Analysis and Integration of Power-based Models for Query Processing

A PROJECT REPORT

SUBMITTED IN PARTIAL FULFILMENT OF THE

REQUIREMENTS FOR THE DEGREE OF

Master of Engineering

IN

COMPUTER SCIENCE AND ENGINEERING

by

Chhabi Sachan

Computer Science and Automation

Indian Institute of Science

BANGALORE – 560 012

JULY 2013

TO

My Family

&

Friends

Acknowledgements

I thank Prof. Jayant R. Haritsa for his guidance and inspiration for this work. I also thank the reader Prof. Matthew Jacob Thazhuthaveetil, for his thorough and careful review of this work and for valuable suggestions. Apart from them, I was able to get much insight into the problems by the discussions I had with Anshuman Dutt,and my Lab mates. My sincere thanks to them. I would also like to thank my family and friends for all their support. Their support made my stay at IISc memorable.

Abstract

Now-a-days, a lot of research is being focused on building Power-aware systems. In a typical data center, database engines are important consumers of electrical energy due to their complex query processing activities. Thus, optimizing the power consumption of these database servers has become an active research area recently. This motivated researchers to investigate the redesign of the database internals to minimize the energy overheads. Our work deals with the analysis of power-based models for query processing.

A lot of work was already done in this area. The prior literature in this field has dealt with this problem to some extent, where they have proposed energy prediction models for query execution. The work summarized in this report focuses on verifying and analyzing these models for energy prediction. The work also aims at giving directions to build new energy models, which have a more refined and richer feature set, and are more accurate. We began by analyzing the energy model for sequential scan operator and found that some new features may be added to improve its accuracy.

Though modeling is one aspect, equally important an aspect is model integration. The other part of our work deals with designing strategies for integrating peak power models in database servers. One of the strategies designed was also successfully implemented for PostgreSQL, an open-source database server.

Contents

List of Tables

List of Figures

Keywords

Query Processing, Query Optimization, Plan Tree, Degree of Clustering, Selectivity, Periodicity, Dynamic Programming, Inductive pipelines

Notation and Abbreviations

No notation is used in this document.

The following abbreviations have been used:

B: Bytes (as units)

DB: Database

DP: Dynamic Programming

DIMM: Dual in-line Main Memory

IO: Input-output (devices)

MM: Main Memory

PP: Peak Power

W: Watts (as units)

Chapter 1

INTRODUCTION

"Energy Crisis" is one of the most severe problems in the present world, with demand-supply gap rising almost daily. Thus, there is a dire need for efficient energy conservation and utilization techniques.

More and more services are provided over the web these days. These services rely on data centers (also known as digital warehouses). The digital warehouses all over the world used about 30 billion Watts of electricity in 2006, roughly equivalent to the output of 30 nuclear power plants, according to the estimates that the industry experts compiled for The New York Times [16]. Based on the report of U.S. Environmental Protection Agency, "the servers and data centers in U.S.A. alone consumed about 61 billion kilo Watt-hours (kWh) at a cost of $ 4.5 billion". Xu et al showed that electricity consumed by computer servers and cooling systems in a typical data center contributes to around 20% of the total ownership cost equivalent to one-third of the total maintenance cost [13]. Database servers are an integral part of most of the web services (for instance, online banking, shopping portals, etc.). Their complex and intricate data processing activities demand high energy consumption. Therefore, energy-aware query processing engines are highly desirable.

The excessive energy consumption of the data centers inadvertently leads to increase in maintenance costs as well as other unwanted side effects, such as pollution. DBMS is the

major resource consuming application in a data center [9], [11] and [15]. One way to reduce energy consumption of a data center is to reduce the energy consumption of the inherent DBMS [8]. The Claremont report [1] on database research also emphasizes "designing power-aware DBMSs that limit energy costs without sacrificing scalability" as an important research area. Designing such query processing engines is an active area of research. This is evident from the fact in 2007; TPC added a new OLTP benchmark TPC-E [19].

There are two aspects of power that are to be considered during the evaluation of power utilization in DBMS. One is *average power* and the other is *peak power* (discussed later). The database systems that aim to minimize the average power consumption as well as time of execution of a query, are termed *energy-aware database systems*.

There exist both hardware and software approaches for designing energy-aware database systems. The approaches dealing with hardware are called "hardware level optimization", where the emphasis is on using better hardware configuration to minimize the energy consumption of the system as a whole. The hardware level optimization is based on enhancing the architectural features of the machine and is not specific to a particular application. [14] shows few hardware level techniques and their evaluation. This research also established a correlation between energy-efficient configuration and time-efficient configuration. On the other hand, "software level optimization" techniques aim to minimize the energy consumption of a specific application. So in context of DBMS, these techniques attempt to minimize energy consumption of different modules such as query optimizer, executor, etc.

Our work focuses on the software level techniques, and particularly the query optimization module. A query can be executed using a number of different strategies. Such strategies are termed "plans" in DBMS context. Each of these plans is in the form of a tree, referred to as a "plan tree". A query optimizer generates an optimal execution strategy for a particular query. It evaluates a set of strategies and then selects the optimal one with respect to certain performance criterion.

Though time has been the plan selection criterion traditionally, recent research has focused on energy consumption as the plan selection criterion. [8] and [15] have proposed a model for predicting energy consumption of a certain query execution strategy. The query optimizer can then use this model to select the optimal strategy for energy consumption. Our aim in this piece of work is to first verify the set of models proposed, and then analyse them for their feature set.

Developing models for different criteria like time, energy, peak power etc., does not have any importance unless they can be incorporated into a query optimizer. The traditional design strategies for plan selection which are either Dynamic Programming based or Transformation based work well for time and energy as the performance criteria but not for peak power. We thus aimed at designing strategies which use peak power as the performance criterion. In this piece of work, we have proposed a design strategy for peak power model integration into PostgreSQL, an open-source database server.

Chapter 2

RELATED WORK

2.1 Energy

There is an increasing energy demand in database servers today, and its wasteful use not only leads to increased monetary losses but also pollution and other hazardous effects. Thus, designing an energy-efficient database server has become an interesting research area recently. The requests from the customers also have driven the Transaction Processing Performance Council (TPC) to move in this direction, and all TPC benchmarks now have a component for reporting the energy consumed when running the benchmarks. In this regard, TPC has introduced a new benchmark "TPC-E", to address the issues of energy consumption in DBMS [19].

Database community has shown much interest in green database servers over the past few years. Some early work in this field can be found in [4], [5], [6], [9], [14]. Most recent work towards implementing energy-efficiency in DBMSs is either hardware or platform oriented. The hardware optimization techniques proposed in [14]:

- Use Solid State Drives instead of Hard Disk Drives
- Employ CPU Core Parking
- Power down idle memory DIMM banks

In the recent years, much interest has been shifted to the software-level as hardware-level energy management mechanisms can be ineffective without the assistance of energy-aware applications. Some of these software-level techniques are mentioned in [7], [8], [9] and [15].

