 2, we can see there are at least two plan nodes on the non-dominant frontier, which are shown as circled dots in the magnified portion of the graph. The green node (on the right) has a power cost of 5.12 and processing time of 5.15 while the red node to its left has energy and time costs of 4.16 and 5.62, respectively. If we use the original query optimizer, the green one would have been picked because of its small estimated time cost. However, if we choose the red node, although the time may increase to 5.62, the energy cost decreases to 4.16 – that is to trade a 5% performance degradation for a 18.75% power saving. This saving is substantial if the estimation comes with reasonable precision. Apparently, such tradeoffs can be utilized to achieve power reduction. Note here that the larger search space for one query is, the more likely such tradeoffs can be found. On the other hand, single table query scheduling has limited power-saving potential in our current method.

In summary, we believe the descriptions in Section III-A and the above motivating example provides an answer to question 2 listed in Section I. We will also verify our answer experimentally in Section V-B.

IV. Methodology

To capture the power-saving opportunities in query processing, our main approach is to find query plans with low power cost during query optimization. For this purpose, we modify the current query optimizer to make it incorporate a *power model* to estimate the power cost of plans, and a *query evaluation model* that takes both power and performance into account for comparing query plans. We use the open-source PostgreSQL DBMS as our experimental system. We feed the PostgreSQL system with the modified query optimizer with various workloads generated from the TPC-H[3] and TPC-C[4] benchmarks. In this project, we collect two types of experimental data:

1. *query processing performance*, which is obtained by recording the starting and ending time of the workload processing period within the DBMS (i.e., ignoring the communication delays); and
2. *power consumption*, which is measured by power meters connected to the server via a USB connection. Thus, the power we record is that of the whole server instead of individual hardware components. Although power is

[2]Specifically, it is query No. 2 in the original TPC-H benchmark tool.

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

[4]http://www.tpc.org/tpcc

TABLE III

POWER COSTS OF BASIC DATABASE OPERATIONS IN THE EXPERIMENTAL SYSTEM.

Function Parameter	Symbol	Default Value	Meaning
DEFAULT_CPU_TUPLE_POWER_COST	α	0.4	Estimated CPU power cost per tuple
DEFAULT_CPU_INDEX_TUPLE_POWER_COST	τ	0.05	Estimated CPU power cost per indexed tuple
DEFAULT_PAGE_POWER_COST	β	4.7	Estimated power cost for W/R one page without buffering

the focal point of this paper, we also discuss energy consumption in several occasions. Note that energy is the product of power consumption and query processing time.

The architecture of our experimental platform is shown in Fig. 3. For the ease of reproducing our results, we provide more details of the components in this platform as follows.

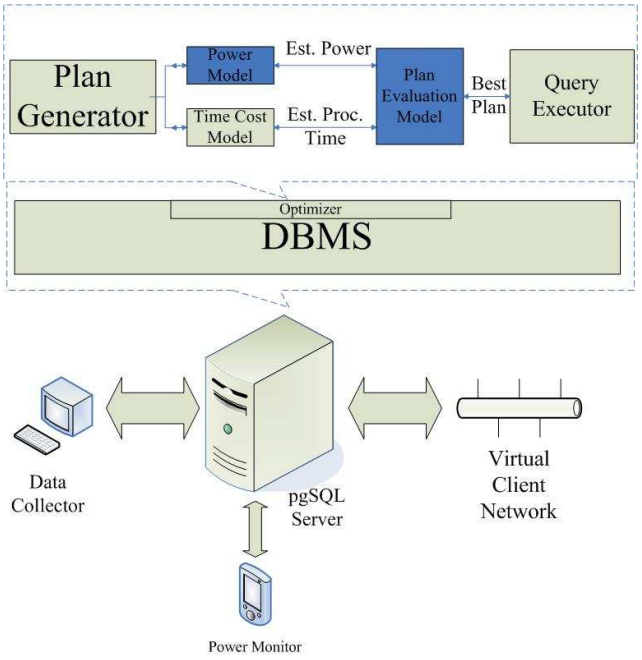

Fig. 3. Organization of the experimental platform.

A. Power Model

The cost model to estimate the plans' processing time in PostgreSQL works as follows: it first maps each plan to the number of basic operations (e.g., tuples to process) needed to compute the result following this plan. We call such numbers the *operation vector*. The plan's time cost is calculated from a set of static parameters that describe the resource holding time per basic operation (e.g., CPU time to process one tuple). These parameters are different in each machine and are calibrated at compile-time. This part of the query optimizer is not modified in this project.

In this study, we also follow the above strategy to estimate the power cost of plans. We first define a static power profile for each basic operation in query processing and maintain such power costs as system parameters of the DBMS. By this, we can calculate the power cost of an entire plan given its operation vector. The power profile of basic operations for our experimental system is shown in Table III. The initial values of these parameters (data not shown) were obtained from the specifications of the hardware components we find in our server as in I and II and divided by related estimated time from PostgreSQL. For example, in a CPU of active power (i.e., peak power minus the idle power) 20 watts, PostgreSQL estimates the time to process one non-indexed tuple to be 0.04ms, the estimated CPU power per tuple would be 20W times 0.04ms and divided by one second – the result is 0.75mW. The power estimation for disk and memory follows the same way.

TABLE IV

HARDWARE POWER SPECIFICATIONS IN THE EXPERIMENTAL DATABASE SERVER. UNIT FOR ALL DATA IS WATT.

	CPU	Memory	Disk	Total
Power	98	$9 * 4 = 36$	2.63	136.63

Moreover, in calculating the power costs, we assume (as did in [29]) "the peak power consumption of an entire system during the measurement interval is identical to the aggregate of the individual nameplate power consumption". As for main memory, the approximate power consumption is set to 9 watts per DIMM module, as suggested by [13]. The power consumption of disk drives and CPU are obtained from the manufacturers' web sites. The power specifications of the main hardware components in our experimental system are listed in Table IV.

