Rethinking Query Processing for Energy Efficiency: Slowing Down to Win the Race

Willis Lang, Ramakrishnan Kandhan, Jignesh M. Patel
Computer Sciences Department
University of Wisconsin – Madison

Abstract

The biggest change in the TPC benchmarks in over two decades is now well underway – namely the addition of an energy efficiency metric along with traditional performance metrics. This change is fueled by the growing, real, and urgent demand for energy-efficient database processing. Database query processing engines must now consider becoming energy-aware, else they risk missing many opportunities for significant energy savings. While other recent work has focused on solely optimizing for energy efficiency, we recognize that such methods are only practical if they also consider performance requirements specified in SLAs. The focus of this paper is on the design and evaluation of a general framework for query optimization that considers both performance constraints and energy consumption as first-class optimization criteria. Our method recognizes and exploits the evolution of modern computing hardware that allows hardware components to operate in different energy and performance states. Our optimization framework considers these states and uses an energy consumption model for database query operations. We have also built a model for an actual commercial DBMS. Using our model the query optimizer can pick query plans that meet traditional performance goals (e.g., specified by SLAs), but result in lower energy consumption. Our experimental evaluations show that our system-wide energy savings can be significant and point toward greater opportunities with upcoming energy-aware technologies on the horizon.

1 Introduction

Energy management has become a critical aspect in the design and operation of database management systems (DBMSs). The emergence of this new paradigm as an optimization goal is driven by the following facts: a) Servers consume tremendous amounts of energy – 61B kiloWatt-hours in 2006 alone and doubling by 2011 [3]; b) The energy component of the total cost of ownership (TCO) for servers is already high, and growing rapidly. The server energy component of the three-year TCO is expected to dwarf its initial purchase cost [9]. A big contributing factor to this trend is that processors are expected to continue doubling in the number of cores every 18 months, but the performance per Watt doubles at a slower rate of once every two years [8]; c) To make matters worse, typical servers are over provisioned to meet peak demands, and as a result, they are idle or underutilized most of the time. Barroso and Hölzle [7] have reported average server utilization in the 20–30% range; d)

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

Unfortunately, when servers are idle, or nearly idle, they tend to consume energy that is disproportional to their utilization – for example, an idle server can consume more than 50% of its peak power [7].

With these rising energy costs and energy-inefficient server deployments, it is clear that there is a need for DBMSs to consider energy efficiency as a first class operational goal. In fact, driven by requests from its customers, the Transaction Processing Performance Council (TPC) has moved in this direction, and all TPC benchmarks now have a component for reporting the energy consumed when running the benchmarks.

The challenge here is to reduce the energy consumption of a DBMS while maintaining the performance levels that are typically expected and accepted by the end users. Thus, there is huge opportunity for the database community to take on this challenge head-on and find methods to make DBMSs more energy-efficient. In this paper, we tackle the query processing component, and develop a framework for energy-efficient query processing.

To begin, one might think that perhaps doing business as usual might work for energy-aware query processing. Specifically, we already know how to optimize queries for response time/throughput metrics. So it is natural to pose the following question: Is processing (optimizing and executing) the query as fast as possible always the most energy-efficient way to operate a DBMS? As we show in this paper, the answer to this question is no. The reason for this negative answer has to do with typical server operational characteristics and the power/performance characteristics of hardware components in modern servers.

First, typical servers often run at low utilization. This means that a server has many opportunities to execute the query slower, if the additional delay is acceptable. For example, this situation may occur if the Service Level Agreement (SLA) permits the additional response time penalty. Since typical SLAs are written to meet peak or near peak demand, SLAs often have some slack in the frequent low utilization periods, which could be exploited by the DBMS. (The incentive for doing this is particularly high when the DBMS server is running as a hosted cloud service, and any cost savings that do not violate the SLAs, directly adds to the bottom line.)

Second, components on modern server motherboards offer various power/performance states that can be manipulated from software. Of these components, the CPU is currently the most amenable for transitioning between such power/performance states, but nearly every power–hungry component connected to the server motherboard (e.g., memory chips and network cards) is moving towards providing multiple power/performance states for more energy-efficient operation. For instance, recent research has shown that 'hot' power-down (parking) of physical memory DIMMs may save upwards of 40% of system energy consumption [6, 29].

This paper proposes a new way of thinking about processing and optimizing queries. In our framework, we assume that queries have some response time goal, potentially driven by an SLA. The query optimization problem now becomes: *Find and execute the most energy-efficient plan that meets the SLA*.

A crucial aspect of our work that distinguishes it from previous work [31, 32] is our focus on the slack available in performance between the optimal performance and the SLA goal, and leveraging this slack to reduce the energy consumption (and thereby overall operating cost). To the best of our knowledge this is the first paper that proposes looking at query optimization and evaluation in a DBMS from this new perspective.