It has been shown in [8] that processing a query as fast as possible does not always turn out to be the most energy-efficient way to operate a DBMS. This is due to the typical server operational characteristics and the power performance characteristics of hardware components in modern servers. This is explained in detail in [8]. Apart from these, two position papers [8] and [15] provided high level ideas on improving energy- efficiency of databases, and both emphasized on energy-aware query optimization. The work done in [8] attempts to develop models for some specific database operators like select, project and join. Another work [9] proposed query rescheduling and CPU frequency control as two means for green data management and supported their claims with experimental results.

As compared to the above work, we focus on a systematic framework for energy conservation inside a DBMS. The key challenge is to identify the main components in existing DBMSs that lead to low energy-efficiency and develop relevant mechanisms to alleviate the problem. The proposed work, however, is unique in that:

- we consider energy-aware result clustering as a means of energy reduction
- we extend the feature set of the energy models proposed so far

2.2 Peak Power

Peak power is the maximum power consumed during query execution. [7] made an attempt to model the peak power consumption in database servers. In this work, regression modelling was used to predict the peak power consumption by a query in database engine, which was within 15% error range of the actual values. In [7], a pipeline-based model of query execution plans was developed, that accurately estimates peak power consumption. For each of these pipelines, they applied a mathematical function that takes as input the rates and

sizes of the data flowing through the pipeline operators, and outputs an estimate of the peak power consumption.

No attempts, till now, were made towards integrating the Peak Power models into current optimizers. Here we propose various design strategies to deal with this problem.

Organization of Thesis: The thesis is divided into two parts: First deals with verifying and analysing the energy models; while the other deals with proposing various design strategies for integration of peak power models into a database server. We introduce the problem of both the parts in Chapter 3 and Chapter 4 respectively. Chapter 3 further includes details about experimental setup, experiments conducted, their results and some supporting explanation for them. Chapter 4 formulates design strategies for Peak Power Model Integration, further including some details about its implementation and evaluations. In Chapter 5, we summarize our conclusions and give an outlook to the future work.

Chapter 3

VERIFICATION AND ANALYSIS OF ENERGY MODELS

The models presented in [8] and [15] state that energy consumption and time taken are linearly related to work done that is evaluated in terms of Main Memory accessed, IO and CPU used, which in turn depend on database parameters like relation cardinality, tuple width, no. of pages on disk, etc.

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

where,

T : Query Execution Time,

I_Q : No. of CPU instructions,

R_Q : No. of disk pages read,

W_Q : No. of disk pages written,

M_Q : No. of memory pages accessed,

$C_{cpu}, C_R, C_W, C_{MM}$ and C_{other} : tunable parameters.

The above equation quoted in [8] represents a model for computing the average power consumption of a query execution, that takes as input a query plan and uses database statistics to estimate the model parameters I_Q, R_Q, W_Q and M_Q to compute the cost of query plan.

These models were evaluated for some database operators like scan, project and join. We have to verify these models, with certain set of experiments, for other database operators and even full queries involving combination of these operators. Initially, we analysed the proposed model for SEQUENTIAL SCAN.

3.1 System Implementation

3.1.1 Hardware

Our experimental testbed (shown in Figure 3.1) for model verification consists of two computing machines for different purposes. The one called "Server" contains a 2.5 GHz dual-core AMD-Opteron processor, 4 GB of RAM, 2*240 GB 7200 RPM SAS Hard drives. CPU frequency of server machine is fixed at 2.4 GHz. It is used to run the database server and thus the target for our experiments. The other one called "Monitor" is responsible for runtime collection of experimental data including query statistics and power. Monitor is responsible for collecting power per second consumed by the queries executing on Server. Power measurements are done by a digital power meter (Brand Electronics model 201850/CI [17]) with a 1W measurement resolution, 3 kHz sampling frequency and 1 Hz logging. The meter is directly connected between the electrical mains and the database workstation, and therefore measures the workstation's overall power consumption. The power values are transmitted through an interface cable to monitor machine on which they are logged and processed, thereby ensuring that the measurement apparatus does not modulate the monitored system.

3.1.2 Software

Our server machine is installed with PostgreSQL 8.4.10 DBMS under 64-bit Ubuntu Linux 12.04. For handling the CPU frequencies, we use a GUI based tool called "cpufreqindicator" available in Linux.

3.1.3 Power Measurements

Since it is difficult to measure the power consumption of each of the components attached to DB server like motherboard, hard disk, CPU, cooling systems, etc., we have adopted a way for measuring the energy/power consumed by DB servers directly. The power consumed just before the query execution is noted down as "Ambient Power", which is assumed to be the power consumed by the components not involved during query processing.

Ensuring Correctness of DB Power Measurements:

- Ambient Power is subtracted from the overall power consumption of the system to obtain the dynamic DB power. (for the above mentioned setup, Ambient Power=115 W)
- Only essential programs are run.
- Server is not connected to the network (avoids power fluctuations due to network connections).
- Cold Cache Condition[7]: makes sure that no query gets benefited from the data lying in main memory or buffer cache due to execution of previous queries.
  - Flushing Cache: To flush the buffer cache, we use the "drop_cache" command[18] available in Linux.
  - Flushing RAM: To flush the main memory, we restart the DB server and perform sequential scan on an unrelated relation.

3.2 Verification of Models in the Literature

In order to verify the models in [8] and [15], we have performed an exhaustive exploration of sequential scan operator. Our experiments aimed to verify the below mentioned corollary to the claim in [8] and [15].

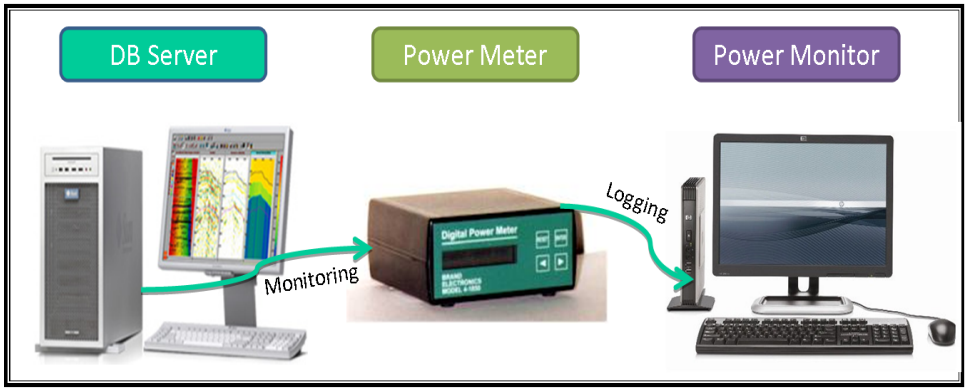

Figure 3.1: Experimental Setup

Corollary: When the work done by queries is identical then the energy consumption and time taken for query execution must also be identical.

