EMPIRICAL COST MODELS FOR ESTIMATING POWER AND ENERGY CONSUMPTION IN DATABASE SERVERS

By

Harold Dwight Valdivia Garcia

A thesis submitted in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

in

COMPUTER ENGINEERING

UNIVERSITY OF PUERTO RICO
MAYAGÜEZ CAMPUS

December, 2010

Approved by:

Nayda G. Santiago Santiago, Ph.D Member, Graduate Committee	Date
Pedro Rivera Vega, Ph.D Member, Graduate Committee	Date
Manuel Rodríguez Martínez, Ph.D President, Graduate Committee	Date
Uroyoán R. Walker Ramos, Ph.D Representative of Graduate Studies	Date
Erick E. Aponte, Ph.D Chairperson of the Department	Date

Abstract of Dissertation Presented to the Graduate School
of the University of Puerto Rico in Partial Fulfillment of the
Requirements for the Degree of Master of Science

EMPIRICAL COST MODELS FOR ESTIMATING POWER AND ENERGY CONSUMPTION IN DATABASE SERVERS

By
Harold Dwight Valdivia Garcia
December 2010

Chair: Manuel Rodríguez Martínez
Major Department: Electrical and Computer Engineering

The explosive growth in the size of data centers, coupled with the widespread use of virtualization technology has brought power and energy consumption as major concerns for data center administrators. Provisioning decisions must take into consideration not only target application performance but also the power demands and total energy consumption incurred by the hardware and software to be deployed at the data center. Failure to do so will result in damaged equipment, power outages, and inefficient operation. Since database servers comprise one of the most popular and important server applications deployed in such facilities, it becomes necessary to have accurate cost models that can predict the power and energy demands that each database workloads will impose in the system. In this work we present an empirical methodology to estimate the power and energy cost of database operations. Our methodology uses multiple-linear regression to derive accurate cost models that depend only on readily available statistics such as selectivity factors, tuple size, numbers columns and relational cardinality. Moreover, our method does not need measurement of individual hardware components, but rather total power and energy consumption measured at a server. We have implemented our methodology, and ran experiments with several server configurations. Our experiments indicate that we can predict power and energy more accurately than alternative methods found in the literature.

Resumen de Disertación Presentado a Escuela Graduada
de la Universidad de Puerto Rico como requisito parcial de los
Requerimientos para el grado de Maestría en Ciencias

MODELOS DE COSTOS EMPÍRICOS PARA LA ESTIMACIÓN DEL CONSUMO DE POTENCIA Y ENERGIA EN SERVIDORES DE BASES DE DATOS

Por
Harold Dwight Valdivia Garcia
Diciembre 2010

Consejero: Manuel Rodríguez Martínez
Departamento: Ingeniería Eléctrica y Computadoras

El crecimiento vertiginoso de los centros de cómputo a gran escala, junto con el uso generalizado de tecnologías de virtualización, han hecho que el consumo de energía y potencia máxima esten entre las principales preocupaciones para los administradores de estos centros de cómputo a gran escala. Las decisiones de aprovisionamiento ahora deben tomar en consideración no solo el rendimiento de las aplicaciones, sino también la demanda de energia incurrida por el hardware y el software a ser utilizados en los centros de cómputo. Fallar en ello resultará en daños a los equipos, cortes de energía y operaciones ineficientes. Ya que los servidores de base de datos constituyen uno de los más populares e importantes servidores de aplicación en estas facilidades, es necesario tener modelos de costos más precisos que puedan predecir las demandas de energía y la potencia máxima que cada carga de trabajo impondrá en el sistema. En este trabajo nosotros presentamos una metodología empírica para estimar la potencia máxima y el costo de energía de las operaciones de un servidor de base de datos. Nuestra metodología utiliza regresión lineal multiple para derivar modelos de costos que dependan solo de estadísticas disponibles en la base de datos, tales como la selectividad del "query", el tamaño promedio de los registros, el número de columnas y la cardinalidad de una relación. Además nuestra metodología no necesita medir individualmente los componentes hardware de un computador. En vez de ello,

se mide la potencia máxima y el consumo de energía del servidor. Hemos implementado nuestra metodología y corrido experimentados con diferentes configuraciones de servidores. Nuestros experimentos indican que podemos predecir la potencia máxima y la energía con mayor precisión que otros métodos alternativos encontrados en la literatura.

A mi queridísima madre, por ser mi luz y mi esperanza.

ACKNOWLEDGMENTS

First I would like to thank my advisor and mentor, Dr. Manuel, for his constant support and guidance during these two last years. Working with him has been a great research experience. I would also like to thank my graduate committee, Dr. Nayda Santiago and Dr. Pedro I. Rivera Vega, for their interest in this research topic.

I would also like to thank all the professors and staff at the University of Puerto Rico, Mayagüez Campus that provided me with their knowledge and support during these last years.

Thanks to all my friends at ADMG laboratory, for their support and help during these years. Thanks to my research partners Edward Betancourt and Jose Rivera Santuche, the implementation of this thesis work would not be possible without their help. Also, thanks to my mentor and friend Eliana Valenzuela for her help and counseling.

Thanks to my mother, Naty Garcia for her love, support and prayers. Without her, I would be lost. Thanks to my bothers Janhsen, Johann and John Jack and my niece Lucianita for their love and understanding as well. Thanks to my beloved Paola for sharing with me her life.

Finally, I would like to thank my friends of my support groups for sharing with me their strength, hope since I arrived in Puerto Rico.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Internet-scale computing environments such as those envisioned by the cloud computing paradigm contain thousands of host machines with different hardware/software characteristics. These pools of machines are often referred to as *clouds*, and customers can lease time to run jobs on them. This enables customers to outsource the entire IT infrastructure they need to run their business from a highly specialized and trustworthy IT company that runs the cloud. In some cases, most of these machines will be housed at a single data center [1] and be physically contained within a small area, such as a shipping container [1, 2]. Some hosts available for data processing will be real machines, as we know them, while others will be virtual hosts running on a virtualization platform such as VMware, Xen, or KVM.

Cloud computing companies need large data centers to accommodate the needs of their growing user base. Hence, electrical energy consumption in data centers is rapidly becoming a major cost factor in the operation of these facilities. For this reason, research efforts in new, energy-efficient hardware/software systems are necessary to curb server energy consumption.

Database management systems are common server applications deployed at data centers. For many years, developers of DBMS have focused their optimization efforts into minimizing either the response time or the resource usage time required to run queries. From this perspective, the priority is to maximize the performance of the database applications by minimizing time, and thus increasing system throughput. But, in cloud environment where the electrical energy consumption has becamo an important concern, the optimization also must build efficient query plans in terms of the power demands. Failure to do so would result in high cost of energy, environmental problems, overloading servers and racks to the point at which circuit breakers and other protection equipment would shutdown electric energy to protect the equipment.

1.1 Problem Statement

In short, our goal is to develop simple and accurate cost models that can be used to predict the power and energy costs that queries will have on a physical server. The DBMS that evaluates these queries can be run directly on the physical machine, or it can be run on a VM. Having these cost models enable the following technological features at modern data centers:

- **Automated DBMS Provisioning** - A software controller can place the DBMS needed to run a given workload on a real or virtual machine with enough electric power to sustain the utilization imposed by the query workload. This is particularly important in multi-tenant environments running single-site database servers.
- **Energy-aware Optimizer** - A query optimizer that understands the energy, peak power, and average power of the query plans can pick not only energy-efficient plans but also pick the appropriate server machines on which to run each query operator.

Formally, consider a workload of n queries $Q_1, Q_2, ..., Q_n$ to be run on a machine M. For each query Q_i we need to find cost functions E_i, P_i, $\bar{P}_i$ where:

- E_i is the energy consumed to run the whole query.
- P_i is the peak power reached while running the query.
- $\bar{P}_i$ is the average power reached while running the query.

The formulation is similar if we are running on a parallel DBMS, distributed DBMS or MapReduce environment, where we have a set of k machines $M_1, M_2, ..., M_k$. In such case, we need to find cost functions E_i^j, P_i^j, and $\bar{P}_i^j$ to get the energy, peak power, and average power for running operators for query Q_i on machine M_j. In this work, however, we focus on the cost models for deployments in which there is either: a) one single-site DBMS to run all queries, or b) several single-site DBMSs cooperating to run queries over horizontally partitioned data.

1.2 Proposed Solution

This research is focused on how to model the energy and power consumption of the database servers. We shall illustrate the software infrastructure needed to run a workload on a multi database server environment. We shall also describe the tools to collect the peak power, average power and energy consumption of the nodes. Our methodology characterizes the electrical consumption behavior of different queries submitted to the system. Three types of queries are considered in our methodology: selection, projection and joins. For each of these types of queries we obtain cost models to estimate the power and energy costs for queries, by means of regression techniques. We analyze the effect of different hardware configurations in the power consumption of queries, and how virtualization affects the query execution.

1.3 Contributions

Our research makes the following original contributions:

- Illustrates the challenges in provisioning database server machines and placing query operators when power/energy are a major cost factor.
- Introduces a methodology to derive cost models that predict power and energy cost of queries. This methodology can be incorporated into query optimization frameworks and tools used to do energy provisioning or power capping on environments that include strict power budgets to the physical servers.
- Presents a performance study, based on TPC-H, showing that our framework can accurately predict the power/energy costs of query operators. It also shows that it is more accurate when compared with other methods based on response time models, that estimated power as a cumulative metric.

1.4 Outline

The rest of the thesis is organized as follows. Chapter 2 gives the reader necessary background information. Chapter 3 presents our methodology for power and energy estimation. Chapter 4 contains experiments that validate our framework. Finally, conclusions are presented in Chapter 5.

CHAPTER 2

LITERATURE REVIEW AND BACKGROUND

In this section we present background necessary to understand the thesis, and then formally state the problem we aim to solve.

2.1 Electric Power and Electric Energy

Electric energy (E) can be computed as the product between average electric power (P) and equipment use time (T):

$$E = P \times T$$

Electric energy is measured in Joules (J) and denotes capacity to perform work. Electric power is defined as the rate at which energy is consumed or produced [3], and is measured in Watts (W). Thus, electric power can be used to denote how much energy a computer is consuming at a given point in time t. The measurement units Joules and Watts are related: one Watt is equivalent to one Joule consumed per second, 1W = 1J/s.

It is important to grasp the relation and magnitude of these metrics. Electric power plants are rated in terms of their power capacity, which is a measure of how much power they can generate. For example, an oil-fired thermoelectric plant might have a capacity of 250 megawatts (MW), while a nuclear plant

might have a capacity of 1000 MW. A commercial building can consume a few megawatts, while a small city requires a few hundred megawatts. In 2004, New York City officials forecasted that consumption of electricity in the City would reach about 10,000 megawatts by 2008 [4]. Electric utilities typically charge for electricity in terms of kilowatt-hours (kWh), which is the amount of energy of a steady power of 1 kilowatt running for 1 hour. For example, a 250 W computer running at peak power for 20 hrs would consume 5000 Wh, or 5 kWh.

Notice that understanding the required computer electric power is necessary for provisioning electricity to the data center, since enough electric power must be available to feed the machines deployed at the facility. Peak power refers to the maximal power drawn by a computer at any time t during its operation. In contrast, average power refers to the mean power consumption over a time interval $[t_0, t_1]$. Meanwhile, understanding of the electric energy consumed by the machines is necessary to estimate the energy expenses to be incurred by the facility during a given time period.

2.2 Power Consumption in Data Centers

Internet service companies are building new, colossal datacenter based on cheap commodity computers, thanks to falling server prices. However, there are new technical, environmental and economics challenges that make it difficult to create and deploy data centers. We review these challenges next.

Since the 90s, computer vendors are building more expensive and powerful servers. The performance and the performance-per-server price have grown with each new server generation, however the performance per watt has stayed constant. This scenario suggests that the cost for powering these powerful machines is increasing quickly and linearly [5]. Nowdays, the power consumption is becoming the principal factor in the Total Cost ownership. The EPA reports that

data centers consumed 1.5% of the USA energy in 2006 [6]. McKinsey reports that the world's servers produce 0.2 % of all CO2 emissions in the world [1]. Heat is another problem that impacts negatively the operation of a data center. Traditionally, air chiller systems consume 1W for cooling operations for every 1W used to power servers [6]. Additionally, the space requirement of such systems limits the power density. To deal with these issues, data centers are turning to new cooling technologies like water based cooling systems, external cold air, and smart cooling with sensors, among many others. Building and designing greener and more energy efficient computers may cut the CO2 emissions generated by the IT sector [7].