Note that the hardware specifications are provided by the hardware vendors and are best-effort estimations of the real parameters. In order to get more realistic estimations, we run calibration tests to get the final per-operation power costs listed in Table III. More details about model calibration will be presented in IV-C.

The power cost of a plan is calculated from those of the higher-level operations, which consist of basic operations shown in Table III. Table V lists the formulae for the computation of the power cost of single-relation scans via different access methods and two-table joins. Clearly, these formulae use the values of basic operations as building blocks. Again, we follow the exact same mechanism for calculating time costs in PostgreSQL to generate these formulae.

TABLE V

POWER COST FUNCTIONS FOR ACCESSING SINGLE RELATIONS AND JOIN OPERATIONS. T IS NUMBER OF TUPLES TO RETRIEVE; N IS THE PAGES OF DATA TO READ FROM DISK; THE THREE GREEK LETTERS, α, β AND τ ARE DEFAULT CPU TUPLE POWER COST, DEFAULT INDEXED CPU TUPLE COST, AND DEFAULT AVERAGE PAGE POWER COST SHOWN IN TABLE III; n IS THE TIMES OF LOOP AND C IS A CONSTANT.

Methods	Cost function
Seq Scan	$\alpha T + \beta N$
Index Scan	$\tau T + \beta N + C$
Bitmap Scan	$n(\tau T + \beta N) + C$
Subquery Scan	$\alpha T + \beta N$
Function Scan	$\alpha T + \beta N$
Nested loop	$E(outer - paths)$ $+n * E(inner - paths)$
Merge	$n * (E(outer - paths)$ $+E(inner - paths))$
Hash	$E(outer - paths)+$ $n * E(inner - paths) + C$

B. Plan Evaluation Model

The evaluation model is actually a criterion used to evaluate the superiority of alternative query plans towards an optimization goal. The model of current PostgreSQL is simple as it only involves the processing time. In this study, we need a criterion to reflect adjustable tradeoffs between power cost (denoted as P) and processing time (denoted as T). In general, this cost should a function of P, T, and n, which is the relative weight put on the two former factors. Relate this to Fig. 2, all points with the same cost will form a pareto frontier whose shape is determined by the function and parameter n. In this paper, we use a model as suggested by various research projects in hardware [25] and software [21], [31] systems.

Specifically, an metric model of the following format is adapted:

$$C = PT^n = ET^{n-1} \tag{1}$$

where C is the aggregated cost and n is a coefficient that reflects the relative importance of P and T. Intuitively, it means we are willing to sacrifice a d^n-time degradation in performance to achieve a d-time power reduction. The model is general in that we can be used for different optimization goals with the choice of n. When $n = \infty$, we only consider the time cost (i.e., as in the original PostgreSQL does. In practice, we simply disregard P by setting $C = T$.); for $n = 0$, we optimize towards lowest power consumption; and for $n = 1$, power and time performance are both taken into consideration – the cost of a plan is basically its energy consumption.

C. Power Model Calibration

An iterative approach is used for calibrating parameters α, β, τ. Some assumptions are made in such calibration process. First, the environment, such as temperature, affects the result a lot. We assume all the benchmarks are running under the same humidity and temperature. Another important factor in power evaluation is the complexity of other components in the experimental system such as competing applications and system balance [29]. We ignore those factors to simplify the calibration process.

TABLE VI

POWER COSTS OF THREE SINGLE-TABLE SCAN OPERATIONS IN OUR MODEL VERIFICATION EXPERIMENTS. TABLE SHOWS BOTH ESTIMATED COSTS (FROM THE REFINED BASIC OPERATION COSTS LISTED IN TABLE III) AND REAL COSTS MEASURED BY POWER METER.

	Estimated Power (w)	Measure Power (w)	Difference
Seq Scan	24.1	20.2	7.2%
Index Scan	35.2	30.8	14.5%
Bitmap Scan	30.5	28.1	8.5%

We use workloads consisting of simple queries whose execution plans use only one of the following table scan methods: sequential scan, index scan and bitmap scan. The calibration procedure is as follows:

- Step 1. Select one workload, execute it with the initial α, β, and τ values. Record the power consumption using a power meter;
- Step 2. Compare the estimated power consumption with the real power cost. If the difference is under 20%,[5] then keep the current values of α, β, and τ. Otherwise, update the three parameters using the real power consumption, number of tuples processed, and number of pages read. The latter two can be obtained from the calibration workload and the new parameters can be calculated from equations listed in Table V ;
- Step 3. Repeat Step 2 by running a different workload; After the test is repeated 1000 times, we terminate the calibration process and the set of parameters obtained will be used as the calibrated values.[6]

We also verified the calibrated parameters by using them to estimate the power costs of high-level operations. Table VI shows the estimated power consumption of the three single-table scan operations calculated from the refined parameters after calibration (in Table III), compared with those measured by power meter in a scan-only query processing experiment. All the queries in these experiments are executed by a single type of basic table scan methods, without any other scan methods or join operations because it is difficult to tell the power consumption of different operations in a complex query plan. For each experiment involving one basic scan method, we feed the PostgreSQL with 100 such concurrent queries and record the power consumption of the whole server.[7] The estimated values are calculated from formulae listed in Table

[5] The 20% threshold used throughout the calibration procedure is borrowed from [29]

[6] In the calibration procedure, the workloads are generated by varying the percentage of queries using three different scan methods(sequential, index and bitmap). Therefore, there could be a large number of workloads to choose from.

[7] Protocols of such experiments are designed to be very similar to those of our main experiments whose details can be found in Sections IV-E and V.

V with the T and N valued obtained from the PostgreSQL optimizer. Table VI shows the difference between power model estimation using the calibrated parameters and actual peak power when executing these simple queries. It is clear that the power model overestimates the total power consumption in the server for single plan processing. However, these estimations are considered close enough to the real values. The difference varies from 7.2% to 14.5% – these are very promising results as compared to 18% reported in [29].