3.2.1 The Setting

In these experiments, same relation Lineitem 10 GB from TPC-H [20] benchmark, is sequentially scanned to make IO equal; the output cardinality (50% selectivity) is also equal to enforce identical MM accesses. The difference exists in the underlying data arrangement of result tuples on the disk. We have considered two types of result arrangement:

- Clustered: Result tuples lie in the first half of the relation.
- Alternate: Every second tuple qualifies the predicate and becomes result tuple.

3.2.2 Contradiction

From Table 3.1 and Figure 3.2, it is clear that the energy consumed was not same even though the work done was the same. In fact, a decline can be seen in all the three metrics when the result arrangement was changed from clustered to alternate. A detailed study of why a decrease in the energy consumption and execution time comes is done, while the decrease of Peak power consumption is not studied in this piece of work.

Arrangement	**Energy** (in Joules)	**Time** (in seconds)	**Peak Power** (in Watt)
Clustered	1923	184	12
Alternate	1390	154	9

Table 3.1: Time Gap

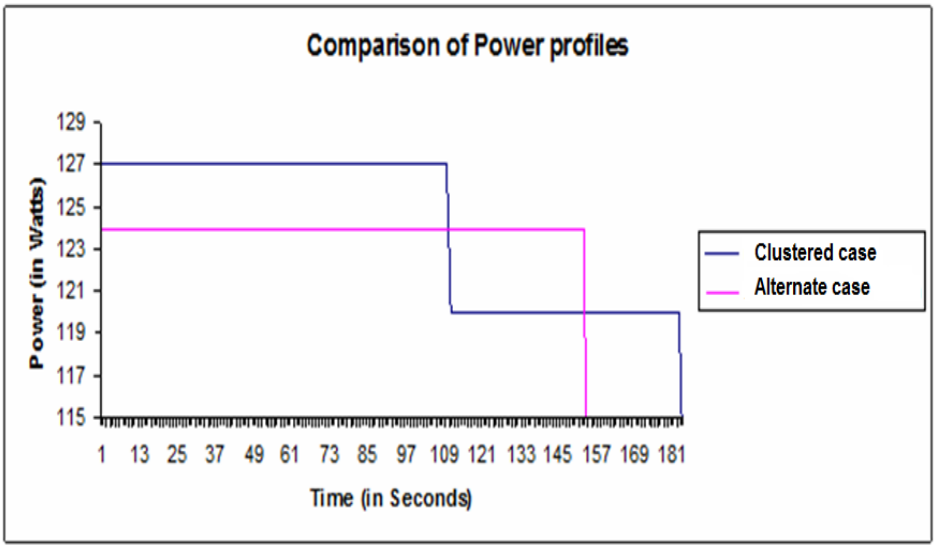

Figure 3.2: Time Gap

Experimentally, it was found that the energy gap and time gap increase with the size of the relation being sequentially scanned. Similar results held across different processor families and different DB parameters like relation size (2GB, 5GB, 10GB, 30GB), tuple size (4B, 50B, 100B, 200B, 250B).

Although we were able to find a new feature which when included to the current feature set of energy models might improve the prediction accuracy, but we are currently unclear about the converting the new feature set in to feasible energy model.

3.3 Explanation of the above behavior

The only parameter that was varied in the experiments conducted was "degree of clustering", which is the extent to which the query result is logically clustered on the disk. This parameter varies from query predicate to predicate, it is dynamic in nature. Measurement of it for a query execution is also not known and studied here. Hence, the usefulness of this new feature for converting to a feasible energy model is unclear at this stage. More investigation in this direction is suggested as a future work.

The behavior observed in previous experiments can be explained by RPW Pipeline Model.

RPW Pipeline Model: In this model, every database operator is composed of three sub-operations, which are defined for sequential scan operator as follows:

- R: tuple read to memory
- P: tuple processed for predicate qualification
- W: tuple written to output buffer if qualified in P

System bus is a constrained resource here. The R operation is either from disk or main memory and hence needs the system bus. The W operation copies the result tuple from one

location of main memory to the output buffer, which also lies in main memory. Hence, W also needs the system bus. Thus, a conflict occurs between R and W operations for usage of system bus. Conflicting R and W operations lead to instruction stalling, there by increasing the execution time in clustered data arrangement.

Figure 3.3 depicts the clustered data arrangement where all the tuples get qualified, therefore the CPU has to perform all the three sub-operations, viz., R, P and W. While in Figure 3.4, which represents the alternate case, for every alternate tuple, the CPU has to perform all the three sub-operations; for rest of the tuples, it has to perform only two operations, viz., R and P.

In the alternate case, the time slots of all the W operations encapsulate those of the P operations. But in the clustered case, the time slots of some of the W operations do not encapsulate those of the P operations. This is the reason for the increase in the execution time for the clustered data arrangement. Moreover, an increase in the size of the relation increases the number of these W operations, which in turn increases the time (energy) gap. This was verified experimentally also where same experiment was conducted for larger relation sizes. It was found that the execution time and energy consumption increased linearly with increase in relation sizes.

3.3.1 Experimental Validation of the Explanation

1. Power Gap

The Setting: Here the same relation Lineitem 10 GB from TPC-H [20] benchmark is scanned but with varying selectivity:

- 100% selectivity: all the tuples qualify the predicate and become result tuples
- 0% selectivity: none of the tuples qualifies the predicate and hence does not become result tuple