To enable this framework, we propose extending existing query optimizers with an energy consumption model, in addition to the traditional response time model. With these two models, the enhanced query optimizer can now generate what we call an *Energy Response Time Profile (ERP)*. The ERP is a structure that details the energy and response time cost of executing each query plan at every possible power/performance setting under consideration. The ERP of a query can then be used to select the appropriate "energy-enhanced" strategy to execute the query.

Notice that the framework proposed in this paper is not limited to the single node database environments that were studied by [31, 32], as it can be applied to optimizers for parallel DBMSs as well. Such parallel DBMSs also have system settings that include cluster configuration as well as server configuration [20, 21]. Such considerations for parallel DBMSs is part of future work.

To show the potential and validity of this approach, in Figure 1 we show an actual ERP for a single equijoin query on two Wisconsin Benchmark relations [11]. The plot shows the system energy consumption and response

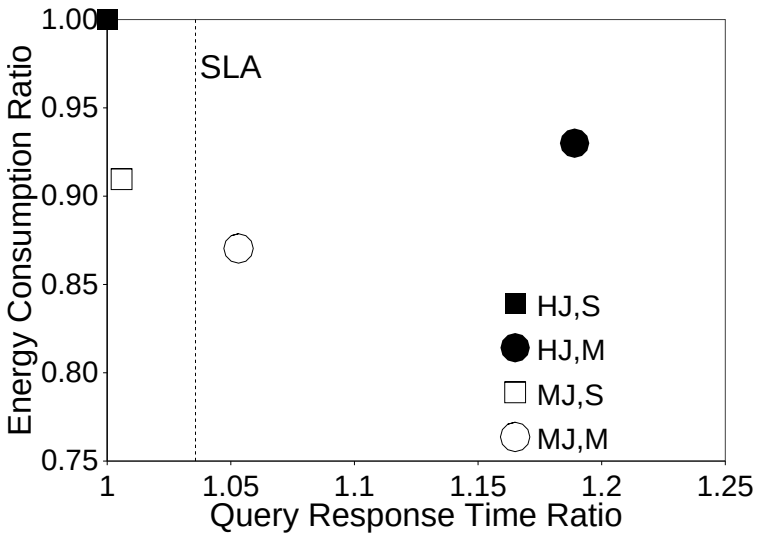

Figure 1: Energy Response Time Profile (ERP) for an equijoin query on two 50M tuple (5GB) Wisconsin Benchmark relations, on the attribute $four$, and a 0.01% selection on both relations. The energy and response time values are scaled relative to the stock settings (HJ, S) that is currently used by DBMSs.

time measurements for executing the query using two different query plans – hash join (HJ) and sort-merge join (MJ) at two different system settings (labeled 'S', 'M'), on an actual commercial DBMS. System setting 'S' corresponds to the "stock" (high power/performance) system settings. System setting 'M' is the power/performance setting that can save energy by reducing the memory capacity [6, 29]. (Details about the experimental setup and systems setting for this experiment is presented in more detail in Section 4.) The energy measurement is the actual energy that is drawn by the entire server box – i.e., we measure the energy drawn from the wall socket by the entire system. In this figure, we have plotted all the other data points proportionally relative to (HJ, S), for both the energy consumption (on the y-axis), and the response time (on the x-axis). Traditional query optimizers often incorporate response time into their primary metric so they would choose HJ and system setting S to execute the query, since this is the fastest query plan. However, by choosing the sort-merge (MJ) plan and system setting M, we can reduce the energy consumption by 13% for only a 5% increase in response time.

However, it may not always be possible to choose a lower energy setting to run the query due to some performance constraint. For example, let us say an SLA exists between the client and the server, which only allows for an additional 4% delay over the optimal response time (HJ, S). This SLA is represented in Figure 1 as a vertically dotted line. Now the most energy-efficient choice is MJ and system setting S, which reduces the energy consumption by 9% and still meets the SLA! *In this case, changing the join algorithm while holding the system settings constant reduces the energy consumption.* This ability to optimize for energy while maintaining performance constraints makes our framework particularly attractive for DBMSs.

To keep the scope of this work limited to a single paper, we focus on the following important class of queries: single join queries with selection and projections. However, our framework can be extended for more complex query processing, using the single join queries as essential building blocks. As the reader will see, our research points to many open research problems in this emerging field of energy-aware data management, only a small part of which we can address in this paper (details in Section 4.3).

This paper makes the following contributions:

- We present a new design for query optimizers that uses both energy and performance as optimization criteria while adhering to performance constraints.

- We present the notion of an Energy Response Time Profile (ERP) that can be used by our query optimization framework to explore all the combinations of system power/performance settings that the hardware provides. The ERP can then be used by the optimizer in picking an "energy-enhanced" query plan that meets any existing response time targets.
- We validate the energy model using an actual commercial DBMS and demonstrate the end-to-end benefit of our approach, which can result in significant energy savings.