D. Workload

TPC-C and TPC-H are two mainstream benchmarks for testing database performance. They provide constrains and specifications of various simulations of different real-world tasks, such as business oriented ad-hoc queries and concurrent data modifications. These are the areas where large-scale database servers are heavily used therefore match our expectations of a target database system.

In the experiment, we chose TPC-H as our main testing benchmark. The TPC-H benchmark illustrates decision support systems that examine large volumes of data, execute 22 different types of queries with a high degree of complexity. Two salient features of the TPC-H benchmark are: 1) there are official benchmark tools available - these can save us a lot of time in generating workloads following the TPC-H specification; 2) The individual components of the benchmark - the queries - are directly accessible. This provides a chance to study the resource consumption pattern of each query. By this, we can generate different workloads dynamically from query groups that hold common features. For TPC-C, we use an open source implementation named TPCC-UVa,[8] which implements all the specifications and requirements that listed in TPC-C description. We use TPC-C for supplementary tests because, unlike TPC-H, all the queries in TPC-Uva are not directly accessible.

E. Experimental Design

To test our hypothesis and provide further answers to questions listed in Section I, we perform experiments that can be divided into the following three components:

- **Task 1.** Test the performance and power consumption of query processing under standard query workloads provided by the TPC benchmarks to explore the potential of power saving in databases. These tests are performed under different database sizes;
- **Task 2.** Characterize the power profile of individual queries in the TPC-H benchmark for the purpose of identifying sources of power savings (if verified by Task 1) on single-query level. This is expected to build the foundation of synthetic workloads for further exploration on power-aware DBMS;
- **Task 3.** To further explore the power and energy saving potential in DBMSs, we create workloads with different power consumption patterns and test the impact of such patterns on system performance and power/energy costs. The synthetic workload will be generated based on the characterization work in Task 2.

The detailed protocols for the above tasks will be described when we present the experimental results in Section V.

We run all our experiments in a single database server equipped with one Dual-core AMD Opteron Processor (type 2222 SE), 4 GB of main memory, and a single 135 GB hard disk. The DBMS is PostgreSQL release 8.3.3 running on Linux kernel 2.6.14. In all experiments, PostgreSQL is configured to perform heuristic (i.e., non-exhaustive) search in query optimization. Every query, when issued, will be eventually executed until completion. HDD read pre-fetching is disabled and there are no other CPU or I/O intensive applications running on the system. At runtime, the instantaneous power consumption of the whole server is recorded using a Watts up? PRO power meter[9] at its highest sampling frequency – one hertz. In the next section, we will show that this frequency is sufficient.

V. Experimental Results

In this section, we present the results of the three tasks. In all discussions related to power consumption, only the power used for processing the workload will be presented. We call this the *active power*, which is basically the measured power during query processing less the idle power of the server.

A. Main results (Task 1)

In this subsection, we present the results of a series of experiments running the standard workload provided by both TPC-H and TPC-C benchmarks. Specifically, we set up 100 clients (i.e., the maximum number of clients allowed in PostgreSQL) to send out queries concurrently. For each client, queries are randomly drawn from the pool of all TPC-H queries. All the clients will finish sending jobs within an one-second window, which means the database server will soon reach its maximum utilization and keep working on full throttle afterwards. We configure the DBMS to the plan evaluation model in Eq. (1) with a different time-invariant n value. Specifically, we choose: 1) $n = \infty$, which is basically the performance-only scheme in the original PostgreSQL; 2) $n = 1$, which reflects equal preference on power and performance; and 3) $n = 0$, power-only optimization. We repeat the same experiments under three different database sizes: 500MB, 1GB, and 10GB for TPC-H; 1GB, 5GB, and 10GB for TPC-C.

In Fig. 4 and 5, we plot the instantaneous power consumption of the workload in the first 800 seconds of each experiment.[10] We can clearly see that, system runs on significantly lower power when we choose a query evaluation model that favors low-power plans. When we compare the energy-only ($n = 0$) with the performance-only ($n = \infty$) results, we

[8] http://www.infor.uva.es/~diego/tpcc-uva.html

[9] http://www.wattsupmeters.com

[10] More data is available but not plotted here. However, they do not provide more information since, in all experiments, power stabilizes after the first 800 seconds. The total workload processing time can be found in Table VII.

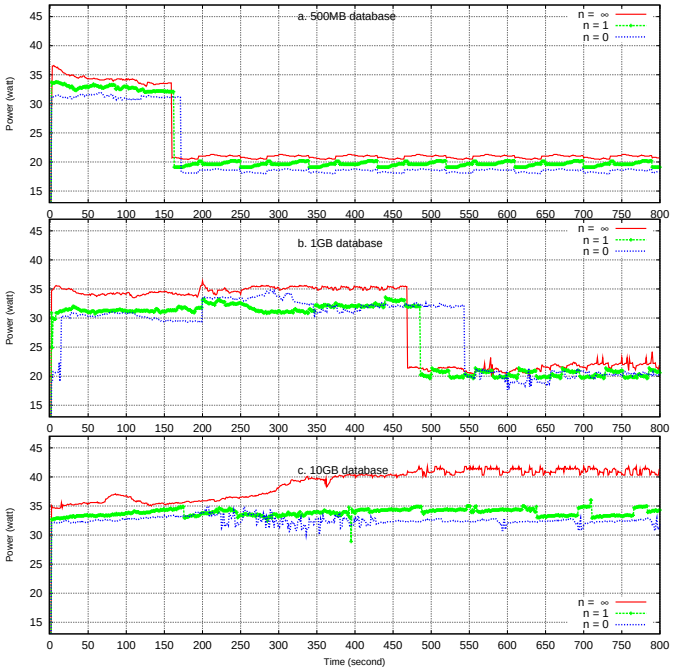

Fig. 4. Power consumption of the TPC-H workloads under three different database sizes.