Companies like Google, Amazon, Microsoft with data centers of thousands and more systems incur on enormous cost of infrastructure and time consuming task, for example, shipping and packaging computing equipments, powering, networking, housing in a security location, cooling systems, fixing and replacing all these systems.

To alleviate these challenges, the work in [2] proposes to use a shipping container as building block of new generations of data centers. This kind of modules have many advantages: they are weatherproof and inexpensive in comparison to conventional large rack rooms. A container is designed to hold delicate equipments and it can be shipped over land, air or even sea. Rackable System, Verari and Sun Microsystem sell these units as fully operational macro-modules consisting of thousands of blade servers and integrated with power, networking and cooling system. For example, the Rackable ICE Cube container has a capacity of 22400 cores and 11.8 Petabytes of storage. Although deploying data centers can still be expensive, by using shipping containers the process is highly simplified. These containers are cheap to move to regions with affordable electricity, reliable networking or some tax benefits. To increase storage and computing power on

data centers due a growing user demand requires to attach more shipping containers and feed them power, water and networking. Another advantage is that data centers no longer need servers room with raised floors, nor Computer Room Air Conditioner (CRAC consumes aproximately 50% of the total floor space) instead containers employ water-based chillers to remove the heat. As result, such configuration supports high power density and data centers can be fully populated.

2.3 Data Center Organization

Traditionally, data centers have been built on special purpose buildings, featuring raised floors and very large air conditioning units to cool down the servers. The work in [1] reports that data centers consuming 20 MW of power are commonplace, and that new data centers consuming 200 MW are expected in the near future. The work in [1, 8] indicates that for every Watt spent powering a server, at least another half a Watt is needed to cool it down. This results in a situation in which the cost of powering and cooling a server over its useful lifetime exceeds the cost of purchasing it [9].

Recently, however, a new trend has emerged for building data centers in a modular fashion. Racks full of servers are packed into a shipping container, and cooling is done with chilled water. The shipping containers are placed in a warehouse-like facility where electricity, water and Internet connectivity are supplied to each container. This scheme simplifies data center deployment, and reduces the costs associated with powering and cooling the servers [2].

Still, provisioning power for a data center becomes a complicated and delicate matter. Figure 2–1 shows the architecture of a data center power distribution infrastructure, as discussed in [9]. A power source, backed by an emergency generator, feeds the data center. The electricity is delivered to a pair of redundant

Uninterruptible Power Supplies (UPS), which in turn feed one or more Power Distribution Units (PDU). Notice that each PDU has a static transfer switch (STS), used to get power from any one of the available UPS. The PDU provides electricity to an electric panel from which electric circuits emerge to feed the racks containing the physical servers.

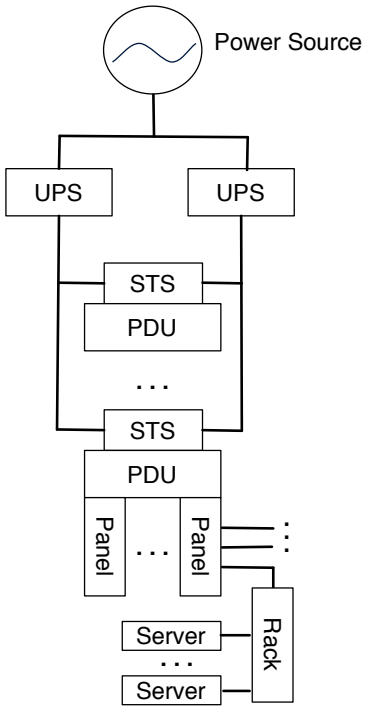


Figure 2–1: Power distribution architecture.

2.4 Server Power Management

Servers can reach their peak power when their utilization level is high. But, idle servers also consume power. In fact, an idle server can consume more than 50% of its peak power [7]. The nameplate power is the peak power specified by the hardware vendor. In practice, however, the actual peak power is much less than the nameplate specification [9]. Another key observation is that most servers operate between 10% and 50% their utilization, hence, not all machines

are operating at peak power at the same time. These observations can be used by data center architects to accommodate more machines in a panel than what can be powered if all of them were working at peak power at the same time. This scheme helps reduce costs by provisioning for the amount of power that will be likely consumed instead of provisioning for a worst case scenario.

The tradeoff is that power caps now need to be assigned to servers and racks to ensure that machines do not overload the electric system during an unexpected surge in utilization [10]. Each server gets assigned a power budget that indicates how high can its power reach at any time instant t. If some server A needs more power, some other server needs to relinquish a portion of its power budget. If no machine is found to donate power and the server exceeds its budget it will be powered down or downgraded to a lesser performance state, using techniques such as CPU throttling. Additionally, CPU throttling can be employed when the machine stays at peak power for too long since it will overheat and malfunction. Thus, when a virtual machine is provisioned, care must be taken to place it on a physical server that has both sufficient computational capacity to meet the performance demands and enough power left in its budget to accommodate the increase in power due to higher utilization.

2.5 Virtualization and Clouds

Virtualization permits the consolidation of idle servers by allocating multiple virtual servers into one physical server. This scheme simplifies data center administration since there is less equipment to manage. It also reduces energy costs as there are less servers consuming energy, and less cooling is required to keep the proper temperature. Virtual machines are managed by a software module known as the hypervisor.

Virtualization enables a type of cloud infrastructure known as infrastructure as a service (IaaS). In this scheme, multiple virtual machines are offered to end-users as replacements to physical machines. This is the model used by Amazon EC2, Eucalyptus, and Microsoft Azure. Users get charged by the amount of time they use the virtual machines. Users can customize the virtual machines with third party applications as needed. Typically, there is a cloud controller node that manages the cloud. Each physical server that can host virtual machines is controlled by a node controller module. This node controller takes care of starting, modifying and destroying virtual machines as needed. The cloud controller takes care of finding the proper physical server on which to provision a virtual machine. This provisioning decision takes into consideration the characteristics of the machines requested by the user, including features such number of virtual CPUs, memory size, disk size, and throughput.

2.6 Related Work

Statistical methods have been used to estimate several DBMS parameters, particularly selectivity factors [11–13]. But none of these methods considers peak power or energy, and hence, our work makes a contribution and complements them. The work in [8] proposed the use of multiple-linear regression to estimate peak power in terms of CPU, memory, disk, and network utilization. However, this work focused on the overall utilization of the machine, without concern about the specific workload being run on it. Therefore, it is more suitable for systems aiming at curbing power by using throttling techniques. In contrast, our method provides detailed cost estimates for individual queries, and can be used in query optimizers, or for DBMS provisioning decisions.

The work in [10] used the regression methods from [8] to develop a multi-level control system to manage the power levels of blade servers, blade enclosures, racks, and zones in the data center, with the goal of keeping all components within their power budgets. Similarly, the work in [9] used the methods in [8] to analyze power costs at a large data center, and provision power based on likely consumption instead of worst-case consumption. These examples show that regression methods, such as ours, are a viable mechanism to model power and energy costs in servers for data centers.

The need for energy-aware database systems was presented in [14], which discussed several challenges to develop energy-aware storage, indexing, optimization, and query processing methods. The authors in [6] studied several approaches to make a DBMS energy-efficient. These include energy-aware optimizers, resource consolidation, better storage managers, and perhaps trading-off performance in favor of more energy-efficient hardware and software environments. The work in [15] introduced various techniques to help curb the power consumed by a DBMS. The first technique, named Processor Voltage/Frequency Control (PVC) allows a controller in the server running the DBMS to reduce/increase the CPU voltage or frequency in response to changes in the query workload. Recall that CPU performance is related with the CPU frequency, which is also related with electrical power (higher frequency usually means higher power). This technique can be used to keep the CPU at an energy level that is consistent with the performance required by the workload at a given point in time. The second technique, Improved Query Energy-efficiency by Introducing Explicit Delays (QED), advocates for the grouping of queries with similar access paths. This enables results to be reused, in an effort to reduce wasted energy and computer resources incurred in recomputing the same operators in queries submitted simultaneously. Our method complements all these efforts by providing a way to find accurate

models for power that can guide the optimizers, storage managers, PVC controllers, and QED controllers.

Our work is most closely related with the research described in [16]. The authors of this work use a statistical method to derive the cost models for power in a DBMS, and modify the PostgreSQL optimizer with such models. However, these models are based on the optimization models used by PostgreSQL, which are better suited to make relative comparisons between resource usage of plans rather than capturing accurate cost. Also, their work uses the estimated power and estimated resource usage time from the optimizer to calculate the energy consumption, and they threat power as a cumulative quantity. Our work differs but complements this effort because our methods are able to accurately estimate both power and energy, with relatively small errors. As we saw in the experimental section, our methods show behavior closer to the real measured values compared with models borrowed from the time estimates done by a System-R optimizer.

The work in [17] studied energy efficiency in single-site database servers, and concluded that optimizing for time results in optimization for energy-efficiency. But, this might be the case if the server is not housed in a power-capped environment. If that is the case, then the optimizer must be aware of those caps, or it will generate plans that will surpass the power budgets of machines, forcing the controllers that manage the physical machines to reduce their CPU frequency or perform a shutdown. The same situation applies to parallel and distributed environments.

CHAPTER 3

ESTIMATING POWER AND ENERGY

We started our efforts to develop a methodology for estimating the power and energy in a DBMS with three major design goals. First, we wanted our models to be as accurate as possible, reflecting the actual power/energy costs rather than providing a relative measure to compare how the physical machine is stressed by alternative query plans [16]. Second, we wanted simple methods to collect the statistics necessary to build the models. On a large data center, it is impractical to open up servers and take measurements on individual hardware components as done in [18]. Rather, we wanted to use the measurements already taken by special-purpose sensors that come integrated with commercial server machines. Alternatively, we would resort to collect power/energy measurements by attaching a power meter to the server. Finally, we wanted the system to be adaptive in the sense that the models could be easily adjusted as needed, based on up-to-date measurements of the system. In this regards, we wanted to avoid hardwired formulas and constants, and instead, derive the cost models from actual measurements and statistical methods.

3.1 Monitoring Architecture

Figure 3–1 depicts the architecture of the system we designed to collect measurements from our servers and derive the cost models. The physical server can be located inside a rack, or housed in a stand-alone tower. Power measurements can be obtained from two sources. The first alternative is to collect power and energy measurement from sensors placed in an internal server management card. This alternative suits better in rack-based server deployments. Examples of these cards include the Dell iDrac 6 card, and the HP iLo system. The second alternative is to have an external power meter (power analyzer) collect power and energy consumption measurements. This alternative fits better with servers that are enclosed in a tower. In this case, the power source is connected to the power meter, and the server is plugged to the meter.

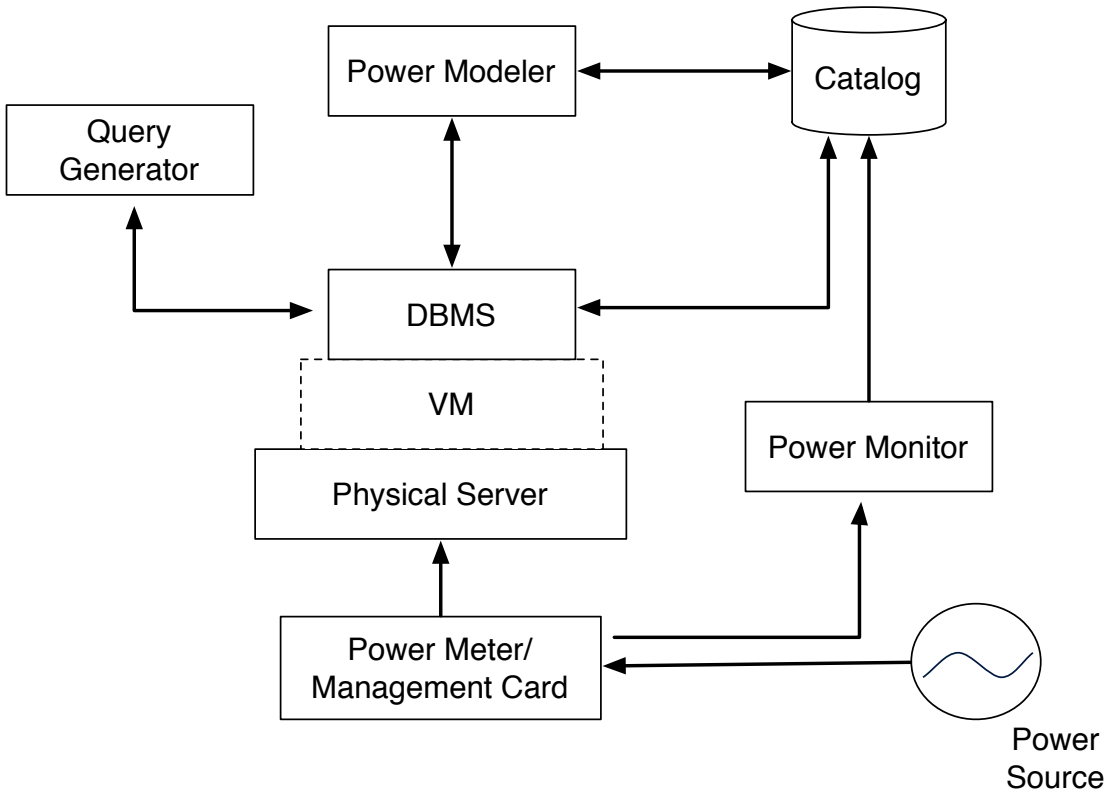


Figure 3–1: Monitoring architecture.