The remainder of this paper is organized as follows: Section 2 presents our new optimizer framework, Section 3 describes our energy cost models. The experimental results are presented in Section 4. Related work and conclusions are described in Section 5 and 6 respectively.

2 Framework

In this section we present a general query processing framework that uses both energy and response time as the optimization criteria. The questions we tackle are: (1) How to redefine the job of the query optimizer in light of its additional responsibility to optimize for energy consumption? (2) Given this new role, how do we design a query optimizer? We discuss answers to these questions below.

2.1 New Role of the Query Optimizer

The traditional query optimizer is primarily concerned with maximizing the query performance. The optimizer's new goal now is to find query plans that have *acceptable* performance, but consume *as little energy as possible*. As shown in Figure 1, we want the query optimizer to return an energy-enhanced query plan that is to the left of the SLA-dictated performance requirement, and as low as possible along the y-axis. (We note that performance SLAs are often not rigid and violating SLAs could be compensated by other mechanisms – e.g., some financial compensation. There are potential business decisions to be made about when violating SLAs are okay, but these considerations are beyond the scope of this study.)

The main task here is to generate ERP plots like that shown in Figure 1. To generate these plots, the query optimizer needs to *quickly* and *accurately* estimate both the response time and the energy consumption for each query plan. Query optimizers today are fairly good at predicting the response time, but doing so while accounting for varying hardware configuration is a new challenge. Of course, this challenge of predicting the energy consumption of a given query plan for a specific system setting also requires that the query optimizer understands the power/performance settings that the hardware offers.

2.2 System States, Optimization, and ERP

Driven by the need for energy-efficient computing, hardware components such as CPU [19], memory [29], and NIC [12] are increasingly becoming energy-aware, and offer multiple power/performance settings that can be manipulated directly from software at runtime. Many studies have looked at CPU power/ performance control by capping the maximum CPU frequency [19,31,32] and so we do not focus on system states based on this type of control. We have performed our own CPU studies on a manufacturer-provided instrumented NUMA server (which none of the prior works have studied) that gave direct on-motherboard measurements of the CPU, memory, and northbridge energy consumption. For an in-memory clustered index scan on a commercial DBMS, with this server we found that dropping the maximum CPU frequency by 25% using standard Microsoft Windows Server CPU controls can save 12% in the three component aggregate consumption while being penalized 2% in response time. The reason for this result was that capping the maximum frequency increases CPU utilization and energy efficiency. (We omit these results here in the interest of space.)

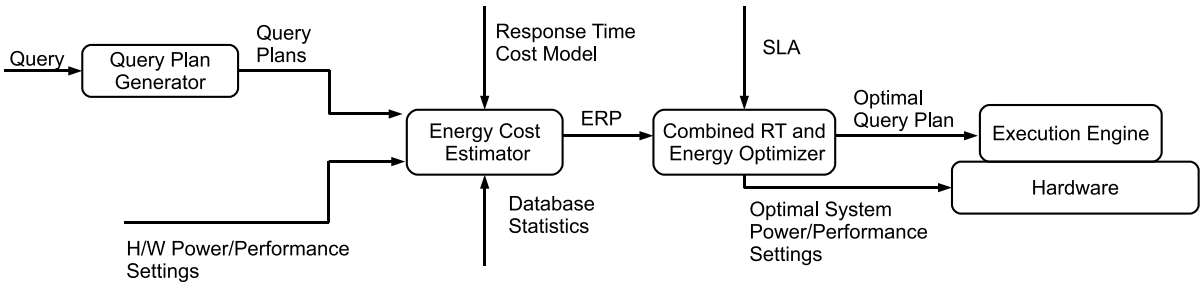

Figure 2: An overview of the framework that optimizes for both energy and response time. The energy cost estimator is described in more detail in Figure 3.

While we saw interesting wins with CPU capping, as mentioned above, this paper focuses on memory-based system configurations, since database query working sets are increasingly completely or nearly fully resident in main memory. In this setting, most of the database processing costs are outside the traditional IO subsystem, which in turn makes memory a significant source of energy consumption (up to 45% and 30% as has been discussed in [29] and [6] respectively). The specific memory mechanism we are exploiting is the ability to "park" main memory DIMMs similar to how CPU cores can currently be "parked". Major manufacturers like Intel have begun developing these mechanisms and methods for exploiting these mechanisms are been actively explored [6,29]. Our idea for using this mechanism is that if the current query does not require all of the available main memory, then some of the DIMM banks can be powered down to save energy. This new ability requires research on issues like physical DIMM-aware buffer pools and the impact this will have on join algorithm selection. (We note that we are simply scratching the surface of various research problems in this space – for example, instead of memory parking one can think of freeing up memory for use by other concurrent task, or considering both memory and CPU power performance states and expanding that scope to consider networking components in a parallel or cluster data processing system. Further discussion can be found in Section 4.3.)