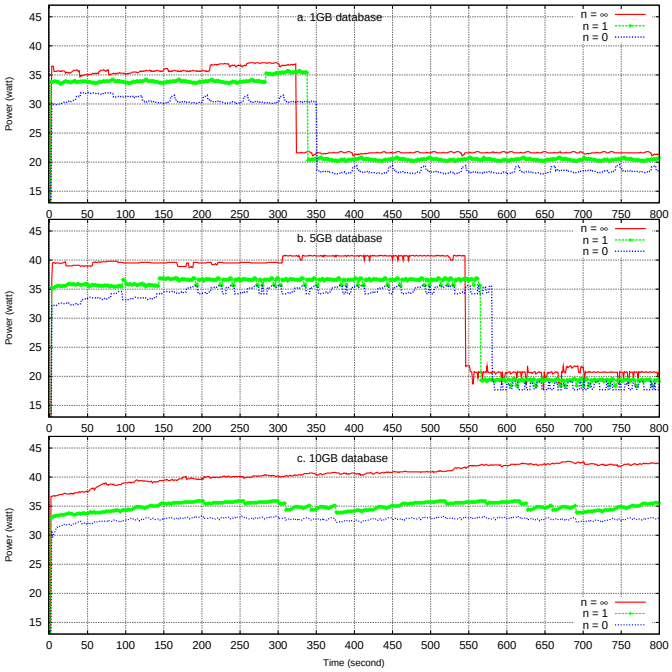

Fig. 5. Power consumption of the TPC-C workloads under three different database sizes.

observe a stable difference of power with magnitude of 7 to 9 watts, which corresponds to power savings ranging from 11% to 22% (Table VII). Without a surprise, the savings of the $n = 1$ case are smaller than those created by the power-only experiment, but it still achieves 11 - 16% power saving. The experiments using the TPC-C workload show comparable levels of power saving.

The power values reported in Table VII are the average power over the entire workload processing period. One might argue that this is not accurate because we may have missed power spikes under the one-second sampling period. However, by reading the data, we believe this should not be a concern. Note that, in all cases, the measured power level is very stable over time without any significant fluctuations. We could have missed some power spikes, but it is very unlikely to miss all of them. In fact, this is also the result of our experimental design: the workload contains heterogeneous queries sent from 100 clients, the DBMS is able to schedule the resources among these concurrent query sessions, giving rise to smooth resource usage at runtime.

Another side of the story is the performance degradation of power-aware DBMSs. In Table VII, we show the total query processing time of the TPC-H experiments. It is obvious that it took more time for the power-aware systems to finish processing all the queries, but the differences are small as compared to the differences of power. As a result, we still yield total energy savings of 3 - 7 %.

It is interesting that, in the experiment under the 500MB database (Fig. 4a), we see a sudden drop of power consumption at the 160th second. This is because the TPC-H workload consists of queries that are bound by I/O and those by computation. All queries in the CPU-bound category finished at around the 160th second therefore the system only runs the I/O bound queries afterwards. The fact that power drops abruptly after the 160th second supports our claim that power consumption of CPUs is significant as compared to that of the disks. From the time when the power drops, we can also see that the processing time of the CPU-bound queries in the power-aware systems are very close to that in the original DBMS. A similar drop can be observed in Fig. 4b, Fig. 5a, and Fig. 5b. At the 200th second in Fig. 4b, we can see a sudden jump of power. We believe this is due to unexpected operating system activities.

An interesting result can be seen from the TPC-C experiment with the 5GB database: the actual query processing time under the power-aware query optimizer (120 mins) is even lower than that under the performance-driven query optimizer (123 mins). This is due to the well-known fact that PostgreSQL does not guarantee the selection of the optimal plan, as a result of the estimation errors imposed by its time cost model. Actually, we also observed this phenomenon in a number of individual queries in the TPC-H experiments: the power-aware optimizer found plans that are superior in both power consumption and actual processing time. It is only in the TPC-C 5GB experiment did we see improvement in the aggregated processing time. This result is positive because the power-aware query optimizer shows even greater potential in power saving than we expected. Its direct outcome is substantial energy savings of 19% and 15.3%.

With the database size increases, more power is consumed

TABLE VII

PERFORMANCE AND POWER/ENERGY CONSUMPTION OF THE EXPERIMENTS USING TPC-H WORKLOAD AND TPC-C WORKLOAD.

DB Size	n	Power (W)	Time (min)	Energy (kJ)	Power Saving	Energy Saving
TPC-H workloads						
0.5GB	∞	23.8	21.05	30.06	–	–
	1	21.1	22.55	28.55	11%	5.1%
	0	20.4	23.88	29.23	14%	2.8%
1GB	∞	36.6	46.55	106.7	–	–
	1	31.9	52.05	99.6	13%	6.7%
	0	30.9	55.29	102.4	16%	4.1%
10GB	∞	41.5	211	525.4	–	–
	1	34.9	242	508.1	16%	3.3%
	0	32.4	263	511.3	22%	3.3%
TPC-C workloads						
1GB	∞	37.5	45.34	102.1	–	–
	1	32.9	48.53	95.79	16%	6.1%
	0	31.1	52.43	97.84	17%	4.2%
5GB	∞	40.8	123	301.4	–	–
	1	34.2	120	245.5	16%	19%
	0	32.1	132	254.3	20%	15.3%
10GB	∞	42.5	223	568.4	–	–
	1	35.9	243	523.8	16%	8%
	0	33.8	263	533.3	21%	7.7%

for all three systems. This is understandable because more data will have to be read and processed to answer the same query. Buffer hit rate will also decrease when the database is larger. In fact, with the 10Gb database, system contention becomes very high – both CPU and disk utilization are close to 100% for most of the time and power consumption goes beyond 40w. The increase of database size has a greater impact on power consumption of the original DBMS: power usage changes from 23.8w to 41.5w in TPC-H and from 37.5w to 42.5w in TPC-C. For the two power-aware DBMSs, the average power does not increase as dramatically as that in the original PostgreSQL. As a result, their power savings increase with the size of database.

B. Query characterization (Task 2)

The purpose of this set of experiments is to investigate the power profiles of individual queries. This helps us further understand the sources of power savings in the standard TPC-H workloads. Based on such knowledge, we can design composite workloads with different query components to study the relationship between power/energy consumption and workload characteristics.