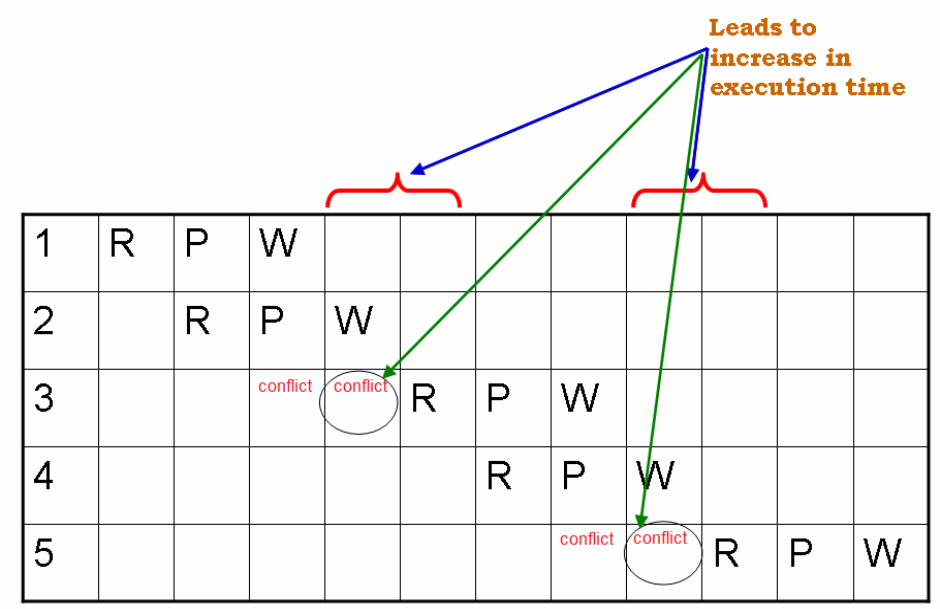

Figure 3.3: Clustered Data Arrangement

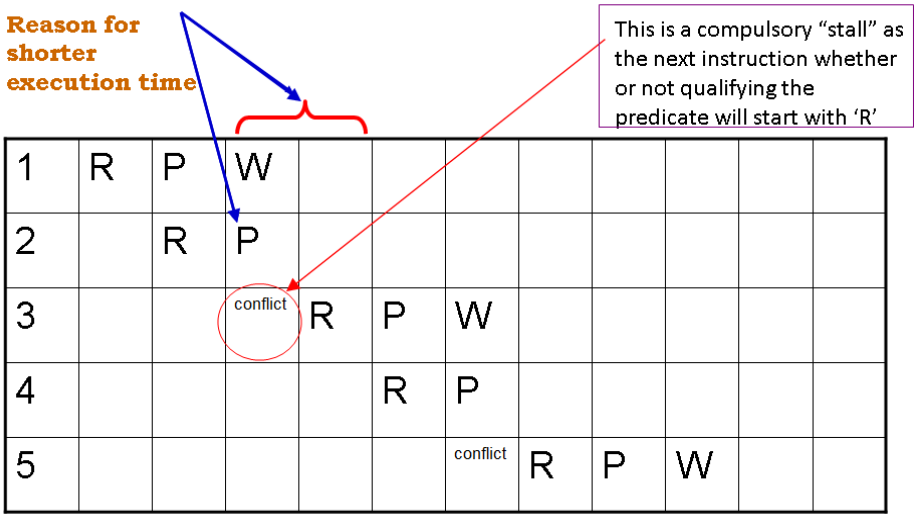

Figure 3.4: Alternate Data Arrangement

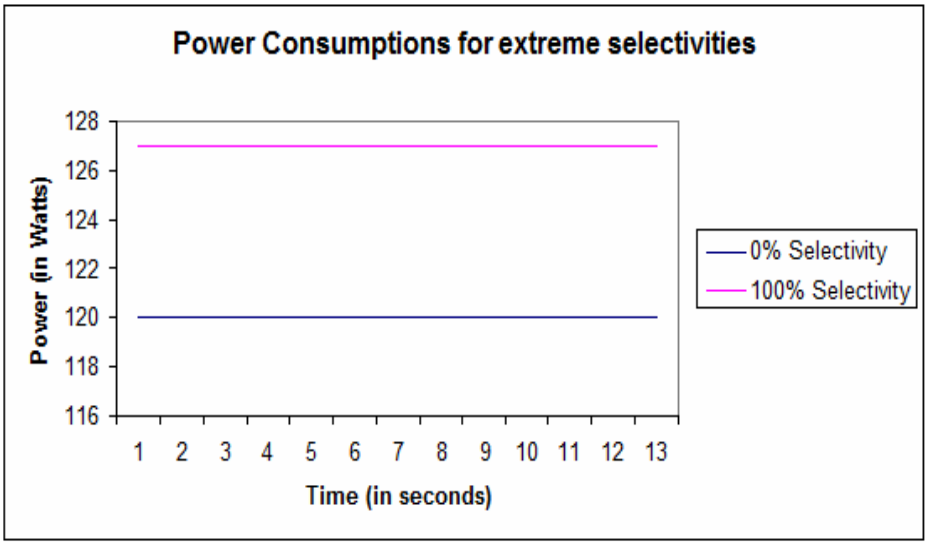

Figure 3.5: Power Gap

Result: Figure 3.5 shows the power profile of the above mentioned two query cases. It is clear from the power plot that power consumption per second in case 1 ($\sim$ 127 W) is much higher than that of case 2 ($\sim$ 120 W).

Inference which can be drawn form the above analysis is that for 50% selectivity, the clustered case will have power consumption of 127 W for half of the execution time and 120 W for remaining half of the execution time. But due to stalling (and time gap), 127 W stays for little longer: 60-70% of execution time.

Explanation: In case 1, every tuple qualifies the predicate. So the system performs R, P and W operations for each of the tuple, while in case 2, only R and P operations are performed. So the extra power of 7 W is consumed by W operation. Hence, the power gap is contributed to the W operations.

Supporting Experiments: Experiments to support the above mentioned claim were done where a C program was written to simulate the process of sequential scan which also gave the

same power gap.

Another experiment executed two programs: one with IO and memcpy()(a C function which copies data form one memory location to another), which involves both disk and MM accesses; while the other one with only IO, which involves only disk operations. The power profile of these programs were plotted and a power difference of around 7-8 W was seen, which verified our claim.

The time gap and the power gap explain the decline in energy consumption from clustered case to alternate case.

2. Periodicity of Qualification

The different result arrangements on disk which lead to time gap in query execution of similar workloads is actually attributed to the periodicity of qualification (or W operations). In clustered case, the periodicity of qualification is maximum, which lead to maximum number of instruction stalls, while in alternate case, the periodicity of qualification is least (for 50% selectivity) which lead to minimum number of instruction stalls and thus lower execution time was seen in this case.

Periodicity of qualification can be represented in %, where 80% periodicity means that 8 tuples get qualified out of 10 tuples. That is, in case of clustered result arrangement, the periodicity of writes is 100% in first half and 0% in later half, while in case of alternate result arrangement, periodicity of writes is 50% throughout, provided selectivity of scan query is 50%.

The Setting: In the following experiments, the same relation is sequentially scanned with 50% selectivity and periodicity of writes was varied from 100% (clustered) all the way down to 50% (alternate) and the effects on execution time were observed.