The "energy-enhanced" query optimizer must be provided with a set of system operating settings, where each system operating state is a combination of different individual hardware component operating settings. The optimizer then uses an energy prediction model along with its current response time model to produce an ERP for that query, like the example shown in Figure 1.

Given a query Q with a set of possible plans $P = \{P_1, ..., P_n\}$ that is to be executed on a machine with system operating settings $H = \{H_1, ..., H_m\}$, the ERP contains one point for every plan for every system operating setting (and hence an ERP has $n \times m$ points). Note that heuristics could be developed to prune the logical plans P so that only a small subset of the n plans are explored for specific queries – e.g., plans for where the 'stock' setting does not meet the performance requirement may be assumed to not meet the requirement in all other system settings. (An interesting direction for future work is to explore this option.)

Several methods can be envisioned to predict the energy cost of a query plan. One simple method is to assume some constant power drawn by the system for any specific query such that Energy α Response Time. But, we have found in our analysis that such simple models are not very accurate. Note that an accurate energy estimation model is essential so that the optimizer does not make a wrong choice – such as choosing query plan *HJ* and system setting *M* in Figure 1, which incurs penalties in response time that lie beyond the SLA and does not save enough energy to make this choice attractive.

To enable this new query optimizing framework, we develop an accurate analytical model to estimate the energy consumption cost for evaluating a query plan. We discuss this model in Section 3.

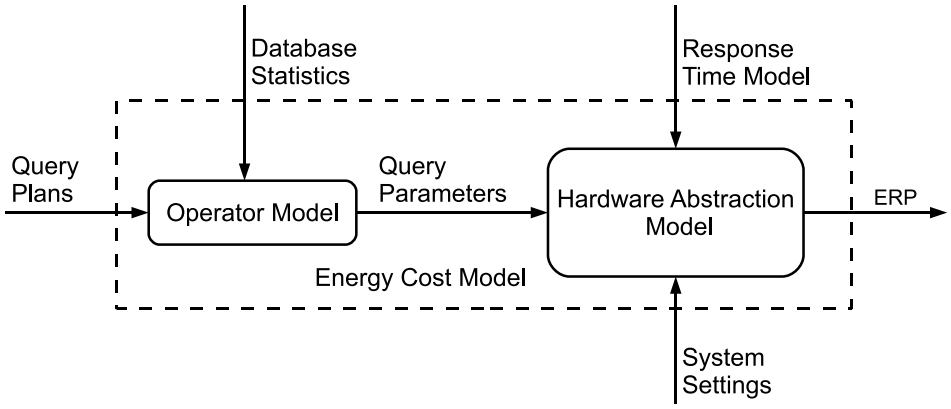

Figure 3: An overview of the energy cost model. The operator model estimates the query parameters required by the hardware abstraction model for each query plan from the database statistics available. The hardware abstraction model uses the query parameters estimated and the system parameters learnt to accurately estimate the energy cost.

2.3 Our Implementation

In this section we discuss our implementation of the framework described above. We had two implementation options – the first one was to integrate this framework into an existing query optimizer in a open-source DBMS like MySQL, and the other to implement it as an external module for a commercial DBMS. We have found that the commercial DBMS that we are using is appreciably faster than the open-source alternative – as much as a factor of 10 in many cases. Consequently, we decided to use the commercial DBMS for our implementation. This choice also forced us to think of a design that is generic and likely to be more portable across other systems, as we have to build the framework using only the high level interfaces that the DBMS provides. (Besides, if one is concerned about energy efficiency, starting with a system that is significantly faster is likely to be a better choice from the energy efficiency perspective.)

Figure 2 gives an overview of our implementation. The query is supplied as input to the query plan generator in the commercial DBMS, which is then requested to list all the promising query plans that the optimizer has identified. These query plans along with the information about available system settings is provided as input to the Energy Cost Estimator, which generates the ERP using an analytical model for predicting the energy consumption (see Section 3). The generated ERP is then used by the combined energy-enhanced query optimizer to choose the most energy-efficient plan that meets SLA constraints. Then, a command to switch to the chosen system operating state is sent to the hardware, followed by sending the optimal query plan to the execution engine.

3 Energy Cost Model

In this section, we briefly describe an analytical model that estimates the energy cost of executing a query at a particular power/ performance system setting. Our model abstracts away the energy cost of a query in terms of system parameters that can be learnt through a "training" procedure, and query parameters (CPU instructions, memory fetches, etc.) that can be estimated from available database statistics.