We feed the database engine with workloads containing 50 copies of one single query extracted from the TPC-H workload. A problem for running single-query workloads is that it may give unrealistic query processing time due to excessively high cache hit rate. To solve this problem, we duplicate the database into five copies and each query is run on a different copy of the database. By this, we can avoid the effects of data caching in our experiments. There are 22 queries in the TPC-H tools with various features. Among these queries, we picked 19 because the other three are either single-table queries with one single dominant plan or those with extremely long running time in our current test environment. The database size is 1GB. We compare the power and performance data from running the TPC-H workload with the corresponding estimations given by the (modified) query optimizer, for the purpose of testing the accuracy of such estimations.

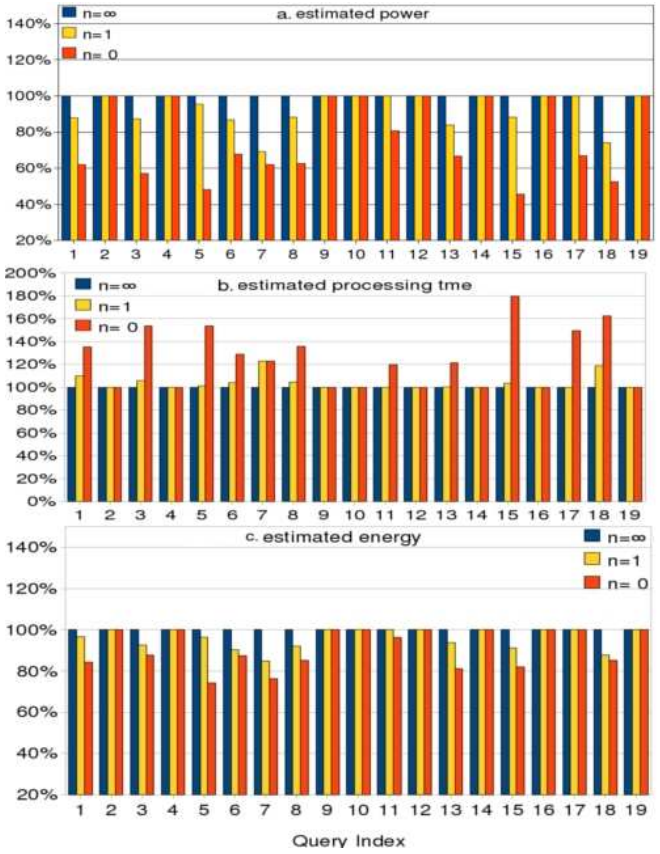

Fig. 6. Power and processing time of the chosen plans for the 19 queries in the TPC-H benchmark as estimated by the query optimizer. Data presented as relative values to the $n = \infty$ case.

Let us first study the estimated power and performance of the plans selected by the query optimizer. From Fig. 6a, we can tell that 11 out of the 19 queries show the potential of power saving. They are: queries 1, 3, 5, 6, 7, 8, 11, 13, 15, 17, and 18. For these queries, naturally, the expected power saving comes with the cost of elongated processing time (Fig. 6b). However, even with the increase of time, most queries still bear a small energy estimation (Fig. 6c). As to those queries that are not expected to provide power saving opportunities, most of them are simple queries (e.g., with joins of only 2-3 tables) that do not have a large search space (of query plans) for the query optimizer to choose from. Thus, the query optimizer ended up choosing the same plan or plans with very similar costs, no matter which n value we use. For the ease of future discussions, we classify all the queries into: 1) *category I* queries: those that are not expected to provide great power saving potential; and 2) *category II*: those that are likely to show large power savings. An interesting pattern we found

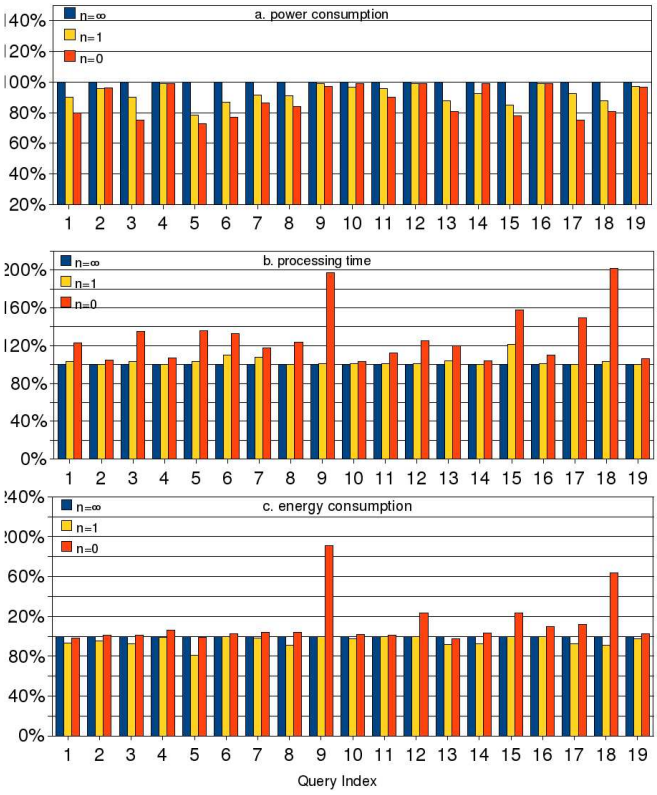

Fig. 7. Power, processing time, and energy consumption of the 19 queries in the TPC-H benchmark as measured by power meter. Data presented as relative values to the $n = \infty$ case.