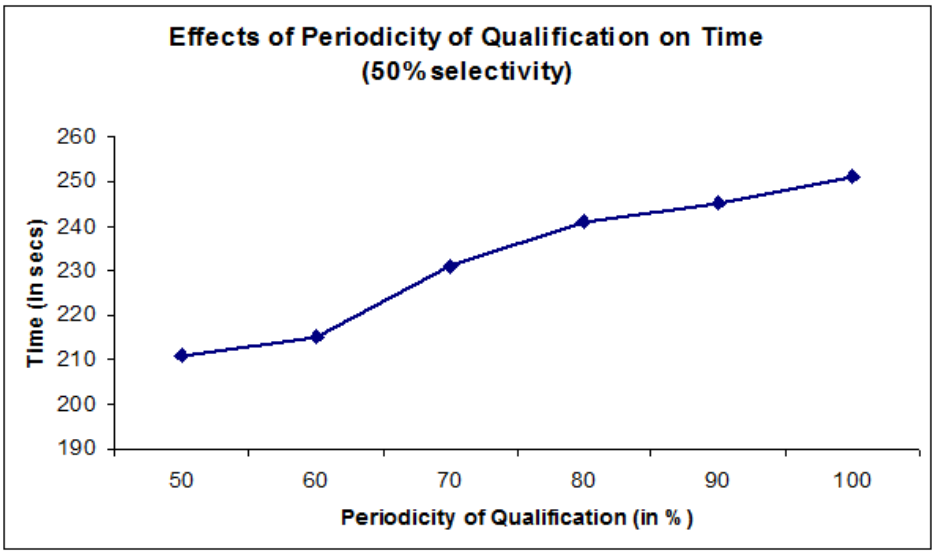

Figure 3.6: Periodicity of Qualification

Analysis: The execution time was observed to be maximum in case of 100% periodicity while minimum in case of 50% periodicity for reasons explained in Section 3.3. The execution time decreased from 100% periodicity (clustered) to 50% periodicity (alternate) as shown in Figure 3.6. Hence, the optimal result arrangement is the alternate case.

3.4 Window of Opportunity

3.4.1 The Setting

We have performed experiments where selectivity was varied all the way from 0% to 100% under both the result arrangements, i.e., clustered and alternate. The tuple width for these experiments was taken to be 109 B, which is the average tuple width for TPC-H relations [20].

3.4.2 Result

The execution times under both the result arrangements were observed and plotted as shown in Figure 3.7. Finally, energy consumed in both the result arrangements was observed and plotted in Figure 3.8.

3.4.3 Explanation

At extreme cases, i.e., 0% and 100% selectivity, the result arrangement does not really matter. With the previously mentioned results and analyses, it is clear that in clustered case, the execution time goes on increasing with selectivity as the number of stalls increase. While in case of alternate or distributed result arrangements, the execution time is constant up to some selectivity (70% in this experiment) and then starts increasing. It is clear from Figure 3.8 and Figure 3.7 that there is opportunity to significantly save execution time and energy consumption if sequential scan is performed on a relation with selectivity between 30-80%. We call this "window of opportunity", as in this bracket of selectivity, just by changing arrangement of result tuples on disk, we can save energy and time.

It was experimentally found that window size depends on tuple width, and it is negligible in case of relations with very small ($\leq$ 4 B) or very large ($\geq$ 250 B) tuple widths. With very small tuple width, the time taken by R/W operations is negligible. Therefore, execution time in this case is attributed only to the P operations, whose number will be same independent of result arrangement on disk. Thus, no effects of different result arrangements can be seen in this case. While with very large tuple width, the time taken by R/W operations is very large which encapsulates the time of P operation of next instruction in pipeline. Thus, no effects of different result arrangements can be seen in this case too.

3.4.4 Analysis

Since average tuple width in case of TPC-H relations is ~ 120 B, we get a decent window of opportunity. An arrangement of result tuples that results in savings in execution time and

energy consumed is called as “careful data arrangement”. For a query involving single column in its predicate, the optimal data arrangement is the alternate case which can be seen from the Figure 3.6. It is clear from Figure 3.6, that for 50% selectivity the query takes least time to execute for 50% periodicity, which corresponds to the alternate data arrangement. Hence for any query involving single column in its predicate and the query selectivity is about 50% the optimal data arrangement is the alternate case.

For relations with a decent tuple width we can obtain a good window of opportunity, which can be exploited in many cases such as, given an operation to be performed on two relation. Interesting case is that in one of the relation the tuples are sorted according to the column appearing in the predicate of the query, while the other relation has the tuples unsorted and uniformly distributed (according to the same required column appearing in the query predicate) over the relation space. Provided the selectivity of the operation to be performed is not the extreme cases, then obviously the relation with uniform distribution will be operated much faster and lesser energy consumption. Another inference that can be drawn from these set of experiments is that if the tuples are sorted then case of Sort-Merge-Join consumes most energy and time among other join options.