3.1.1 Power Monitor

Inside the physical server there is a power monitor application that periodically collects electrical measurements from either the power meter or internal sensors in the management card. In our case, we focus on collecting Watts (power) and kilowatt-hours (energy) consumed by the server. These measurements are taken every second, and then stored in a catalog associated with the DBMS for which power/energy models must be derived. This DBMS can be run directly on the physical server, or atop a VM hosted by the physical server.

3.1.2 Query Generator

The query generator is used to stress the system based on a given workload W. This workload consists of a set of tables and queries to be submitted to the DBMS at a specific rate. The query generator starts threads of execution to submit the queries to the participating nodes (via JDBC or ODBC), and receives all the results

There are three types of queries generated by the query generator: selection, projection and join. For the selection queries this module submits queries $\sigma_p(A)$ with different SFs. We provided hardwired SFs to the generator in our experiments, however these values could be provided by the catalog in a fully implemented data management system. The projection queries are generated varying the number of projected columns.

The query generator must be run from a machine different to the server as it will cause load that might alter the power/energy measurements, decreasing accuracy.

3.1.3 Catalog

The catalog stores information retrieved from two sources: the power monitor and the DBMS servers. The catalog holds the following data for each executed query:

- Electrical measurements: the power, peak power and energy measurement taken at each server during the evaluation of the queries.
- Metadata and statistics for all tables involved in the queries: the table size, number of blocks, cardinality, number of columns, tuples size, column size.
- Query workload and statistics: the query type and SQL specification of the queries sent to the DBMS, along with start and end time for each query, selectivity factors, number of projected columns.
- Servers configuration: The number of DBMS servers that participate in solving the queries.

The information above serves to characterize the expected working configuration and conditions for the system, once it is put in production.

3.1.4 Power Modeler

The power modeler component can be run inside the server, or at some other nearby machine. The power modeler receives input data from the catalog and use it to derive cost models that predict power and energy consumption of future queries. These models become metadata store in the catalog.

3.2 Cost Model Derivation Methods

Our cost models are empirical since they are derived from observations taken from the system. We employ the multiple-linear regression methodology for this purpose.

3.2.1 Multiple-Linear Regression

Using multiple-linear regression [19], we postulate that the power/energy consumed by the server can be computed by adding a linear combination of independent terms. For example, consider a table $R(A, B, C)$, and a selection query $Q = \pi_{A,B}(R)$. We might propose a cost model that relates the peak power P resulting from Q with the relation cardinality, $|R|$, and the number of columns in R, $< R >$. We can express this model as:

$$P = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \epsilon$$

with $x_1 = |R|$ and $x_2 =< R >$. In this methodology, each variable x_i is called a predictor or regressor, and its value represents some quantity that shall have influence on the dependent variable, in this case the power P. Similarly, each constant β_i is called a regression coefficient, with β_0 accounting for the case in which all predictors are zero. In the case of power, the constant β_0 is the power of the idle machine during query processing. The value ϵ is called the error term, and captures the deviation of the model from the real observed value.

In general, we can have have k predictors for the peak power of a query, giving the following formula:

$$P = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_k x_k + \epsilon$$

The energy (E) consumed while running the query can be estimated in similar fashion, so we focus our discussion on power to simplify matters.

The predictors in the model become parameters associated with the workload to be run with the DBMS. These include the relation cardinality, number of columns, tuple size, and so on. The predictors that we consider in our methodology are summarized in Table 3–1.

Statistic	Meaning
$\|R\|$	Cardinality of table R
$< R >$	Number of columns in R
$L(R)$	Average tuple length in R
$W(A)$	Average length for column A in R
$SF_P(R)$	Selectivity factor of predicate P applied to R
$PF(R)$	Fraction of columns retrieved from R
S	Number of servers used

Table 3–1: Parameters used as regressors (or factors) to estimate power or energy consumption in a query

To make the cost formula concrete and usable, one must find the values of the regression coefficients. This is done by performing a series of observations in which the power and predictor values are collected. Thus, each observation becomes a run of a query on the DBMS. During an observation the power gets recored by the measurement equipment. Likewise, the predictor values are read from the catalog. Multiple observations must be taken, each one representing a combination of predictors and the power that results. This leads to a system of linear equations of the form:

$$P_1 = \beta_0 + \beta_1 x_{11} + \beta_2 x_{12} + ... + \beta_k x_{1k} + \epsilon_1$$

$$P_2 = \beta_0 + \beta_1 x_{21} + \beta_2 x_{22} + ... + \beta_k x_{2k} + \epsilon_2$$

$$...$$

$$P_n = \beta_0 + \beta_1 x_{n1} + \beta_2 x_{n2} + ... + \beta_k x_{nk} + \epsilon_n$$

Each x_{ij} represents the value of predictor x_j during observation i. This system can be represented in matrix notation as:

$$\boldsymbol{P} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}$$

The regression coefficients can then be determined by the method of least squares, minimizing the error term ϵ. There are statistical packages such as Minitab and R that can do this computation.

It is important to point out that the formulas in the model need to be checked for accuracy and statistical significance with ANOVA tests. Specifically, one must check that the errors in the model follows a normal distribution. Also, the variance of results in repetitions with specific combinations of predictors should be constant (or near constant).

3.3 Model Calibration and Maintenance

We now provide a series of steps that can be used to calibrate the energy-related cost models for a DBMS:

1. Determine the set of tables and queries that better characterize your workload.
2. Set up your server(s), plus a client machine(s) with the query generator, as shown in Figure 3–1.
3. Run each query, performing repetitions in random order, and using a rate for query submission similar to the expected rate in the production setting.
4. Once all query repetitions are done, extract all power and energy measurements. Do the same for the DBMS metadata used for precitors (factors).
5. For each query type (i.e., selection, projection, or join), use the power modeler to obtain the cost model, and test if the model is correct. The modeler algorithm is described bellow.
6. During production operations, collect all measured values, and re-calibrate the models with the latest n observations.

MODEL-BUILDER ($ElectricalMeasurement$, $Parameters$)

```
1  Predictors_0 ← ∅
2  for p ∈ Parameters
3        Predictors_0 ← Predictors_0 ∪ functionalRelationship(p, ElectricalMeasurement)
4
5  Model ← regression(Predictors_0, ElectricalMeasurement)
6  Model ← revomeNonRelevantPredictors(Model)
7  Predictors_M ← getModelPredictors(Model)
8
9  for i ← 2 to |Predictors_0|
10       for S ⊆ Predictors_0 && |S| = i
11             Model' ← regression(Predictors_M ∪ S, ElectricalMeasurement)
12             Model' ← removeNonRelevantPredictors(Model')
13             if Model' > Model
14                   Model ← Model'
15                   Predictors_M ← getModelPredictors(Model')
16
17 return Model
```

Figure 3–2: Power modeler algorithm.

3.3.1 Power Modeler Algorithm

The algorithm for building the power and energy models iteratively is illustrated in Figure 3–2.

In the first block of code from line 1 to 3 the initial set of predictors is obtained. Instead of using prior assumptions about the functional relationship between the power/energy and the parameters, we use a scatterplot matrix graph to seek hints of the nature of these relationships. Based on this display (plot) the procedure $functionalRelationship$ returns one of the functions shown in the Figure 3–3.

Once the initial set of predictors is found the next step is to build the statistical model by construction. In the second block of code from line 5 to 7 we derive an initial model where the non relevant predictors were removed. The procedure $removeNonRelevantPredictors$ removes the less significant predictors of a model

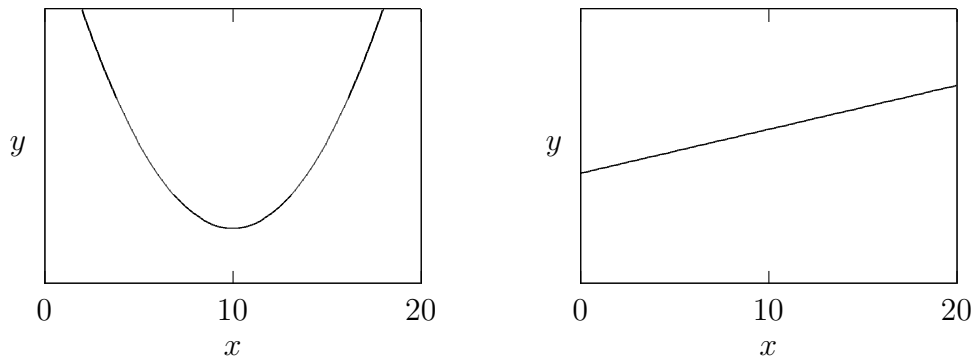
and returns a new derivation. It says that a predictor does not have significant effect in the response variable if its p-value is greater than $\alpha = 0.05$.

In the last block of code, all possible combinations of predictors are tested in the hope of increasing the quality of the model. In lines 11 to 12, we build, for each combination of predictors S, a new model by adding S to $Predictors_M$ and removing all possible non relevant predictors caused by this adding. Then $Model'$ is chosen instead of the current model ($Model$) only if:

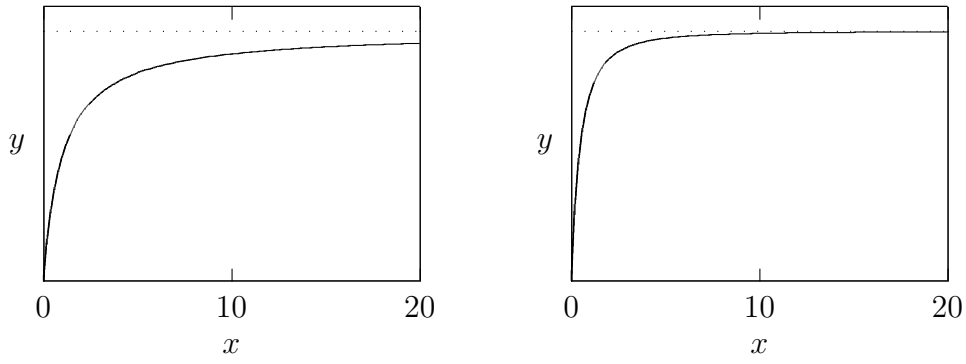
- The ANOVA assumptions are met (normality, independence, constant variance)
- All of its predictors are relevant.
- Its *adjusted determination coefficient*, $adj\text{-}R^2$, is greater than the $adj\text{-}R^2$ of the current model. The $adj\text{-}R^2$ is suggested in [19] as a good comparative statistic for evaluating regression models.

3.3.2 Summary

In this section, we have presented the architecture and methodologies necessary for collecting statistics related to the power/energy consumption of queries. In the next section, we present a set of experiments we carried out using these tools. We discuss the cost models that we postulate for various relational operators, and show how they fared in predicting power/energy.



(a) Quadratic (b) Linear



(c) $f(x) = a/x$ (d) $f(x) = a/x^2$

Figure 3–3: Functional Relationships.

CHAPTER 4

EXPERIMENTS FOR MODEL DERIVATION

We designed a set of experiments to determine if our methodology can help derive accurate cost models for energy-related metrics. Before discussing the results, we present the experimental environment and workloads used in the tests. Then, we move to a discussing in which, for several type of relational queries, we postulate and validate the cost model for power and energy. We found the results for peak and average power to be almost identical, so we will only present peak power results to the sake of brevity.

4.1 Experimental Environment

We carried out all of our experiments using the facilities of the Advanced Database Management Lab (ADM Lab) at the University of Puerto Rico, Mayagüez. We now describe these facilities, as well as the data sets and workloads used in conjunction with them.