3.1 Model Overview

We want to develop a simple, portable, practical, and accurate method to estimate the energy cost of a query plan. Unfortunately, prior techniques used to estimate energy consumption fail to satisfy one or more of these goals. For example a circuit-level model of hardware components [1, 2, 13] can accurately predict the energy consumption, but these models also have a high computational overhead which make them impractical for query optimization. On the other hand, a higher level model that treats the entire system as a black box, though simple and portable, is not very accurate.

In our approach, we use an analytical model that offers a balance between these two extremes. In our models, the power drawn by different hardware components are abstracted into learn-able system parameters that are combined with a simple operator model. Figure 3 gives an overview of the energy cost model that we have designed.

The operator model takes as input the query plan and uses the database statistics to estimate the query parameters that is required by the hardware abstraction model to estimate the energy cost. The hardware abstraction model that we describe below required estimations of four query parameters from the operator model: the number of CPU instructions, the number of memory accesses, the number of disk read and write requests anticipated during query execution. Our operator model provided estimates for these four query parameters for three basic operations: selection, projection, and joins. (See [18] for details.) The hardware abstraction model then uses these four query parameters and the response time model (since response time is dependent on system settings) to estimate the energy cost of evaluating a query using a particular query plan at a particular power/performance system setting, essentially computing the ERP.

While we have considered various hardware abstraction models, in the interest of space, we describe the model that we found to be the most accurate. Equation 1 shows the model for average system power during a query as defined by three types of variables:

1. time (T)
2. query parameters for query Q: CPU instructions - I_Q, disk page reads - R_Q, disk page writes - W_Q, and memory page accesses - M_Q
3. train-able system parameters: CPU - C_{cpu}, disk read - C_R, disk write - C_W, memory access - C_{mm}, remaining system - C_{other}). These parameters quantify the component power properties.

The intuition behind this power model is to sum of the average CPU power ($C_{cpu} * \frac{I_Q}{T}$), read power ($C_R * \frac{R_Q}{T}$), write power ($C_W * \frac{W_Q}{T}$), memory power ($C_{mm} * \frac{M_Q}{T}$), and remaining system power (C_{other}) during the duration of the query.

$$P_{av} \quad = \quad C_{cpu} * \frac{I_Q}{T} + C_R * \frac{R_Q}{T} + C_W * \frac{W_Q}{T} + C_{mm} * \frac{M_Q}{T} + C_{other} \tag{1}$$

In the interest of space, we omit the details of deriving the query parameters in this paper. One can model these query parameters using the work of [24, 28] to provide accurate component power draw.

The key features of our energy cost model are:

- **Simplicity:** The models require no additional database statistics other than those used in traditional query optimizers for response time cost estimation. Also, minimal overhead is incurred by the query optimizer in calculating the most energy efficient power/performance operating setting and query plan. The computational complexity is $O(|H| * |P|)$ where H is the set of valid power/performance system settings and P is the set of query plans for the query.

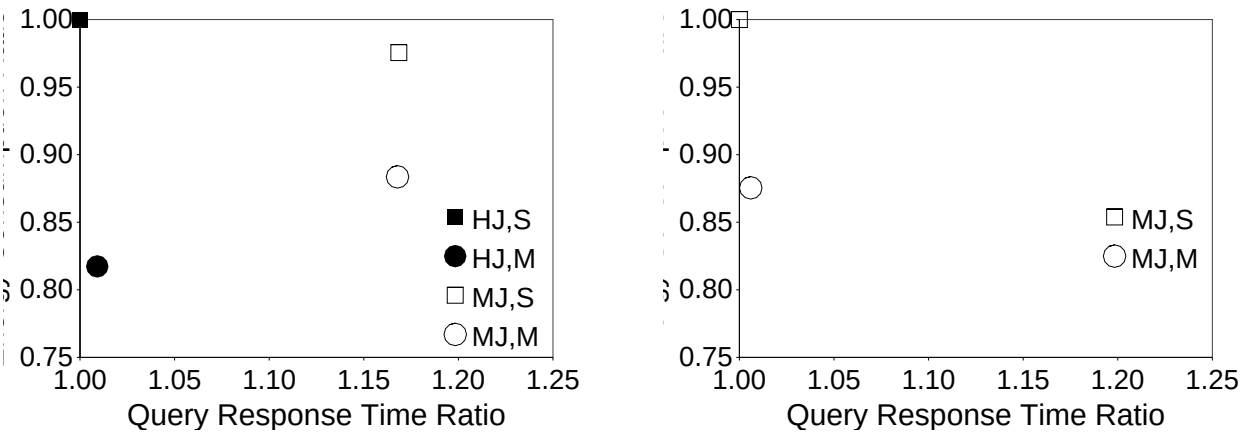

Figure 4: ERPs of two equijoin query classes: (a) Low memory requirement (b) Low memory and I/O heavy. Two join algorithms are used: hash join (HJ) and sort-merge join (MJ); along with 'stock' (S) and low memory (M).