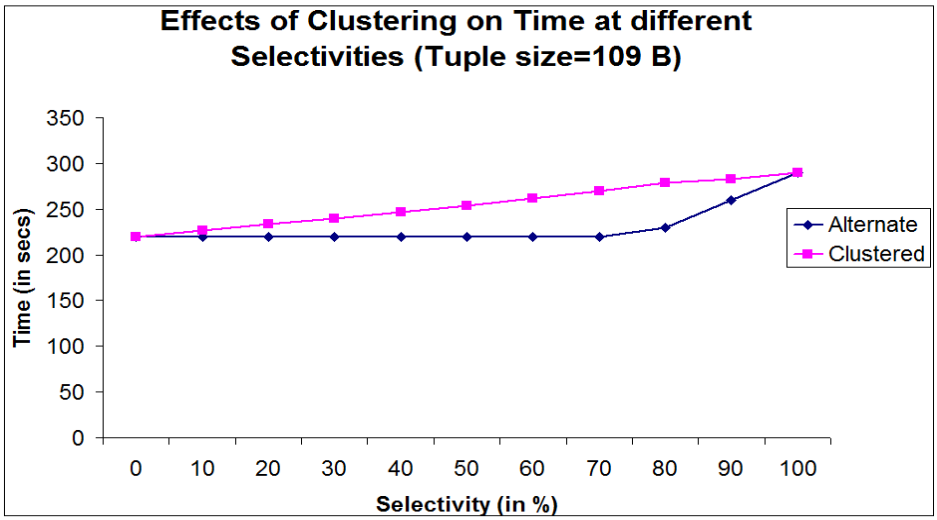

Figure 3.7: Effects of Clustering on Execution Time

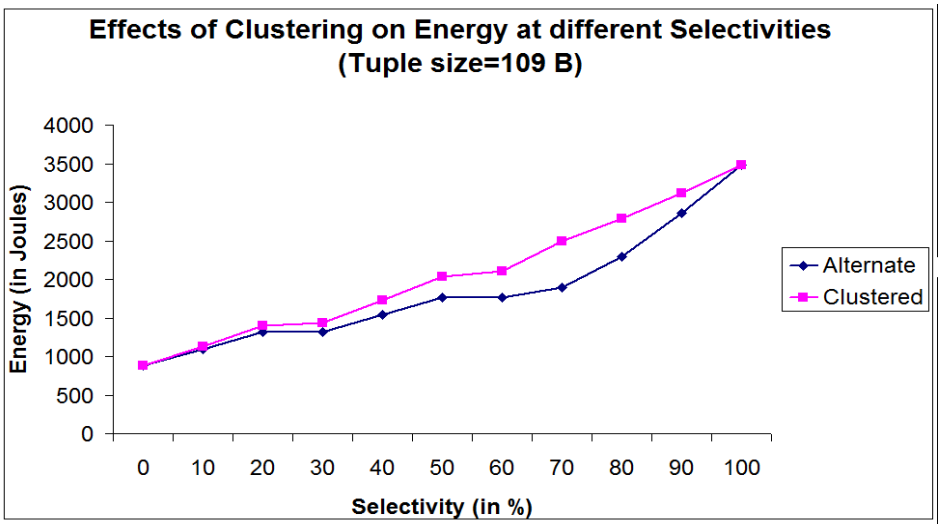

Figure 3.8: Effects of Clustering on Energy Consumption

Chapter 4

DESIGN STRATEGIES FOR PEAK POWER MODEL INTEGRATION

Traditional query optimizer selects a query execution plan that minimizes the estimated query execution time but power and energy considerations are not taken into account. It has been shown by many researches that time-efficient plans are not always power-efficient plans. [7] showed that tradeoffs exist between execution time and peak power consumption of query processing. So in this scenario, it is quite possible that a peak power-efficient plan gets discarded in favor of a time-efficient plan.

In order to make an optimizer capable of quantifying a query plan on peak power, we need to incorporate the models developed in [7] into a query optimizer.

4.1 Peak Power

Peak Power is the maximum power consumed over the entire duration of query execution. Its consumption is of relevance in server design, capacity planning, and prevention of overheating surges. Peak power models explored in [7] are pipeline-based.

4.2 Pipeline

A pipeline is referred to as the concurrent execution of a contiguous sequence of operators. [2] and [10] discussed the process of identifying the pipelines present in a plan tree. The important point to note here is that, unlike energy, peak power of a pipeline is independent of the data size; it is affected by the data rate.

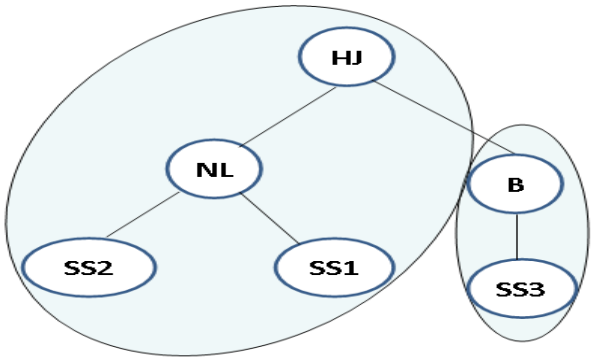

Figure 4.1: Pipeline Annotated Plan Example

Figure 4.1 depicts a pipeline annotated plan of a query involving join on three relations, that consists of two pipelines which are shown in ellipses. The nodes respectively represent the following:

HJ: Hash Join

NL: Nested Loop Join

B: Build

P: Probe

SS: Sequential Scan

4.3 Plan Selection Criteria

Choosing a plan on the basis of peak power depends on one of the following criteria:

- **Minimum Peak Power (MPP):** The execution strategy that will be chosen by the optimizer will have the minimum peak power consumption from among all the possible execution strategies.

  The MPP criterion is not always desirable in real world applications as the query response is also one of the major concerns in query execution. Interesting tradeoffs exist between query execution time and peak power consumption, which can be exploited to generate a plan that has reasonable peak power consumption with reasonable increase in its execution time.

- **Time Budgeted:** The execution strategy with minimum peak power from the set of plans whose query execution times are less than a user-specified threshold. SLA agreement can be used as the threshold. (SLA agreement for energy prediction models is explored in [8]).

- **Power Budgeted:** The execution strategy with peak power less than a specified threshold. The threshold can be decided by the designer of the system, or the user of the system, or can be set to a value which is equal to the peak power of the plan with the minimum execution time. We call such plan selection criterion as **Reasonable Peak Power (RPP)**.

4.4 Approaches to solve the cost-optimization problem

The cost metric used in the modern optimizer is the query execution time which is additive in nature. The various approaches which can help us solve this cost-optimization problem efficiently are as follows:

- **Dynamic Programming (DP):** The approach was proposed in the very first paper in this field [3]. It is a bottom-up approach to generate the best execution strategy by selecting the best option from various choices available at each level of the plan tree. Limitation of this approach is that the query optimization takes too long.

- **Transformation:** It is a top-down approach which builds an initial plan, then iteratively changes it to another, level-by-level using a set of rules in order to get the final plan. The obvious advantage of this approach is that it is much faster than standard DP, but doesn't guarantee an optimal solution.

- **Others:** There are some other approaches with the objective to make the process of optimization cheaper compromising on the quality of the solution. For example, Genetic Algorithm used in PostgreSQL database system.

Now the question is, "Can these approaches be used for Peak Power model integration as is, or need to be modified to serve our purpose?" In the next section, we shall see why the currently used standard DP approach is not suitable in this scenario. Then we shall propose new approaches to solve the problem under consideration.

4.5 Unsuitability of Dynamic Programming

Dynamic Programming is a design technique for solving problems defined by or formulated as recurrences with overlapping subproblems. DP exhibits optimal substructure property where an optimal solution can be constructed efficiently from optimal solution of its subproblems.

The models for Peak Power are pipeline-based which lead to the following structural problems in the application of DP here:

- **Lack of optimal substructure property:** Sometimes a pipeline might need to be replaced with a set of pipelines instead of a single one. For example, as shown in Figure 4.2, if a pipeline(left) is found to be expensive, then it can be replaced by two cheaper pipelines(right).