4.1.1 Hardware

We used six server machines for running all our experiments. Two of these machines were Dell Precision Workstations 380 with one 2.8 GHz Pentium D dual core processor, 1 GB of 667MHz DDR2 RAM, and one 160GB, 7200 RPM Western

Digital Caviar hard disk. The remaining four machines were custom-built PCs with one 2.3 GHz AMD Phenom X4 quad core CPU, 8 GB of 800 MHz DDR2 RAM, and two 500 GB, 7200 RPM, SATA-300 Western Digital Caviar Blue hard disks. In each of these four machines, the disks were configured as a RAID-0 storage device. The nameplate power ratings for each type of server machine are presented in Tables 4–1 and 4–2.

	CPU	RAM	Disk	Total
Power	95W	4 x 9 W = 36 W	7.5 W	138.5 W

Table 4–1: Nameplate power ratings for Dell Precision Workstation 380.

	CPU	RAM	Disk	Total
Power	95W	2 x 9W = 18 W	8.77 W	130.54 W

Table 4–2: Nameplate power ratings for our custom-built PC.

Power measurements were taken on each server machine by means of a 120V Watts Up? Pro power meter. We set up the device to log power and kilowatt-hours measurements with a granularity of one second.

4.1.2 Software

We implemented a system based on our methodology as a set of programs and scripts written in Java, C, R, and Minitab. All our machines ran Ubuntu 10.04 LTS Server Edition. We used PostgreSQL 8.3 as the DBMS for experimentation. The query generator program was written as a Java client application that submits queries to PostgreSQL via JDBC. The power monitor program was written as a C daemon, which reads the logs from the power meters as a CSV file. The power modeler ran as an interactive session with Minitab, although it can be easily converted to script in R. We expect that both configurations will be used, as

database architects will likely use interactive mode to experiment with their applications, and later move to an R script to automate the process in a production environment.

4.1.3 Workload

To lead our experiments we used TPC-H, which is a benchmark widely used by the database research community. This benchmark is aimed to examine Decision Support Systems with large volumes of data. It has a set of complex queries that exercise the backend component of the system under test. We modified the queries in the benchmark to suit our needs. The TPC-H generates data for eight relations at different scales. We generated a database of 5GB and another database of 20GB. We ran all queries on both data sets to have observations in both settings, and collect results from small and larger data sets.

The table 4–3 summarizes the TPC-H schema and its features, as used in our experiments. The cardinality column shows the number of tuples generated at a $scale = 1$GB (i.e., the whole database takes up 1GB of space). The tuple length of the tables were obtained from the DBMS.

Table Name	Number of columns	Cardinality (in rows)	Tuple length (in bytes)	Table size (in MB)
SUPPLIER	7	$10,000$	159	2
PART	9	$200,000$	155	30
PARTSUPP	5	$800,000$	145	110
CUSTOMER	8	$150,000$	160	26
ORDERS	9	$1,500,000$	109	149
LINEITEM	16	$6,001,215$	125	641
NATION	4	25	128	< 1
REGION	3	5	124	< 1

Table 4–3: TPC-H Schema.

4.2 Comparison System

To validate our ideas we compare our regression based models with the models presented in [16], which are based on the System-R style cost models found in PostgreSQL. In these models, power is estimated as the sum of the power drawn by the CPU, and disk during query processing, treating power as a cumulative quantity. To ease our discussion, we shall refer to the energy-related cost models derived from System-R as Method A, and refer to our models as Method B.

4.3 Models for Peak Power of Selection Queries

We started out by working on the models for the peak power of a set of selection queries based on TPC-H, shown in Table 4–4. Figure 4–1 shows the peak power that was measured for the various selection queries, running on the custom-built PCs. For these machines, their idle power is approximately 85W. In the figure, the x-axis refers to the queries that we were issuing to the system, and the y-axis shows the average peak power reached while running each query. For the 2-node and 4-node configuration, the peak power is the average experienced by each of the machines. Recall that in the multi-server configuration the queries are sent to each node, as the data are horizontally partitioned. Notice in the figure

Sql Statement	TPC-H scale	Queries Id
select * from partsupp where ps_suppkey < highSelVal1	5 GB	A
select * from partsupp where ps_suppkey < lowSelVal1	5 GB	A′
select * from lineitem where l_partkey < highSelVal2	5 GB	B
select * from lineitem where l_partkey < lowSelVal2	5 GB	B′
select * from partsupp where ps_suppkey < highSelVal3	20 GB	C
select * from partsupp where ps_suppkey < lowSelVal3	20 GB	C′
select * from lineitem where l_partkey < highSelVal4	20 GB	D
select * from lineitem where l_partkey < lowSelVal4	20 GB	D′

Table 4–4: Selection queries used in the study. Constants lowSelValX and highSelValX are chosen to force low and high selectivity in the queries.

the drop in peak power from the 1-node configuration to the 2-node configuration for queries B′ and D′. By splitting the load, the machines experience less load and hence the power drawn is less. This change is not so pronounced when we move to the 4-node configuration, indicating that the benefits of partitioning the data are less relevant. The trend is that a linear increase in servers does not cause a linear decrease in power.

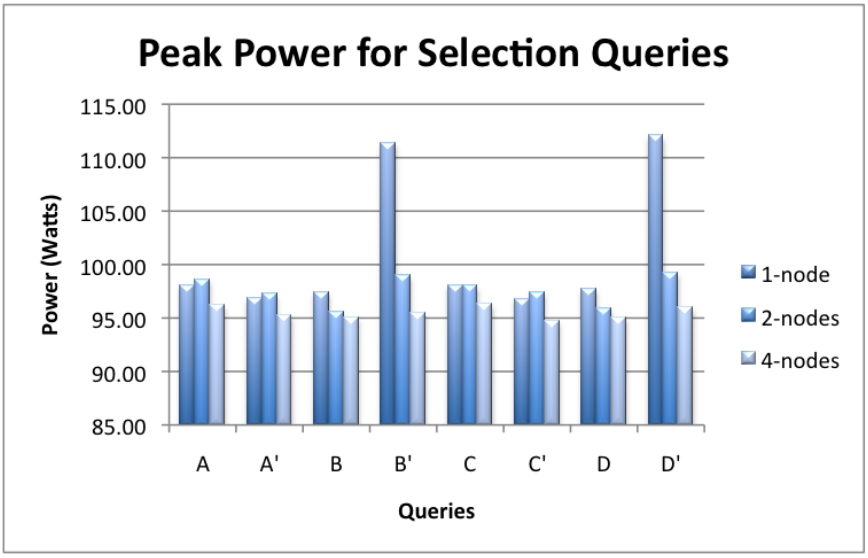


Figure 4–1: Measured Peak Power for Selection Queries.

For selection queries, we started with a model formulation in which the peak power was assumed to be of the form:

$$P_\sigma = \beta_0 + \beta_1 < R > + \beta_2 SF_p(R) + \beta_3 S + \beta_4 |R|$$

This model relates the peak power, P_σ, of a given selection query with the number of columns in the table, the selectivity of the selection predicate, the number of servers in the configuration, and the cardinality of the table used in each query (denoted as R). Notice that we do not include constants such as CPU speed, disk bandwidth, network bandwidth, and so on. The reason for that is that the regression coefficients will capture those factors, so we focused on using predictors that

were actual variables. Notice, however, that the calibration process needs to be run on each type of server machine that might be available for use.

Unfortunately, after closer inspection, we found this initial model not to be statistically sound because our data showed that the there was no linearity. However, our statistical analysis at this stage confirmed that the increase in the number of servers did not result in a linear decrease in peak power, as it was inferred from Figure 4–1. Also, our statistical analysis with ANOVA and p-values showed that the cardinality of the tables was not significant. At first, this was puzzling to us since we expected that large tables would stress the system more. However, one must realize that peak power is not a cumulative measure, but rather an instantaneous measure of system stress. Therefore, the important issue to drive peak power is not how many tuples are processed but rather how much effort is put to process each tuple. If we have tables with many columns, and average tuple size, the disk will be busy reading the tuples, and the CPU will be busy parsing the columns to locate the attributes on which to evaluate the predicates. This situation will simultaneously stress CPU and disk, driving peak power up. We also found that predicates with low selectivity would drive peak power up because more result tuples are sent to the client, and this factor also increases system stress as the CPU needs manage the communications with the client.

We tried several combinations of parameters as regressors. Some of these combinations are not linear, but the important issue in multiple-linear regression is to have linearity between the dependent variable and the regression coefficients [19]. Based on this, we postulated a the model:

$$\begin{aligned} P_\sigma = \quad & \beta_0 + \beta_1 < R > + \beta_2 SF_p(R) + \beta_3 \tfrac{1}{S^2} + \\ & \beta_4 < R > SF_p(R) + \beta_5 \tfrac{<R>}{S^2} + \beta_6 \tfrac{SF_p(R)}{S^2} \end{aligned}$$

In this model, the last three terms capture the interaction between the selectivity factor, the number of columns, and the number of servers. After, running the model through Minitab we arrived to the following cost formula:

$$P_\sigma = \quad 101 + 0.441 < R > -0.0972 SF_p(R) - 6.72 \frac{1}{S^2} + \\ 0.00785 < R > SF_p(R) + 0.692 \frac{< R >}{S^2} + 0.118 \frac{SF_p(R)}{S^2}$$

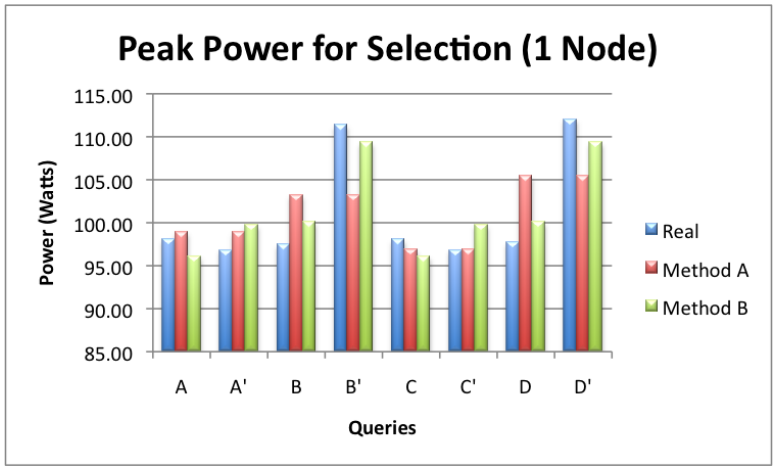
This model was shown to be statistically sound. Notice the direct dependence on selectivity factor and number of columns, with an inverse dependence on the square of the number of servers.

4.3.1 Model Validation

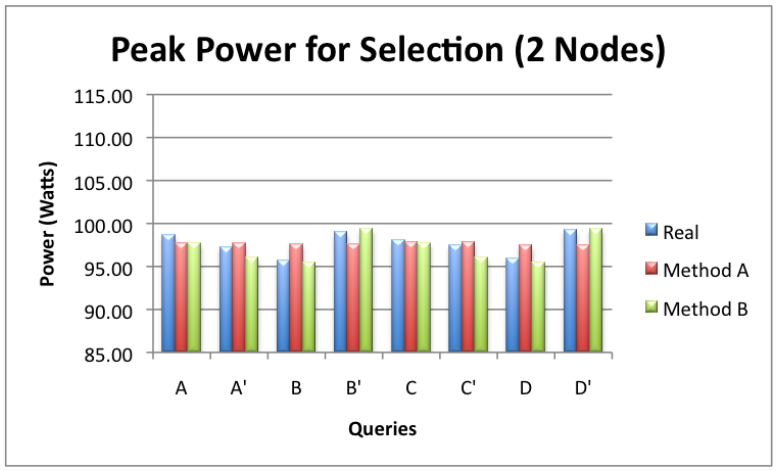
To validate this model as an accurate predictor of peak power, we generated a second set of similar selection queries with different selectivities, and ran them on each server configuration. We recorded their peak power, and we compare them with the peak power predicted by our models (Method B) and with the peak power predicted by System-R models (Method A). Figure 4–2 presents this comparison for each of the server configurations. Notice that as the number of server increases, the power begins to converge to a constant value. Further research needs to be conducted to determine the behavior when more nodes are added and these reach a high utilization level. As we see, our method makes a more consistent approximation of power, particularly when the server utilization is higher (i.e., 1-node configuration). As we said before, the results are similar for average power, so we do not present them here.