- **Portability:** The model makes few assumptions about the underlying hardware or the database engine internals and can be ported across many DBMSs and machines. In fact, we have been able to implement this model on top of a commercial DBMS, treating it as a black box, and the model still achieves high accuracy.
- **Practical:** Detailed system simulators like DRAMSim [1] and Sim-Panalyzer [2] model hardware components at the circuit level to estimate power consumption. This process though accurate is computationally very expensive, and is hence not practical for use in a query optimizer. In our model we abstract the power drawn by different components into learn-able system parameters.
- **Accuracy:** In our tests, our models have an average error rate of around 3% and a peak error rate of 8%.

4 Evaluation and Discussion

In this section we will present results demonstrating the potential benefit of our optimization framework on a commercial DBMS using end-to-end measurements.

4.1 System Under Test

The system that we use in this paper has the following main components: ASUS P5Q3 Deluxe Wifi-AP motherboard (which has inbuilt mechanisms for component-level power measurements), Intel Core2 Duo E8500, 4x1GB Kingston DDR3 main memory, ASUS GeForce 8400GS 256M, and a 32G Intel X25-E SSD. The power supply unit (PSU) is Corsair VX450W. System power draw was measured by a Yokogawa WT210 unit (as suggested by SPEC and TPC energy benchmarks) connected to a client measuring system. Energy consumption was measured by tapping into the line that supplied power to the entire box, so our measurements and results below are real end-to-end gains. The operating system used was Microsoft Windows Server 2008, and we use a leading commercial DBMS. Note that we use an SSD-based IO system as many studies have shown that SSD-based configurations are more energy efficient [31] and also provide a better total cost of ownership because of their higher performance-per-$ [4].

Table 1: Both queries have the template (SELECT * FROM R, S WHERE <predicate>). These queries are used for the ERP plotted in Figure 4. All relations are modified (100 byte tuples) Wisconsin Benchmark relations.

Query	Predicate	Size of R, S
A	`R.unique2 < 0.1*||R|| AND R.unique1 = S.unique2`	1GB, 1GB
B	`R.unique2 = S.unique2`	5GB, 5GB

4.2 End-to-End Results: ERP Effectiveness

We now present end-to-end results using the techniques that we have proposed in this paper. We use the two system settings described in Section 1; namely, (1) **S** – the default stock settings of our system, and (2) **M** – a reduced memory setting where the memory is reduced from 4GB to 2GB.

In the interest of space, here we present results with only the two queries shown in Table 1. Both queries are join queries on two modified Wisconsin Benchmark [11] tables, where the tuple length has been reduced to 100 bytes. Query A is a 10% selectivity join on two small 1GB tables while Query B is a full join on the sorted keys of two 5GB tables.

First, let us examine the scenario when switching to a lower power/performance state has little effect on the response time. With only 2GB of memory, we expect that a query whose peak main memory requirement is less than 2GB will take approximately the same amount of time to execute, and hence will provide significant energy savings. Figure 4 (a) shows the ERP of query A in Table 1. As we can see, retaining the same hash join plan but using system setting 'M' *reduces the energy consumption by 18% but increases the query response time by only 1%.* In comparison, changing the join algorithm to sort-merge incurs significantly higher response time penalties.

Query B in Table 1, is an equijoin on two tables that are clustered on the join attribute 'unique2'. The two 5GB tables are relatively large compared to the amount of available main memory, but since the tables are already sorted on the join attribute the inputs do not need to be sorted before joining using a merge join operation. Query B is I/O-intensive, requires minimal computation, and has a low peak memory requirement. For this query, as shown in Figure 4 (b), using the sort-merge join along with reducing the memory requirement produces a win of 12% energy savings for less than 1% performance penalty. The response times for the hash join plans are far greater than sort-merge for Query A, and are therefore left out of Figure 4 (b).

4.3 Summary

Our preliminary results described above shows that significant (>10% for the queries above) energy savings can be attained by careful optimization of both query plans and system settings in a commercial DBMS based on end-to-end measurements. We now summarize key takeaways messages from our study.

Energy-Aware Query Optimization: Current DBMS query optimizers can be made energy aware using our modular optimization framework. By introducing a Hardware Abstraction Model (Figure 3) in addition to the traditional Operator Model, we are able to create Energy Response Time Profiles – ERPs (Figure 1). The Hardware Abstraction Model is a train-able model for estimating both the query response time *and* the query energy consumption. ERPs allow us to maximize the query's energy efficiency while adhering to performance SLAs.