- **Incomplete pipelines:** Peak Power optimization works at the pipeline level rather than the operator level in the plan tree. If we calculate the peak power at a particular operator level of the incomplete pipeline, the overall peak power consumption of the pipeline

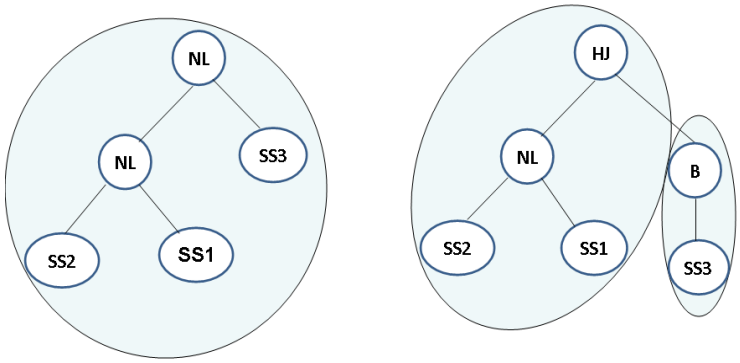

Figure 4.2: Lack of Optimal Substructure Property

might decrease as we climb up the pipeline. This prevents the calculation of peak power at operator level there by makes DP unsuitable to solve the problem under consideration.

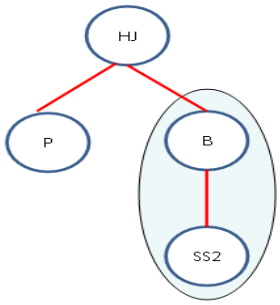

Figure 4.3: Incomplete Pipeline

In Figure 4.3, the pipeline at *HJ may be* incomplete and the pipeline at *B* is complete. Whether the pipeline at HJ is complete or not depends on the parent node of HJ, which is not known at this point. On the contrary, the pipeline at B is complete as B is a blocking operator. Hence decision of *pruning or rejection* of the pipeline can not be made for the pipeline with HJ operator.

Definition: An *Inductive Pipeline* of *n*+1 operators, PP_{n+1} is a pipeline that is created by adding an operator to a pipeline of *n* similar operators, PP_n. An inductive pipeline at an operator level is an incomplete pipeline.

Claim: Peak Power of PP_{n+1} will be *less than or equal* to that of PP_n.

Proof: Generic model for nested hash joins for peak power estimation [7]:

$$PP_{n+1} = K * PP_n + C_{n+1} + A_{n+1} * R_{n+2}$$

where,

C_{n+1} : parameter coefficient,

A_{n+1} : parameter coefficient,

R_{n+2} : output data rate of $(n+1)^{th}$ operator.

K reflects the back-pressure impact of the additional join on the upstream operators.

Following are the peak power models for nested hash joins for 3, 4 and 5 relations respectively:

$$PP_2 = -1.302 * 10^{-4} * R_2 + 45.384 + 5.556 * 10^{-4} * R_3$$
$$PP_3 = 0.71 * PP_2 - 26.214 + 8.326 * 10^{-4} * R_4$$
$$PP_4 = 0.71 * PP_3 + 2.28 + 9.28 * 10^{-4} * R_5$$

For worst case analysis, we can assume $\forall i, R_i = R_1$, the rate at which the actual data was scanned. This shall yield upper bounds of peak power for various pipelines. So the above equations now become:

$$PP_2 = 45.384 - 0.8 * 10^{-4} * R_1$$
$$PP_3 = 5.3 - 8.7 * 10^{-4} * R_1$$
$$PP_4 = 6.043 - 50.49 * 10^{-4} * R_1$$

For $R_1 \geq 10^5$, the relative ordering of upper bounds of peak power for the pipelines in consideration will be:

$PP_2 \geq PP_3 \geq ... \geq PP_n \geq PP_{n+1}$

This proves that as we move up in a incomplete pipeline its peak power consumption might

decrease. The practical values of R_1 are much higher (typically in MBps) than what was used in our proof.

Implications from the above proof:

- If an incomplete pipeline is pruned at a certain level (due to high power consumption), it might eventually lead to less power consumption at higher levels.
- This proves that standard dynamic programming approaches cannot be used.

4.6 Proposed Approach

We propose an approach that aims to generate RPP plan instead of MPP plan. The proposed approach is a modification to the standard dynamic programming. First, we look into the brute force or naive approach.

Naïve Approach

Input: Query
Output: Minimum Peak Power(MPP) plan

1 Enumerate all pipeline annotated plans
2 For each plan
  2.1 Compute the power of each pipeline involved
  2.2 Compute the peak power of the plan
3 Return the plan with minimum Peak Power

First, we look into the brute force or naive approach. This approach guarantees the generation of MPP plan at the cost of query optimization time. But,

- MPP plan generation requires *all* the plans to be enumerated
- An MPP plan need not necessarily be a time-efficient plan.

Join Type	**Peak Power** (in Watt)	**Time** (in seconds)
Nested Loop	124	1253
Hash	128	139
Merge	130	183

Table 4.1: Evaluation of Naive Approach

For instance, for a query with a join operation, the Naive approach enumerates all three join types, whose peak power consumption and execution times are shown in Table 4.1.

From Table 4.1, it is clear that the Naive approach outputs Nested Loop Join as MPP plan, but its execution time is nearly ten times that of Hash Join, which is the time-efficient plan in this case.

Naive approach is not the only approach that generates a peak power efficient plan, there can be other approaches with pruning heuristics that does the same job. Since we are not able to come-up with an algorithm that generates MPP plan that do not enumerate all possible plans, we propose an algorithm that generates a good approximation to MPP plan. Since this new approach generates a good approximation to MPP plan we call this as RPP plan, i.e., reasonably good plan with Reasonable Peak Power (RPP) consumption. Table 4.1 showed a problem that although the power consumption of an MPP plan is least but execution time is also the highest, where an RPP plan can be a better choice. We formulated a DP-like algorithm that generates an RPP plan. The algorithm is given below.

Threshold T is used here as a pruning heuristic. Appropriate values of T yield better performance of the algorithm. A higher value of T will lead to a lot of enumeration and no pruning while a very low value of it may lead to no plan getting selected at all. This threshold can be decided by the designer of the system, or the user of the system, or can be set to a value which is equal to the peak power of the plan with the minimum execution time. We will call this threshold **Pilot Peak Power**. The RPP algorithm proposed will thus produce a set of plans

RPP Based Approach

Input: Query, Threshold T
Output: Reasonable Peak Power (RPP) plan

1 Enumerate all possible options starting from the lowest level

2 At each level, find if complete pipelines exist

3 If such a pipeline P exists and Power(P) > T, then

  3.1 Prune subtree of P

  3.2 Prune currently enumerated plans which have P as subplan

4 Return the RPP plan from the set of budgeted plans

whose peak power consumption might be much less than that of time-efficient plan, as in case shown in Table 4.1.

In RPP approach, we follow *look behind model*, where we look behind by one level and check for the presence of complete pipelines. The incomplete pipelines are not rejected and are maintained as a potential option at every level. This is similar to that of maintaining *interesting orders* in case of execution time optimization in DP for SystemR [3], etc.