4.4 Models for Energy of Selection Queries

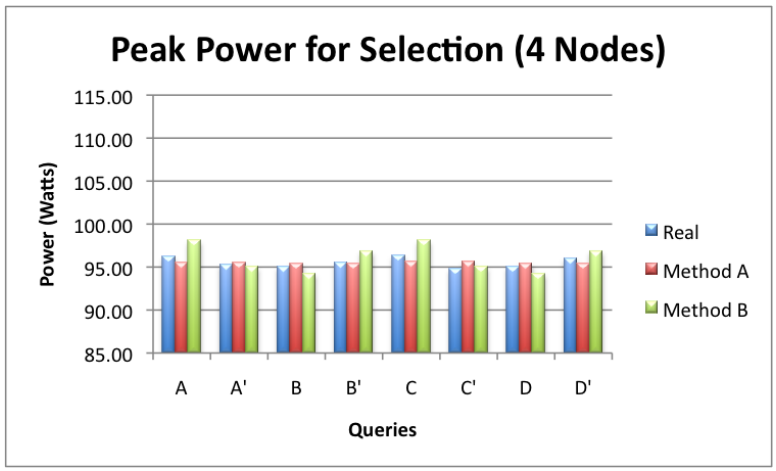
Turning now our attention to energy consumption, Figure 4–3 shows the watt-hours consumed by the test queries we used. Here the x-axis identifies the



(a) One node configuration



(b) Two node configuration



(c) Four node configuration

Figure 4–2: Comparisons of measured peak power with the predictions made by Methods A and B.

queries, and the y-axis indicates energy consumed in terms of watt-hours. As in the case for power, we started out with a model for energy that assumed a linear combination of factors, and it did not work. We recheck on the use of regression, and we realized we could still use it since the linearity had to be on the coefficients and not in the regressors.

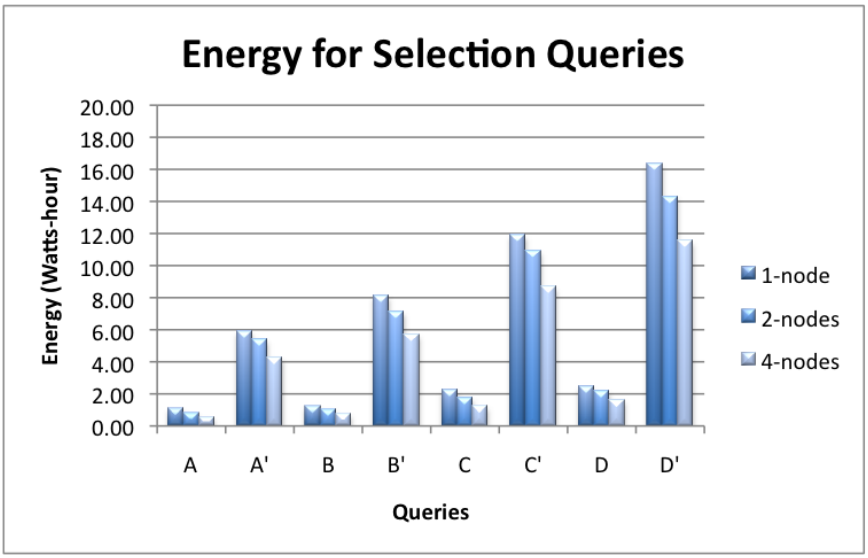


Figure 4–3: Measured Energy Consumption for Selection Queries.

The model that we developed is a follows:

$$E_\sigma = \quad 0.33 - 0.6S + 0.12 < R > +0.0043\,|R|\,SF_p(R)$$

In this case, the relation of energy and the number of servers is proportional to the number of servers. In contrast, in the case of peak power this relationship was inversely proportional with the square of the number of servers. Notice also, that cardinality and the selectivity now become relevant factors. Both behaviors make sense since the total energy consumption is cumulative, and shall be affected by the number of tuples of to be processed (total effort) and the number of servers (machines doing the effort).

4.4.1 Model Validation

We repeated our validation exercise for the energy models. Figure 4–4 summarizes the results. As we see, our methodology makes a better job of predicting the energy compared with Method B, which consistently overestimates and underestimates the value.

4.5 Models for Peak Power of Projection Queries

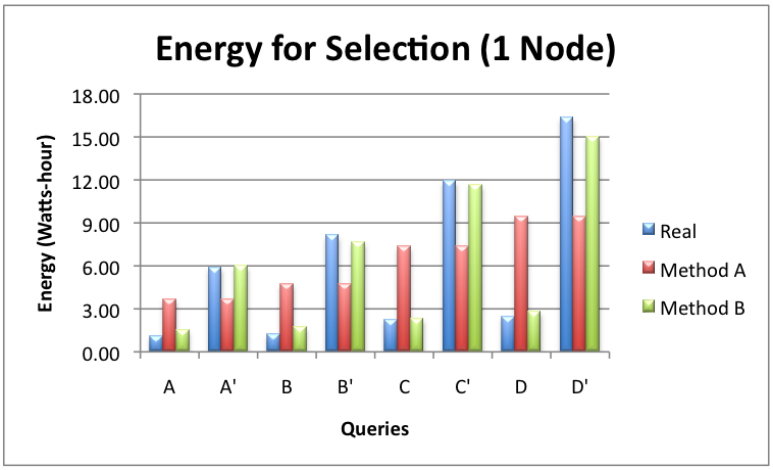
Table 4–5 shows the queries used for experiments with the projection operator. Figure 4–5 shows the peak power cost for projection queries. We see similar trends as in the case for selections. Following the same statistical procedures as done for the case of selections, we arrived at the following model:

$$P_\pi = \ 95.2 + 2.65\frac{1}{S} - 0.380 < R >^2 + 0.0692\frac{<R>^2}{S}$$

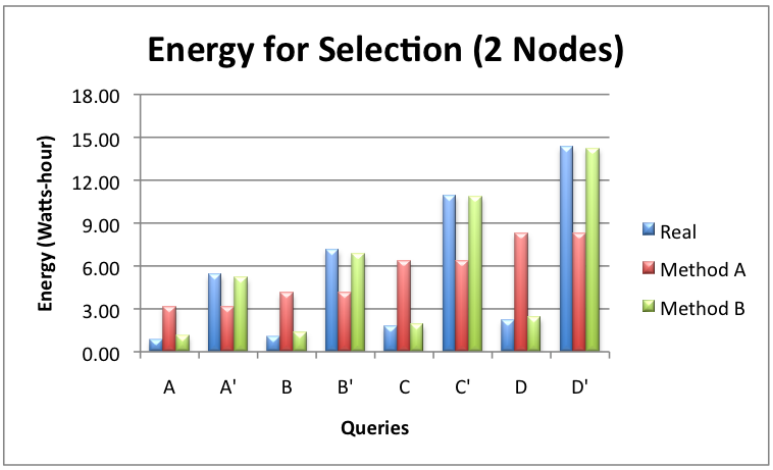
This model is far simpler than the model for selections. Notice that now the number of columns has a more prominent role in the cost, with the last two terms containing the square of the number of columns. In this model, the relation cardinality did not have statistical importance.

Sql Statement	TPC-H scale	Queries Id
select ps_partkey from partsupp	5GB	A
select * from partsupp	5GB	A′
select l_orderkey from lineitem	5GB	B
select * from lineitem	5GB	B′
select ps_partkey from partsupp	20GB	C
select * from partsupp	20GB	C′
select l_orderkey from lineitem	20GB	D
select * from lineitem	20GB	D′

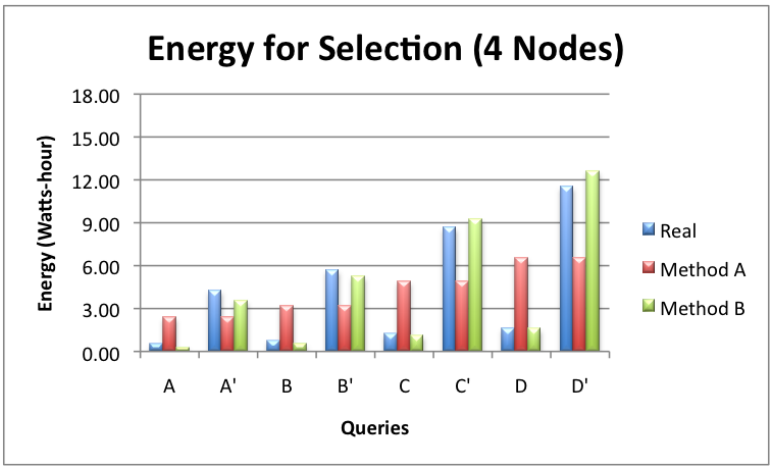
Table 4–5: Projection queries used in the study.



(a) One node configuration



(b) Two node configuration



(c) Four node configuration

Figure 4–4: Comparisons of measured electric energy consumption with the predictions made by Methods A and B.

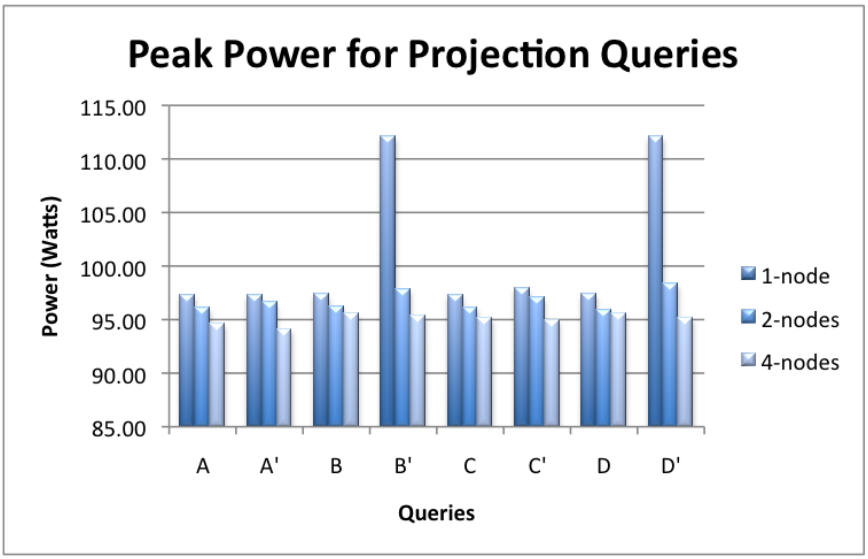


Figure 4–5: Measured Peak Power for Projection Queries.

4.5.1 Model Validation

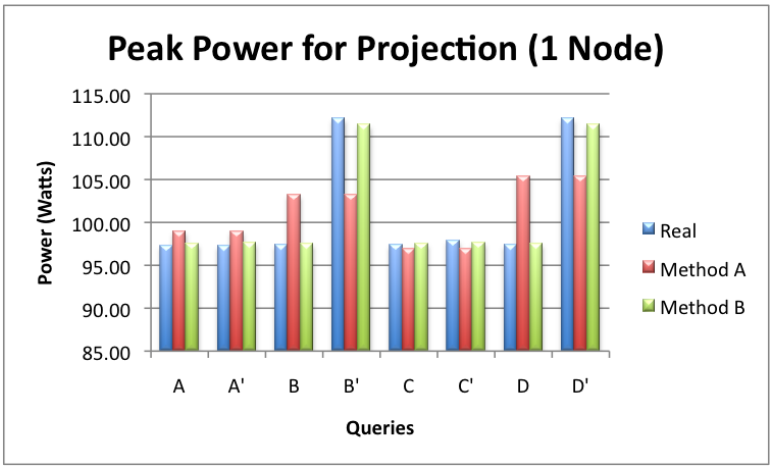
Figure 4–6 shows the validation of the peak power for the projection queries. As we see, our method works quite well for the one node case since the machine becomes more stressed than in the other configurations.