SLA-Driven Resource Allocation: The main concept behind the ERP is that by analyzing the relationship between different query plans and different hardware power/ performance settings, we can trade decreased query performance for increased energy efficiency in the presence of slack in SLAs. In some cases, this trade-off is disproportionate, and in some cases, this trade-off can be highly favorable for energy efficiency. By plotting the SLA performance limit on the ERP, we can easily discover the most energy-efficient combination

of query execution plan and hardware system settings while still adhering to the SLA limit. Figures 4 (a,b) show preliminary results where only two plans with two system settings can provide very interesting execution options.

Exploring New Hardware Mechanisms: Traditionally, CPU has been the first target of studies in server energy management [19]. In these results, we have shown that emerging hardware mechanisms such as memory DIMM parking [6, 29] can provide significant energy savings for DBMSs. Other hardware parts such as the networking components [5, 12] are also becoming energy-aware, which will further increase the number of system settings that comprise the ERP in cluster database environments. In addition, other parameters such as cluster size [20] and cluster heterogeneity may also impact the sizd of the ERP space that must be considered.

Complex Queries and Concurrent Queries: The preliminary results presented here point to interesting opportunities for energy savings with simple queries, and it would be interesting to extend this study to more complex queries, and query workloads (e.g., concurrent queries).

5 Related Work

Interest in the area of energy management for DBMSs has begun to grow since the early database publications on this subject [14–16, 19, 27]. Much of the drive has come because of the observation that the energy costs is a big and growing component in the total cost of ownership (TCO) for large server clusters [17, 22, 25].

There are many data center-wide methods for energy management. One popular method is consolidation which forces cluster nodes to be highly utilized and thus, as energy-efficient as possible [10, 26, 30]. Other mechanisms of powering down cluster nodes have been explored in [20, 21, 23]. Cluster level methods can also benefit from the optimization framework we have presented here since our energy optimization module neatly fits in with traditional performance-based optimizers. Cluster level methods can result in improvements in the energy efficiency of data centers and can largely be used orthogonally to "local-level" methods (that work on single nodes) for reducing energy consumption.

Local-level results found in [32] also showed that opportunities for energy-based query optimization exist in PostgreSQL. However, they primarily focused on energy efficiency as the only goal, while our work targets energy efficiency along with SLA constraints. This consideration of SLAs is also a key difference between our work and another recent study [31]. Furthermore, the conclusions from that work suggested that based on traditional server configurations, no new energy-based optimization is necessary because energy efficiency directly follows performance. However, our work shows that when considering SLAs, energy optimal and performance optimal are not always the same operating points.

In summary, we believe that this is the first work to cast energy efficiency as a first-class goal that works in conjunction with performance-based SLA constraints for DBMS query processing. With the move towards cloud computing, we believe this setting is more appropriate and interesting from the perspective of hosted DBMS deployments.

6 Conclusions and Future Work

This paper presents a new framework for energy-aware query processing. The framework augments query plans produced by traditional query optimizers with an energy consumption prediction to produce an Energy Response Time Profile (ERP) for a query. These ERPs can then be used by the DBMS in various interesting ways, including finding the most energy-efficient query plan that meets certain performance constraints (dictated by SLAs). To enable the above framework, a DBMS needs an energy consumption model for queries, and we have developed a simple, portable, practical, and accurate model for an important subset of database operations and algorithms. We have used our framework to augment a commercial DBMS using actual energy measurements, and demonstrated that significant energy savings are possible in some cases.

This area of energy-aware data processing is in its inception, and there are many directions for future work. Some of these directions include extending our framework to more query operations/algorithms, considering more complex/concurrent queries, and considering breaking down "complex" operations such like hash join into smaller components (the build and the probe phase) while also switching between hardware power/performance states. Other promising directions include designing new DBMS techniques to deal with new and evolving power-related hardware mechanisms, studying the application to parallel query optimizers, and working and influencing the development of new hardware features (e.g. memory) to make it more amenable for DBMS to exploit for energy-aware query processing.

7 Acknowledgments

This research was supported in part by a grant from the Microsoft Jim Gray Systems Lab and by the National Science Foundation under grant IIS-0963993.

References

[1] DRAMSim: A Detailed Memory-System Simulation Framework. *http://www.ece.umd.edu/dramsim/*.

[2] The SimpleScalar-Arm Power Modeling Project. *http://www.eecs.umich.edu/~panalyzer/*.

[3] Report To Congress on Server and Data Center Energy Efficiency. In *U.S. EPA Technical Report*, 2007.

[4] A Comparison of SSD, ioDrives, and SAS rotational drives usi ng TPC-H Benchmark. Technical Report White Paper, HP Development Company, 2010.

[5] D. Abts, M. R. Marty, P. M. Wells, P. Klausler, and H. Liu. Energy Proportional Datacenter Networks. In *ISCA*, 2010.

[6] R. Ayoub, K. R. Indukuri, and T. S. Rosing. Energy Efficient Proactive Thermal Management in Memory Subsystem. In *ISLPED*, 2010.