In the example shown in Figure 4.4, when we encountered the first *HJ*(from bottom of the plan tree), we looked behind by one level and found a complete pipeline at *B*. But as the peak power of this pipeline was found to be lesser than the threshold, *HJ* was further enumerated. At the next level, we found another complete pipeline at *B* whose peak power was greater than the threshold. Hence, all the plans with this pipeline as a subplan are pruned. This results set of plans at end of the algorithm shown in Figure 4.5.

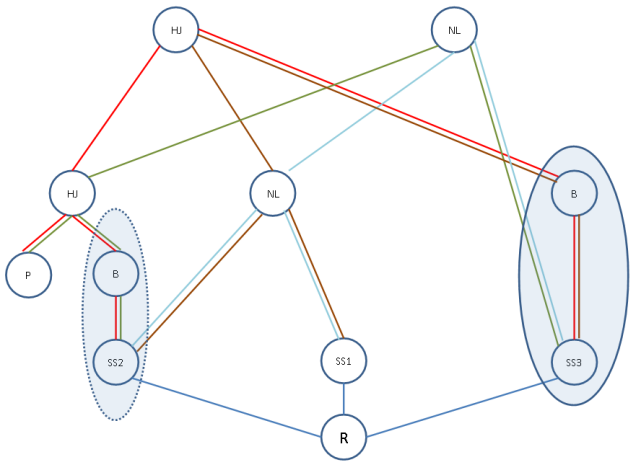

Figure 4.4: Example Run: RPP

Following assumptions are made for making the example run of the above mentioned approach simple to understand:

- A query with join on three relations
- Left deep joins are shown
- Only two join operators exist
- Only one scan operator exists

Limitations of RPP Based Approach

- Longer pipelines lead to more enumerations and less pruning
- RPP plans might result in higher execution times. For example, *Inductive Nested Loop Join Pipeline*.
- For a higher value of threshold, the algorithm might end up enumerating all possible plans

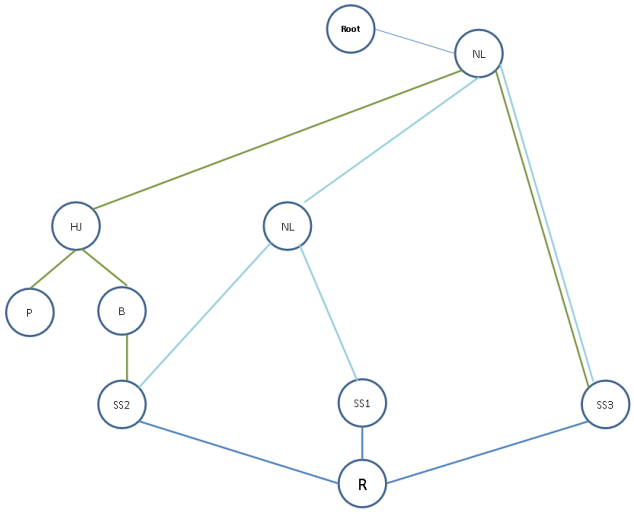

Figure 4.5: Example Run: RPP

To workaround the above stated limitations, we can use *cost(time) based pruning*. This results in lesser enumerations and also produces cost(time)-effective power budgeted plans.

Chapter 5

CONCLUSIONS

- The energy models developed for a database server so far, are incomplete. Hence, we have extended the feature set of these models.
- For a query involving a single column in its predicate, the optimal result arrangement is the *alternate* case.
- Similar results were experimentally shown to hold good for not only sequential scan but also other operators that have similar processing. For instance, nested loops, etc.
- Given a schema (provided it will be repeatedly queried), there exists a data arrangement which leads to savings in time taken and energy consumption.
- To the best of our knowledge, this is the first work that shows that careful arrangement of result tuples can improve both the metrics, i.e., energy consumed and time taken.
- Surprisingly there is no trade-off and practically we can achieve decrease in all the three metrics; viz. energy consumption, execution time and peak power consumption, by changing the arrangement of result tuples on the disk.
- RPP based approach does offer some advantage over the brute force approach w.r.t. query optimization time.

Bibliography

[1] R. Agrawal et al, The Claremont report on database research , *SIGMOD Record, 2008.*

[2] S. Chaudhuri, V. Narasayya and R. Ramamurthy, "Estimating progress of execution for SQL queries", *Proc. of SIGMOD Conf., 2004.*

[3] P. Griffiths Selinger, M. M. Astrahan, D. D. Chamberlin, R. A. Lorie and T. G. Price, "Access Path Selection in a Relational Database Management System", *SIGMOD, 1979.*

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

[5] J. Hamilton, "Where does Power Go in DCs and How to get it back?", *HotPower, 2008.*

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

[7] M. Kunjir, P. Birwa and J. Haritsa,"Peak Power Plays in database engines", *EDBT, 2012.*

[8] Willis Lang, Ramakrishnan Kandhan and J. M. Patel, "Rethinking Query Processing for Energy Efficiency: Slowing Down to Win the Race", *ICDE, 2011.*

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

[10] G. Luo, J. Naughton, C. Ellmann and M. Watzke, "Toward a progress indicator for database queries", *Proc. of SIGMOD Conf., 2004.*

[11] Meikel Poess and Raghunath Othayoth Nambiar, "Energy Cost, The Key Challenge of Today's Data Centers: A Power Consumption Analysis of TPC-C Results", *Proc. of PVLDB, 2008.*

[12] S. Rivoire, M.A. Shah, P. Ranganathan and C. Kozyrakis, "JouleSort: a balanced energy-efficiency benchmark", *Proc. of SIGMOD Conf., 2007.*

[13] Manuel Rodriguez-Martinez, Harold Valdivia, Jaime Seguel and Melvin Greer, "Estimating Power/Energy Consumption in Database Servers", *Procedia Computer Science Vol. 6: 112-117, 2011.*

[14] D. Tsirogiannis, S. Harizopoulos and M. A. Shah, "Analyzing the Energy Efficiency of a Database server", *Proc. of SIGMOD Conf., 2010.*

[15] Z. Xu, Y. Tu and X. Wang, "Exploring Power-Performance Tradeoffs in Database Systems", *Proc. of ICDE Conf., 2010.*

[16] "Internet's well-guarded secret: It's a power-guzzler & polluter", *Deccan Herald, Tuesday 23 October 2012*, http://www.deccanherald.com/content/280861/internets-well-guarded-secret-its.html

[17] Brand Electronics Digital Power Meters, http://www.brandelectronics.com/meters.html.

[18] "Drop Caches", http://linux-mm.org/Drop_Caches

[19] TPC-E, http://www.tpc.org/tpce/

[20] TPC-H, http://www.tpc.org/tpch/