4.6 Models for Energy of Projection Queries

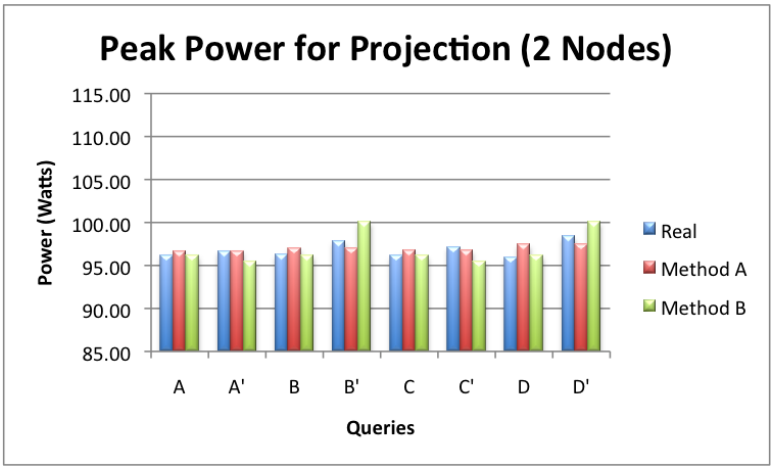
The measured energy costs for projection queries are presented in Figure 4–7. The cost model for energy-consumption in projection queries is as follows:

$$E_\pi = \quad 1.80 - 0.398S - 0.0617S < R > +0.0183 < R > |R|$$

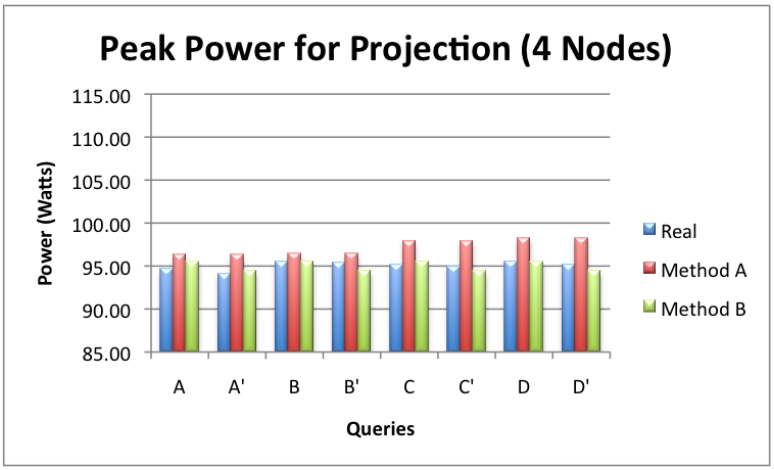
As with case of selections, cardinality plays a statistical significant role in the cost estimation for energy. Notice that the cardinality is present in the only non-negative term containing regressors, and is multiplied by the number of columns. As the number of tuples increases, we can expect this term to be the dominant factor in the energy costs for projections.



(a) One node configuration



(b) Two node configuration



(c) Four node configuration

Figure 4–6: Comparisons of measured peak power in projections with the predictions made by Methods A and B.

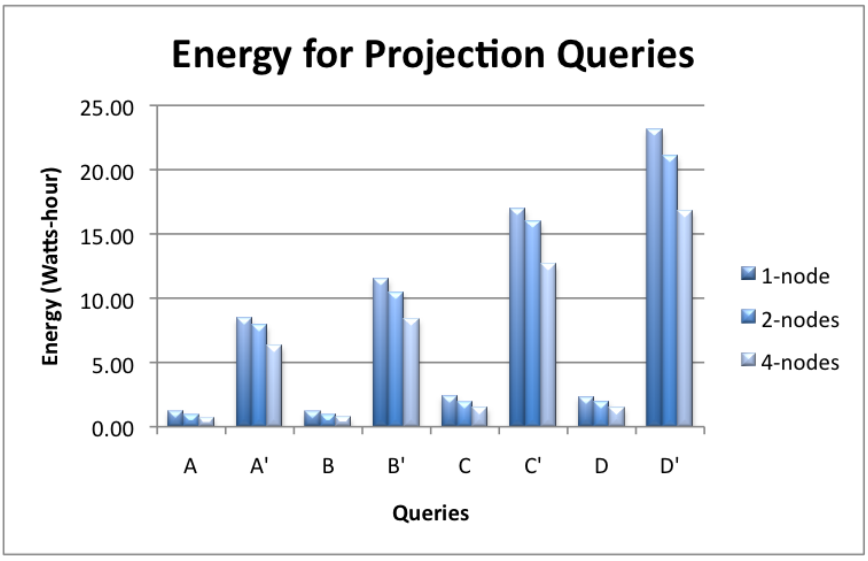


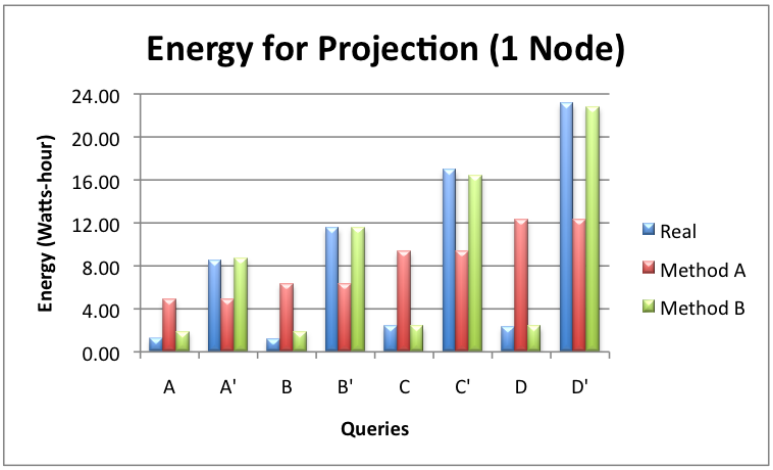
Figure 4–7: Measured Energy Consumption for Projection Queries.

4.6.1 Model Validation

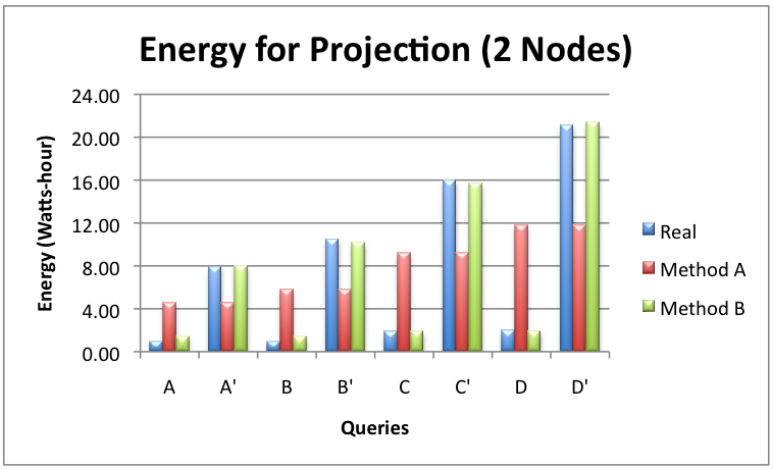
Figure 4–8 shows the results of our validation tests. Again, our regression based methodology, Method B, make a much accurate estimation of the energy cost in comparison with Method A.

4.7 Models for Join Queries

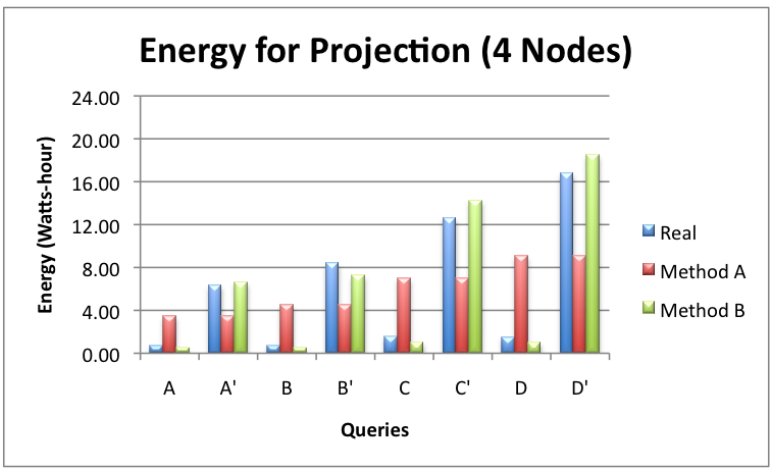
Our join queries are shown in Table 4–6. We started our measurements and analysis for join queries with the expectation of finding complex formulas to model the costs of joins. However, the results in Figure 4–9 suggests that the parameters and the values we chose do not cause significant changes. As we see from the figure, the peak power of the join operations is practically constant regardless of the configuration we used. In the one-node configuration, we let the database server run the query. In the remaining two, we implemented a form of block-oriented nested loops join, moving data from one server to another. But, in all cases the peak power remains nearly constant of around 130 W.



(a) One node configuration



(b) Two node configuration



(c) Four node configuration

Figure 4–8: Comparisons of measured electric energy consumption in projections with the predictions made by Methods A and B.

Sql Statement	TPC-H scale	Queries Id
select * from lineitem, orders where l_orderkey = o_orderkey	5GB	A
select * from lineitem, partsupp where (l_partkey = ps_partkey) and (l_suppkey = ps_suppkey)	5GB	B
select * from orders, customer where o_custkey = c_custkey	5GB	C
select * from lineitem, orders where l_orderkey = o_orderkey	20GB	D
select * from lineitem, partsupp where (l_partkey = ps_partkey) and (l_suppkey = ps_suppkey)	20GB	E
select * from orders, customer where o_custkey = c_custkey	20GB	F

Table 4–6: Join queries used in the study.

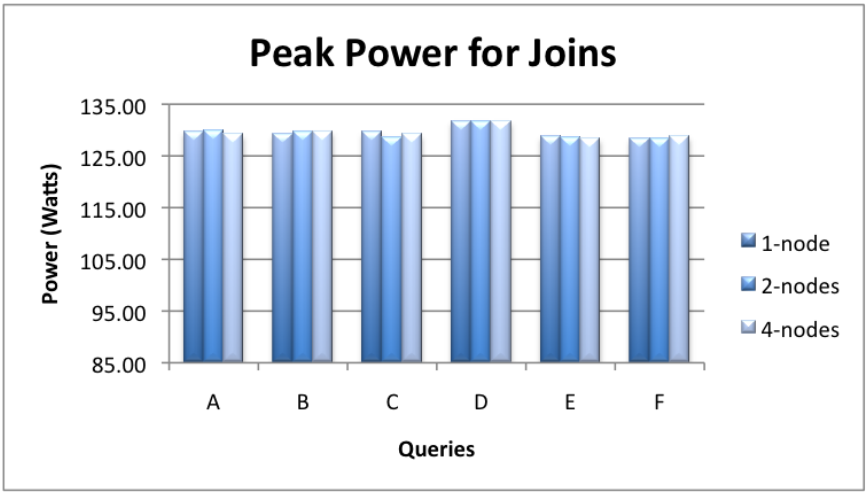


Figure 4–9: Measured Peak Power for Join Queries.

We started looking for explanations to this behavior, and by inspecting the system profile it became clear that the execution is by far dominated by the disk and by the network transfer between the client and the server. Our quad core AMD Phenom X4 CPU is pretty much idle during the evaluation of the queries. Since CPU is the major factor in driving the power of a machine, its idle behavior means that there a few variations in the power drawn. In other words, our join queries, as executed in this setting, became bound by the I/O and network systems, with almost no spikes in the CPU use. This behavior let to a nearly constant peak power for all our queries.

The formula for peak power that our methodology derived is as follows:

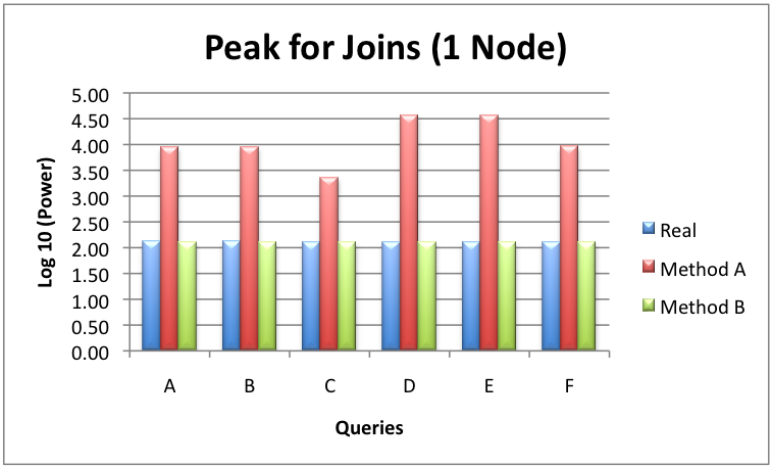
$$P_{\bowtie} = \quad 130 + 0.245 < R > -0.541 < T >$$

Notice that the only factor that has relevance is the number of columns in the joined tables R and T. Moreover, the contribution of this term is somewhat limiting. Unfortunately this formula does not fit well. Although the ANOVA assumptions are met, the coefficient of determination is low, with a value close to 35 %. It means that the model only explains 35 % of the variability of the experiments.