[7] L. A. Barroso and U. Holzle. The Case for Energy-Proportional Computing. *IEEE Computer*, 2007.

[8] C. Belady. In the Data Center, Power and Cooling Costs More than the IT Equipment it Supports. *Electronics Cooling*, 23(1), 2007.

[9] K. G. Brill. Data Center Energy Efficiency and Productivity. In *The Uptime Institute - White Paper*, 2007.

[10] C. Clark, K. Fraser, S. Hand, J. G. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield. Live Migration of Virtual Machines. In *Symposium on Networked Systems Design and Implementation*, 2005.

[11] D. J. DeWitt. The Wisconsin Benchmark: Past, Present, and Future. In J. Gray, editor, *The Benchmark Handbook for Database and Transaction Systems (2nd Edition)*. Morgan Kaufmann, 1993.

[12] C. Gunaratne, K. Christensen, B. Nordman, and S. Suen. Reducing the Energy Consumption of Ethernet with Adaptive Link Rate. *IEEE Transactions on Computers*, 2008.

[13] S. Gurumurthi, A. Sivasubramaniam, M. J. Irwin, N. Vijaykrishnan, M. Jane, I. N. Vijaykrishnan, M. Kandemir, T. Li, and L. K. John. Using Complete Machine Simulation for Software Power Estimation: The SoftWatt Approach. pages 141–150, 2001.

[14] J. Hamilton. Where Does Power Go In DCs and How To Get It Back? http://mvdirona.com/jrh/TalksAndPapers/JamesRH_DCPowerSavingsFooCamp08.ppt, 2008.

[15] J. Hamilton. Cooperative Expendable Micro-slice Servers (CEMS): Low Cost, Low Power Servers for Internet-Scale Services. In *CIDR*, 2009.

[16] S. Harizopoulos, M. A. Shah, J. Meza, and P. Ranganathan. Energy Efficiency: The New Holy Grail of Database Management Systems Research. In *CIDR*, 2009.

[17] J. G. Koomey. Estimating Total Power Consumption by Servers in the US and the World. http://enterprise.amd.com/Downloads/svrpwrusecompletefinal.pdf, 2007.

[18] W. Lang, R. Kandhan, and J. M. Patel. Rethinking Query Processing for Energy Efficiency: Slowing Down to Win the Race. http://cs.wisc.edu/~jignesh/eopt_full.pdf, 2011.

[19] W. Lang and J. M. Patel. Towards Eco-friendly Database Management Systems. In *CIDR*, 2009.

[20] W. Lang and J. M. Patel. Energy Management for MapReduce Clusters. In *VLDB*, 2010.

[21] W. Lang, J. M. Patel, and J. F. Naughton. On Energy Management, Load Balancing and Replication. In *SIGMOD Record*, 2009.

[22] W. Lang, J. M. Patel, and S. Shankar. Wimpy Node Clusters: What About Non-Wimpy Workloads? In *DaMoN*, 2010.

[23] J. Leverich and C. Kozyrakis. On the Energy (In)efficiency of Hadoop Clusters. In *HotPower*, 2009.

[24] S. Manegold, P. A. Boncz, and M. L. Kersten. Generic Database Cost Models for Hierarchical Memory Systems. In *VLDB*, 2002.

[25] C. D. Patel and A. J. Shah. Cost Model for Planning, Development and Operation of a Datacenter. http://www.hpl.hp.com/techreports/2005/HPL-2005-107R1.pdf.

[26] P. Ranganathan, P. Leech, D. Irwin, and J. Chase. Ensemble-level Power Management for Dense Blade Servers. In *ISCA*, 2006.

[27] S. Rivoire, M. A. Shah, P. Ranganathan, and C. Kozyrakis. JouleSort: a balanced energy-efficiency benchmark. In *SIGMOD*, 2007.

[28] L. D. Shapiro. Join processing in database systems with large main memories. *ACM Trans. Database Syst.*, 11(3):239–264, 1986.

[29] M. E. Tolentino, J. Turner, and K. W. Cameron. Memory MISER: Improving Main Memory Energy Efficiency in Servers. In *IEEE Transactions on Computers*, 2009.

[30] N. Tolia, Z. Wang, M. Marwah, C. Bash, P. Ranganathan, and X. Zhu. Delivering Energy Proportionality with Non Energy-Proportional Systems - Optimizing the Ensemble. In *HotPower*, 2008.

[31] D. Tsirogiannis, S. Harizopoulos, and M. A. Shah. Analyzing the Energy Efficiency of a Database Server. In *SIGMOD*, 2010.

[32] Z. Xu, Y.-C. Tu, and X. Wang. Exploring Power-Performance Tradeoffs in Database Systems. In *ICDE*, 2010.