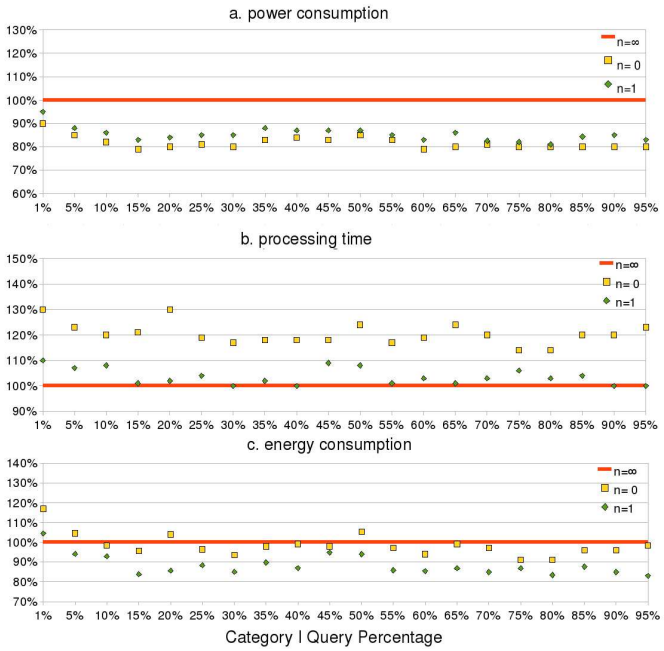

Fig. 8. Relative quantity of power, processing time and energy consumption under workloads with different category I to II query ratios.

is: category I queries are all I/O-bound ones that require a lot of table scanning while most queries in category II are CPU-intensive jobs. This means that, by offering choices of plans with different CPU usage patterns, we can achieve power savings. This, again, supports our claim that CPU plays an important role in the paradigm of power-aware computing.

In comparison, the results of the real experiments are interesting. Fig. 7a shows that, for all the queries, running the modified DBMS engine leads to power savings to some extent. First, for the category I queries, (which should not show any power saving) there are two cases: 1) those whose execution plans are not changed at all. Their power savings are the results of random system errors at run-time; 2) those whose execution plans are different under different n values. The reason why the running power is lower in the situations of $n = 0$ and $n = 1$ could be (run-time) system modeling errors plus modeling errors in the estimation of energy and performance costs. On the other hand, power savings, although not as significant as those indicated by the estimated values (Fig. 6a), can be observed. The real problem is the performance data shown in Fig. 7b: when choosing power-only optimizer (e.g., $n = 0$), all queries have unexpectedly long running time, which gives negative energy savings in most cases (Fig. 7c). This shows that the power-only query optimizer could be an undesirable choice in the PDBMS design. By comparing the results with the estimated costs in Fig. 6, we also see that the method of using multiple copies of the same query has serious limitations in capturing the real (power and time) costs of a query.

While the design of experiments to test the costs of single query is an interesting challenge for future work, we can safely conclude from Fig. 7 that power savings do exist. Salient examples are queries 5, 15, and 17, which show savings up to 25%, while a few others showing savings up to 20%. Another lesson learned is: choosing extreme values of n could yield performance that is too poor to bring any direct energy benefit, it is suggested to choose moderate n values in order to get balanced performance and energy.

C. Multiple workloads (Task 3)

In this set of experiments, we again set up 100 virtual clients. First, we use one of them to send queries from the category I pool and the other 99 clients send queries from the category II query pool. Each query in the pool will be picked by equal chance and sent out to the server concurrently. We repeat the experiment under different ratios of clients (ranging from 1:99 to 95:5) from the two query pools. We use the TPC-H benchmark and set the database size to 1GB.

Fig. 8 shows the power and performance data normalized by the those of the original DBMS. In Fig. 8a, we observe power savings in all 20 cases. However, the savings stay in a relatively steady state as the percentage of category I query increases. On the other hand, relative query processing time has little difference in all workloads (Fig. 8b). Thus, we can still see savings on total energy consumption (Fig. 8c) in most of the cases. As compared to the power-only ($n = 0$) optimizer, the one with the more balanced ($n = 1$) cost model

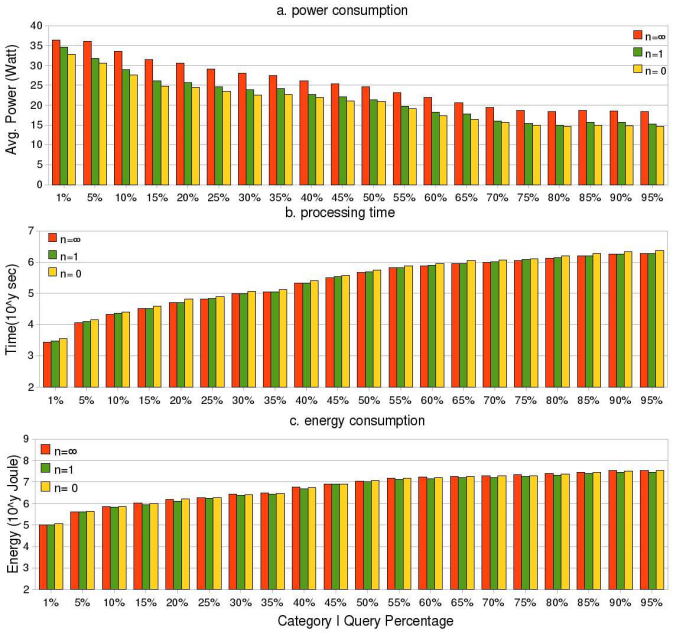

Fig. 9. Absolute quantity of power, processing time and energy consumption under workloads with different category I to II query ratios. Processing time and energy are plotted on logarithmic scale.

achieved similar power savings yet much better performance, resulting in more energy savings.

With the percentage of category I queries increases, power usage decreases in roughly a linear manner in all three systems (Fig. 9a). However, the processing time increases exponentially (Fig. 9b) – it increases from about 45 minutes in the 1% case to about 22 days in the 95% case! Naturally, this trend is carried over to the energy consumption data (Fig. 9c), although the power decreases.

From these results, one can clearly see that query composition in the workloads does not affect the power saving potential of our query optimizer design. We achieve a considerable margin of power saving despite the dramatic changes of query processing time. By visiting Fig. 8a, it seems 20% is a reasonable upper bound for power savings one can reach in any TPC-H workloads.