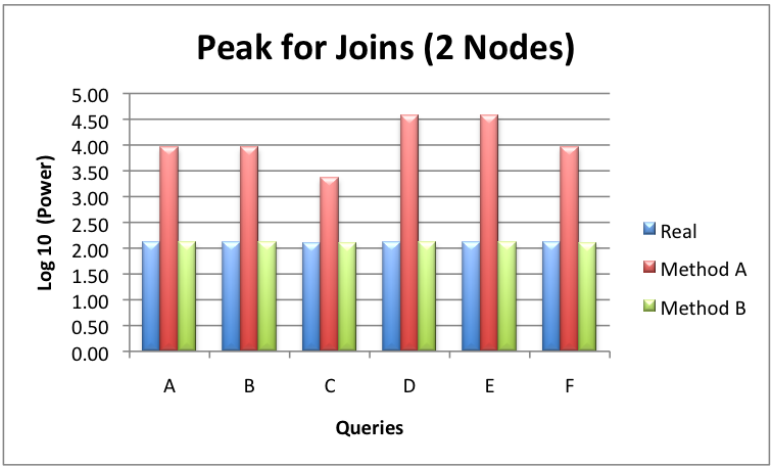
We believe that further research is needed to better understand the behavior of joins. Particularly, we shall perform tests with joins that have complicated projections with User-defined functions (UDF), aggregates, and group-by operations as these will likely cause peaks in CPU utilization.

Figure 4–10 compares the results of estimates for peak power against the actual measurements, and the predictions of Method A. Notice that we had to use a logarithmic scale because Method A overestimates the peak power.

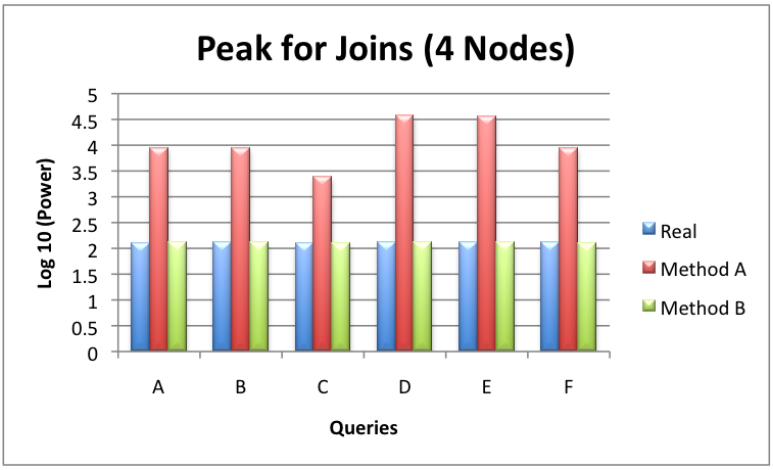
The energy behavior for join queries could be understood better, if one takes into account the two phases of our block nested loop join algorithm:



(a) One node configuration



(b) Two node configuration



(c) Four node configuration

Figure 4–10: Comparisons of measured peak power in joins with the predictions made by Methods A and B.

- Phase 1. Read the first block of the outer table R. Then, retrieve the remote inner table T one page at a time, join it with a block of R and store it into disk.
- Phase 2. Read the remaining blocks of R one page at a time, then join each of these blocks with the inner table T, read locally from disk.

Considering these steps the energy cost can be modeled as:

$$E_{\bowtie} = E_{R_b} + E_{T'} + (\#blocks - 1) * (E_{R_b} + E_T)$$

Where E_{R_b} is the energy of scanning a block of R. $E_{T'}$ is the required energy to retrieve the remote table T and store it into disk. E_T is the energy to read the entire table T from disk and $\#blocks$ is the number of blocks of R. Rewriting $E_{T'}$ as $f * E_T$ and $E_{R_b} + E_T$ as E_{R_bT} we get two equivalences for $E_{\bowtie}$:

$$\begin{aligned} E_{\bowtie} &= \#blocks * E_{R_bT} + (f-1) * E_T \\ E_{\bowtie} &= (\#blocks + f - 1) * E_{R_bT} - (f-1) * E_{R_b} \end{aligned}$$

Here the term f gives the number of times that a block is read or written during a join iteration. From these two equivalences, the energy consumption can be bound by the terms containing E_{R_bT} as follows:

$$\#blocks * E_{R_bT} < E_{\bowtie} < (\#blocks + f - 1) * E_{R_bT}$$

We do not have an estimator for f, because we were not able to measure neither $E_{T'}$ nor E_T. However f must be at least two because each block is read and written at least twice during the process. Now if the upper bound is chosen as approximation of $E_{\bowtie}$ and $f = 2$, then the energy can be express in terms of E_{R_bT} :

$$E_{\bowtie} \approx (\#blocks + 1) * E_{R_bT}$$

The derived formula for E_{R_bT} using our methodology is as follows:

$$E_{R_bT} = 0.122 + 0.0126 < R > + 0.176\,|T| - 0.006\,|T| < T >$$

Notice that the most relevant factors are the cardinality of T and the number of columns of R. Tables with more number of columns stress the CPU more than tables with few number of columns and therefore consume more energy. The scanning of the table T is an I/O bound operation, however as the cardinality increases the time and energy also increases. In this formula the cardinality of R did not appear, that parameter is captured by the term containing $\#blocks$ in the approximated model for the energy.

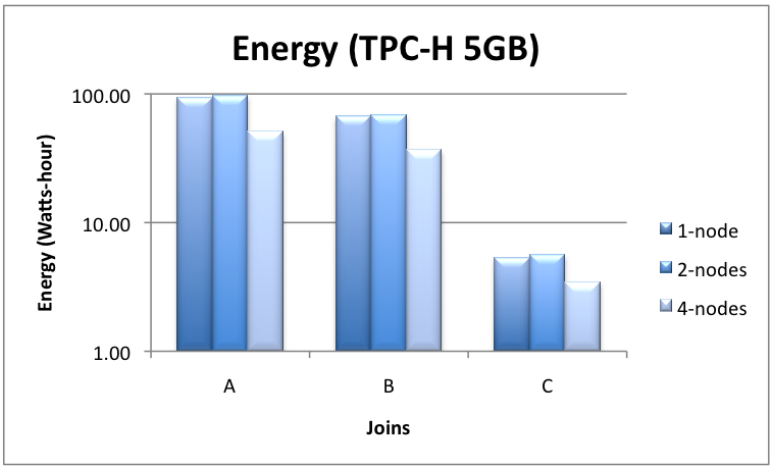
The results for the join queries are shown in the Figure 4–11. Here we use the logarithmic scale because the energy consumption of the C and F join queries are lower than the others.

4.7.1 Model Validation

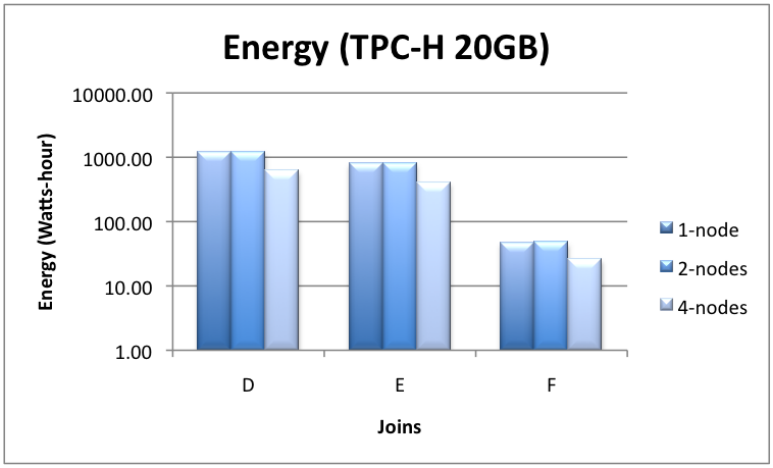
For the validation of the energy for the join queries, we executed our queries with different cardinalities. We could not use the other tables in the TPC-H because these tables are smaller than 1MB (e.g. nation, region). Other tables do not have attributes in common to do a natural join. The Figure 4–12 shows the validation for 2 and 4 servers.

4.8 Effects due to execution platform

We wanted to determine how different platforms would affect the energy, peak power and running times of queries. To do so, we repeated the runs for all our selection, projection, and join queries on the 2-nodes configuration with three different platforms: a) the custom-built PC - denoted as AMD64, b) the Dell Workstation - denoted as PentiumD, and c) a KVM-based virtual machine

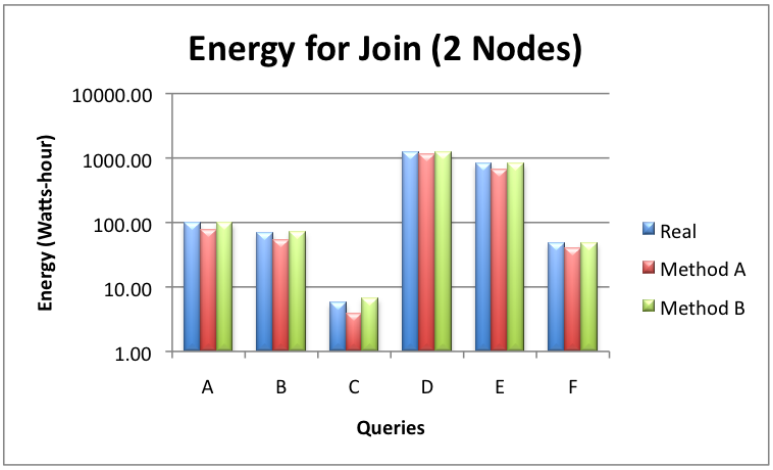


(a) Low data set

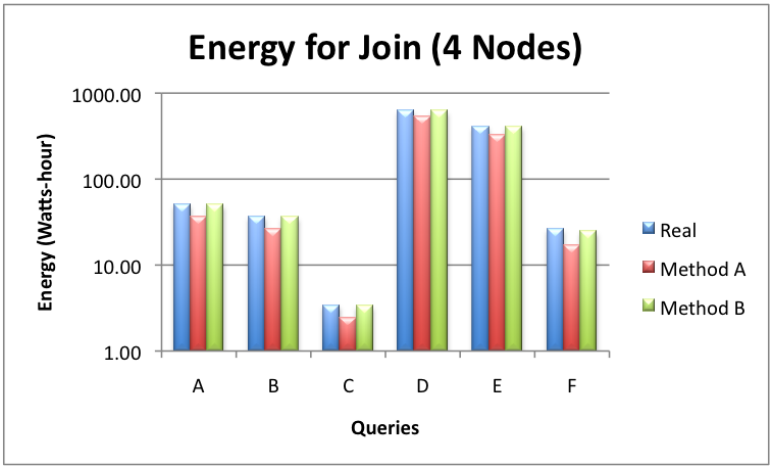


(b) High data set

Figure 4–11: Measured Energy for Join Queries.



(a) Two servers

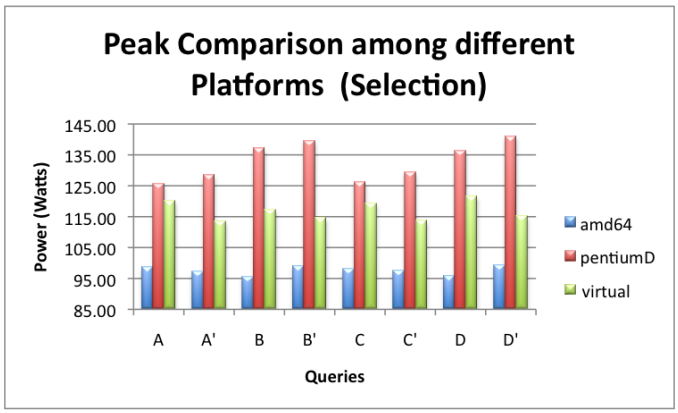


(b) Four servers

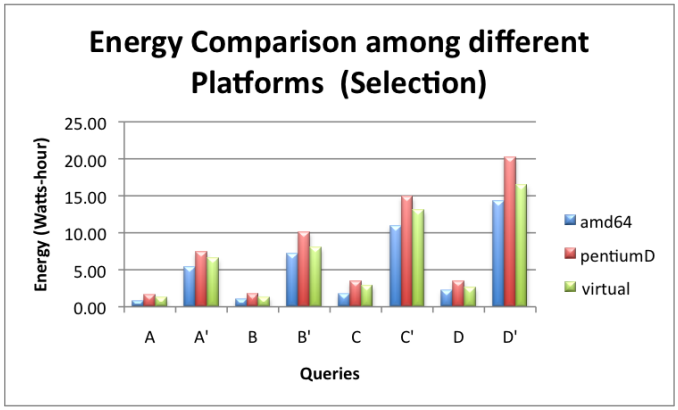
Figure 4–12: Comparison of measured energy consumption in joins with predictions made by Mathods A and B.

running on the custom-built PC - denoted as virtual. In this case, the two nodes where run on two virtual machines running on the same custom-built PC.