D. Discussions

The above results clearly show significant power savings (up to 22%) of an initial design of PDBMS. The direct energy savings seem to be marginal in most cases. Yet it also reach 15.5% and 19% in one set of experiments, giving us much enthusiasm to our approach. In summary, we believe our findings are meaningful for the following reasons.

First, even with the marginal energy savings, the economical impact of the envisioned PDBMS will be great. For example, running a single high-performance 300 W server for one year could consume 2628 KWh of energy, with an additional 748 KWh in cooling this server [5]. The total energy cost for this single server would be 338 US dollars a year (for 0.13/KWh) without counting the costs of air conditioning and power delivery subsystems [5]. With our solution, we can save 17-35 US dollars per server every year (considering 50% of total power needed for idling and the 11-22% power savings we can achieve). Again, this does not count the cost savings achieved by lowered hardware failure and cooling demands.

Second, we only used a primitive power-aware DBMS to capture power-saving opportunities. Given the models we have and the tradeoffs we identify, we have reasons to believe that system power usage could be further lowered with more sophisticated cost modeling mechanisms and control algorithms. For example, an adaptive control solution can be designed to guarantee performance by changing the parameters and even the structure of the query evaluation model. Another paradigm we could explore in the foreseeable future is PDBMS running on servers equipped with power-aware hardware, whose power modes can be controlled by the DBMS (via the OS interfaces). Due to the reasons described in Section I, our method can be combined with OS-level methods for substantially higher level of power savings.

VI. Conclusions and Future Work

In this paper, we are interested in identifying power saving opportunities in current DBMSs. We argue that query optimization mechanisms in traditional DBMSs could easily miss query plans that are highly power-efficient and yet lead to little degradation of performance. Therefore, to find such tradeoffs becomes the main task of this study. For that purpose, we perform thorough experiments on the power consumption patterns of various workloads generated from the TPC-H and TPC-C benchmarks in the PostgreSQL system. We believe we have shown a clear picture of the existence of power-performance tradeoffs, and thus the great potential of power conservation in databases. We have observed active power savings up to 22% and total energy savings up to 19%. In the power-aware research field, these are exciting numbers. Another conclusion we can draw is that designing power-aware query optimizer is a promising direction to enable power conservation: all the above findings are revealed by using a modified PostgreSQL query optimizer equipped with a power-aware query evaluation model. On the other hand, we also demonstrated stable power saving potentials in a wide range of workloads.

Since this paper serves as a testimony of the great economical and technical value of the topic of power-aware data management, abundant future research opportunities can be foreseen. Our vision is to build an unified power-control framework in parallel to the regular query processing modules in the DBMS. Modeling errors in both energy and performance estimation would be a major problem to attack. Specific tasks include refined energy models that capture the dynamics in the system (instead of the static model we used in this paper); search algorithms in the query plan space; advanced cost evaluation criteria; and more importantly, resource management algorithms to lower the baseline power consumption in the server.

ACKNOWLEDGMENT

The authors would like to thank Yefu Wang (University of Tennessee) for his help with setting up the experimental platform and Dr. Ken Kristensen (University of South Florida) for insightful discussions.

REFERENCES

[1] SPECpower_ssj2008, http://www.spec.org/power_ssj2008, 2008.

[2] R. Agrawal, A. Ailamaki, P. A. Bernstein, E. A. Brewer, M. J. Carey, S. Chaudhuri, A. Doan, D. Florescu, M. J. Franklin, H. Garcia-Molina, J. Gehrke, L. Gruenwald, L. M. Haas, A. Y. Halevy, J. M. Hellerstein, Y. E. Ioannidis, H. F. Korth, D. Kossmann, S. Madden, R. Magoulas, B. C. Ooi, T. O'Reilly, R. Ramakrishnan, S. Sarawagi, M. Stonebraker, A. S. Szalay, and G. Weikum. The claremont report on database research. *Commun. ACM*, 52(6):56–65, 2009.

[3] R. Alonso and S. Ganguly. Energy Efficient Query Optimization. Technical report, Matsushita Info Tech Lab, 1992.

[4] D. Anderson, J. Dykes, and E. Riedel. More than an interface—SCSI vs. ATA. In *Proceedings of the 2nd USENIX Conference on File and Storage Technologies (FAST)*, 2003.

[5] R. Bianchini and R. Rajamony. Power and energy management for server systems. *IEEE Computer*, 37(11):68–74, 2004.

[6] P. Bohrer, E. Elnozahy, T. Keller, M. Kistler, C. Lefurgy, C. McDowell, and R. Rajamony. The case for power management in web servers. In R. Graybill and R. Melhem, editors, *Power-Aware Computing*, pages 261–287. Klewer Academic/Plenum Publishers, 2002.

[7] D. Brooks and M. Martonosi. Dynamic thermal management for high-performance microprocessors. In *Proceedings of the 7th International Symposium on High-Performance Computer Architecture (HPCA)*, 2001.

[8] J. S. Chase, D. C. Anderson, P. N. Thakar, A. M. Vahdat, and R. P. Doyle. Managing energy and server resources in hosting centers. In *Proceedings of the 18th ACM Symposium on Operating Systems Principles (SOSP)*, 2001.

[9] Y. Chen, A. Das, W. Qin, A. Sivasubramaniam, Q. Wang, and N. Gautam. Managing server energy and operational costs in hosting centers. In *Proceedings of ACM International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS)*, 2005.

[10] Y. Chen, T. Wang, J. M. Hellserstein, and R. H. Katz. Energy Efficiency of Map Reduce, http://www.eecs.berkeley.edu/Research/Projects/Data/105613.html, 2008.

[11] A. P. Conversion. Determining Total Cost of Ownership for Data Centers and Network Room Infrastructure. ftp://www.apcmedia.com/salestools/CMRP-5T9PQG_R2_EN.pdf, 2005.

[12] M. Elnozahy, M. Kistler, and R. Rajamony. Energy-efficient server clusters. In *Proceedings of the 2nd Workshop on Power-Aware Computing Systems*, 2002.

[13] X. Fan, W.-D. Weber, and L. A. Barroso. Power provisioning for a warehouse-sized computer. In *ISCA '07: Proceedings of the 34th annual international symposium on Computer architecture*, pages 13–23, New York, NY, USA, 2007. ACM.

[14] W. Felter, K. Rajamani, T. Keller, and C. Rusu. A performance-conserving approach for reducing peak power consumption in server systems. In *Proceedings of the 19th annual international conference on Supercomputing (ICS)*, 2005.

[15] T. Heath, A. P. Centeno, P. George, L. Ramos, Y. Jaluria, and R. Bianchini. Mercury and Freon: Temperature emulation and management for server systems. In *Proceedings of the 12th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2006.

[16] T. Horvath, T. Abdelzaher, K. Skadron, and X. Liu. Dynamic voltage scaling in multi-tier web servers with end-to-end delay control. *IEEE Transactions on Computers*, 56(4):444–458, 2007.

[17] H. Huang, C. Lefurgy, K. Rajamani, T. Keller, E. van Hensbergen, F. Rawson, and K. G. Shin. Cooperative Software-Hardware Power Management for Main Memory. In B. Falsafi and T. N. Vijaykumar, editors, *Power-Aware Computer Systems - 4th International Workshop (PACS)*. Springer, December 2004.

[18] IDC. Solutions for Data Centers' Thermal Challenges, http://www.blade.org/docs/wp/idc_cool_blue_whitepaper.pdf, January 2007.

[19] Institute for Energy Efficiency. UC Santa Barbara, http://www.iee.ucsb.edu/greenscale/, 2008.

[20] J. G. Koomey. Estimating Total Power Consumption by Servers in the U.S. and the World. Technical report, Lawrence Berkeley National Laboratory, URL: http://hightech.lbl.gov/documents/DATA_CENTERS/svrpwrusecompletefinal.pdf, February 2007.

[21] J. Laudon. Performance/watt: the new server focus. *SIGARCH Computer Architecture News*, 33(4):5–13, 2005.

[22] C. Lefurgy, K. Rajamani, F. Rawson, W. Felter, M. Kistler, and T. W. Keller. Energy management for commercial servers. *IEEE Computer*, 36(12):39–48, 2003.

[23] Y.-H. Lu, L. Benini, and G. D. Micheli. Operating-system directed power reduction. In *Proceedings of the International Symposium on Low Power Electronics and Design (ISLPED)*, 2000.

[24] J. Mankoff, R. Kravets, and E. Blevis. Some Computer Science Issues in Creating a Sustainable World. *IEEE Computer*, 41(8):102–105, August 2008.

[25] A. Martin, M. Nystrom, and P. Penzes. Et^2: A Metric for Time and Energy Efficiency of Computation. In R. Graybill and R. Melhem, editors, *Power-Aware Computing*, pages 293–313. Klewer Academic/Plenum Publishers, 2002.

[26] T. Mudge. Power: A First-Class Architectural Design Constraint. *IEEE Computer*, 34(3):52–58, April 2001.

[27] R. Nambiar. 3-Year Energy Cost in TPC Benchmarks, 2008.

[28] C. Patel, C. Bash, R. Sharma, M. Beitelmal, and R. Friedrich. Smart cooling of data centers. In *Proceedings of the ASME Interpack Conference*, 2003.

[29] M. Poess and R. O. Nambiar. Energy cost, the key challenge of today's data centers: a power consumption analysis of TPC-C results. *Proceedings of the Very Large Data Bases (VLDB) Endowment*, 1(2):1229–1240, 2008.

[30] L. Ramos and R. Bianchini. C-Oracle: Predictive thermal management for data centers. In *Proceedings of the 14th IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, 2008.

[31] S. Rivoire, M. A. Shah, P. Ranganathan, and C. Kozyrakis. JouleSort: a balanced energy-efficiency benchmark. In *Proceedings of the ACM International Conference on Management of Data (SIGMOD)*, pages 365–376, 2007.

[32] V. Sharma, A. Thomas, T. Abdelzaher, K. Skadron, and Z. Lu. Power-aware QoS management in web servers. In *Proceedings of the 24th IEEE Real-Time Systems Symposium (RTSS)*, 2003.

[33] United States Environmental Protection Agency. Report to congress on server and data center energy efficiency. Technical report, http://www.energystar.gov/ia/partners/proddevelopment/downloads/EPA_Datacenter_Report_Congress_Final1.pdf, February 2007.

[34] X. Wang and M. Chen. Cluster-level feedback power control for performance optimization. In *Proceedings of the 14th IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, 2008.

[35] M. Weiser, B. Welch, A. J. Demers, and S. Shenker. Scheduling for reduced CPU energy. In *Proceedings of the First Symposium on Operating Systems Design and Implementation (OSDI)*, 1994.

[36] C. Xian, Y.-H. Lu, and Z. Li. A Programming Environment with Runtime Energy Characterization for Energy-Aware Applications. In *Proceedings of the International Symposium on Low Power Electronics and Design (ISLPED)*, pages 141–146, 2007.

[37] L.-T. Yeh and R. C. Chu. *Thermal Management of Microelectronic Equipment: Heat Transfer Theory, Analysis Methods, and Design Practices*. ASME Press, 2002.

[38] H. Zeng, C. S. Ellis, A. R. Lebeck, and A. Vahdat. ECOSystem: managing energy as a first class operating system resource. In *Proceedings of the 10th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2002.

[39] H. Zeng, C. S. Ellis, A. R. Lebeck, and A. Vahdat. Ecosystem: managing energy as a first class operating system resource. *SIGPLAN Notices*, 37(10):123–132, 2002.

[40] Y. Zhu and F. Mueller. Feedback EDF scheduling exploiting dynamic voltage scaling. In *Proceedings of the 10th IEEE Real-Time Technology and Applications Symposium (RTAS)*, 2004.