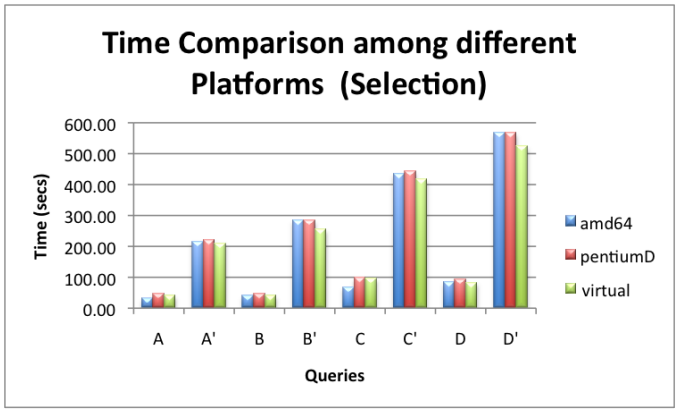
Figure 4–13 shows the peak power, energy consumption, and running times for selection queries. The AMD64 platform provides the best peak power, followed by the virtual configuration, and then the PentiumD configuration. The same patterns happens with energy. But, notice that the running times are very close to each other. The important point to take here is that the virtualized configuration features two virtual nodes running on one machine, and doing the same job that the other configurations do with two separate machines. Although the peak power increases with the virtual configuration, the energy does not double and the running time is practically the same. Thus, by using a virtualized configuration we can have one machine do twice the amount of work with a slight increase in energy. The tradeoff is that the peak power increases considerably, so its accurate estimation becomes important to prevent violations in the power budget. This pattern is also present for projection queries, as show in Figure 4–14. Due to lack of space, we do not include the comparison for joins, but the trend in similar as that for selections and projections. Virtualization really pays off to help curb total energy consumption at the data center.



(a) Peak Power

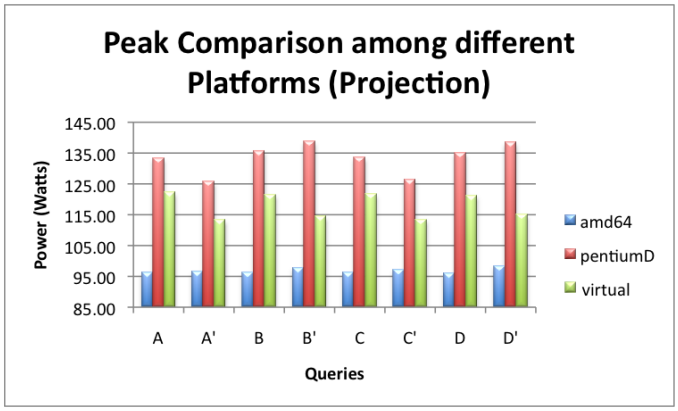


(b) Energy Consumption

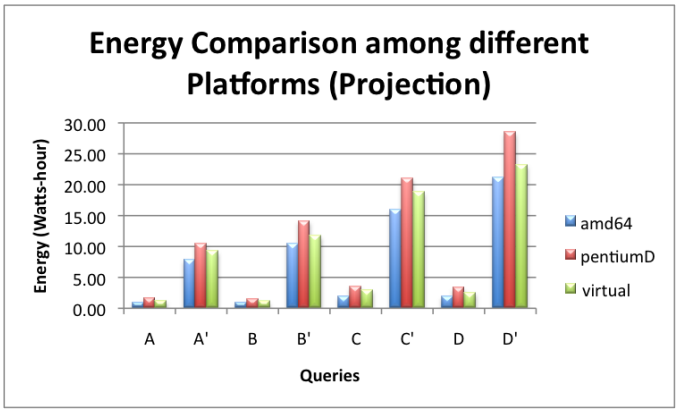


(c) Running Time

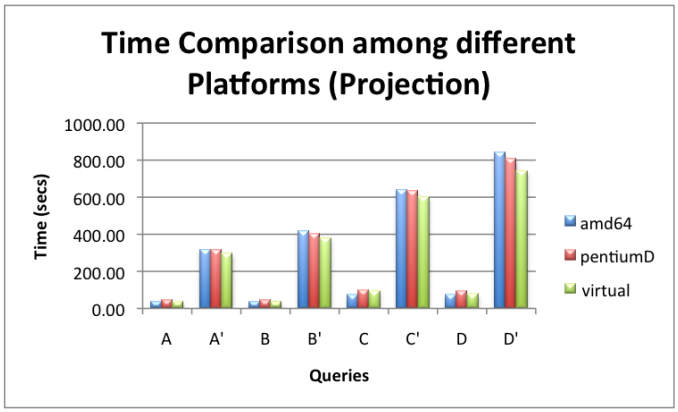
Figure 4–13: Comparison of peak power, energy consumption, and running time for selection queries on various platforms.



(a) Peak Power



(b) Energy Consumption



(c) Running Time

Figure 4–14: Comparison of peak power, energy consumption, and running time for projection queries on various platforms.

CHAPTER 5

CONCLUSION AND FUTURE WORKS

In this thesis, we studied the problem of estimating the power and energy costs of database queries. We introduced a methodology to accurately estimate the power and energy consumption of queries in a database server. Our methodology is based on multiple-linear regression, and a training query workload to derive a cost model that can predict the power and energy cost of queries that are similar to those in training workload. One important feature of our methodology is the fact that there is no need to make measurements on individual hardware components. Instead, the cost models are derived from: a) power/energy measurements taken from internal sensors, or power meters, and b) readily available workload statistics such as relation cardinality, tuple size, number of columns, and number of servers in a multi-server configuration.

We implemented our framework as a set of programs and scripts written in Java, C, R, and Minitab. We performed tests on our framework with TPC-H data sets. In these experiments, we first derived energy/power cost models from a set of data and queries that act as training set. Later, we validated the models by running a second set of queries, and comparing their power/energy cost versus those predicted by the model. Our results show that we can predict power and energy more accurately than alternative methods based on model derived

from those in System-R. We found, however, that join methods are more complex to model since their cost becomes nearly constant when they become I/O bound. We also found that peak power is not influenced by relation cardinality but by selectivity, number of columns in tables, and number of servers used in a multi-server configuration. Finally, we found that energy is more amicable for modeling with these methods. Although further research is needed, these results show that our methodology can be used by tools that perform provisioning of database server machine in a datacenter. Also, our methodology can be incorporated into energy-aware query optimization frameworks for single-site, parallel and distributed database systems designed for clouds and virtualized environments.

5.1 Future Work

We conclude this thesis with a series of tasks that would extend and use the work here presented.

5.1.1 Virtualization

In this work we found preliminary results about the effect of virtualization, but further research and experiments are needed the understand its advantages and limitations. Although Virtualization enables the consolidation of several idle servers into one physical machine, it also causes that the VMs running the database servers compete for the same set of resources. A combination of VMs may exercise the CPUs beyond the power caps used in the datacenter for individual servers. It can also produce heavy access to disk causing process scheduling and I/O scheduling problems that could affect the expected system performance.

5.1.2 Power Consumption in the Presence of Expensive Operators

Modern database servers provided aggregates, sorting functions, and the ability to incorporate libraries with User-defined functions (UDF). All these functions are usually termed as expensive operators because they consume a considerable amount of CPU or I/O to process each tuple. For that reason, we need to develop power/energy models for these operators. We shall extend our methodology to generalized projections and joins queries that includes aggregates, User-defined functions (UDF) and sorting operations.

5.1.3 Query Optimization for MapReduce

The MapReduce programming model [20] has emerged as a key component for building large-scale data processing application in cloud environments. It has also received a lot of attention from the high-performance and database research communities. The Apache Hadoop Project is an open source implementation of MapReduce written in Java. Hadoop has a sub-project called Hive that helps to deal with the lack of schema, query language, cost-based optimization. We plan on investigating how to derive models for all these operators in a Hive and MapReduce setting. This case will be challenging since Hive uses a directed acyclic graph (DAG) of MapReduce tasks to implement a query plan. In contrast, existing database engines use a general tree of operators to represent the query plan. Having a DAG introduces higher dependencies between operators, making the execution and cost models more complex.

REFERENCE LIST

[1] Randy H. Katz. Tech titans building boom. *IEEE Spectrum*, 46(2):40–54, February 2009.

[2] James R. Hamilton. An architecture for modular data centers. In *Proc. 2007 CIDR Conf.*, pages 306–313, Asilomar, CA, USA, January 7-10 2007.

[3] W. Hayt, J. Kemmerly, and S. Durbin. *Engineering Circuit Analysis, 7th ed.* McGraw-Hill, 2007.

[4] New York City Energy Policy Task Force. New york city energy policy: An electricity resource roadmap. A Report to Mayor Michael R. Bloomberg on January 2004, URL: http://www.nyc.gov/html/om/pdf/energy_task_force.pdf.

[5] Luiz André Barroso. The price of performance. *ACM Queue*, 3(7):48–53, 2005.

[6] Stavros Harizopoulos, Mehul A. Shah, Justin Meza, and Parthasarathy Ranganathan. Energy efficiency: The new holy grail of data management systems research. In *CIDR*, 2009.

[7] L. A. Barroso and U. Hölzle. The case for energy-proportional computing. *IEEE Computer*, 40(12):33–37, 2007.

[8] Dimitris Economou, Suzanne Rivoire, Christos Kozyrakis, and Partha Ranganathan. Full-system power analysis and modeling for server environments. In *Workshop on Modeling Benchmarking and Simulation (MOBS) at ISCA*, San Diego, CA, June 2006.

[9] Xiaobo Fan, Wolf-Dietrich Weber, and Luiz Andre Barroso. Power provisioning for a warehouse-sized computer. In *ACM International Symposium on Computer Architecture*, SD, CA, 2007.

[10] Ramya Raghavendra, Parthasarathy Ranganathan, Vanish Talwar, Zhikui Wang, and Xiaoyun Zhu. No "power" struggles: coordinated multi-level power management for the data center. *SIGOPS Oper. Syst. Rev.*, 42(2):48–59, 2008.

[11] Yannis E. Ioannidis and Stavros Christodoulakis. Optimal histograms for limiting worst-case error propagation in the size of join results. *ACM Trans. Database Syst.*, 18(4):709–748, 1993.

[12] Chungmin Melvin Chen and Nick Roussopoulos. Adaptive selectivity estimation using query feedback. In *SIGMOD '94: Proceedings of the 1994 ACM SIGMOD international conference on Management of data*, pages 161–172, New York, NY, USA, 1994. ACM Press.

[13] Christos Faloutsos, Bernhard Seeger, Agma J. M. Traina, and Caetano Traina Jr. Spatial join selectivity using power laws. In *SIGMOD Conference*, pages 177–188, 2000.

[14] Goetz Graefe. Database servers tailored to improve energy efficiency. In *SETMDM '08: Proceedings of the 2008 EDBT workshop on Software engineering for tailor-made data management*, pages 24–28, New York, NY, USA, 2008. ACM.

[15] Willis Lang and Jignesh M. Patel. Towards eco-friendly database management systems. In *CIDR*, 2009.

[16] Zichen Xu, Yi-Cheng Tu, and Xiaorui Wang. Exploring power-performance tradeoffs in database systems. In *ICDE*, pages 485–496, 2010.

[17] Dimitris Tsirogiannis, Stavros Harizopoulos, and Mehul A. Shah. Analyzing the energy efficiency of a database server. In *SIGMOD '10: Proceedings of the 2010 international conference on Management of data*, pages 231–242, New York, NY, USA, 2010. ACM.

[18] Suzanne Rivoire, Mehul A. Shah, Parthasarathy Ranganathan, and Christos Kozyrakis. Joulesort: a balanced energy-efficiency benchmark. In *SIGMOD '07: Proceedings of the 2007 ACM SIGMOD international conference on Management of data*, pages 365–376, New York, NY, USA, 2007. ACM.

[19] Douglas C. Montgomery, Elizabeth A. Peck, and G. Geoffrey Vining. *Introduction to Linear Regression Analysis*. Wiley-Interscience, 4th edition, 2006.

[20] J. Dean and S. Ghemawat. Mapreduce: Simplified data processing on large clusters. In *Proc. of 2004 OSDI*, pages 137–150, San Francisco, CA, USA, 2004.