POWER PROFILING OF DATABASE ENGINES

A Project Report

Submitted in partial fulfilment of the

requirements for the Degree of

Master of Engineering

in

Computer Science and Engineering

by

Mahesh Bale

Computer Science and Automation

Indian Institute of Science

BANGALORE – 560 012

JUNE 2010

TO

My parents

Acknowledgements

First of all I would like to thank my advisor Prof. Jayant Haritsa for giving me the opportunity to pursue my project under his guidance. I thank him, for his constant and valuable guidance, support and encouragement during my stay at IISc.

I would like to thank my project partner Mayuresh Kunjir as well as other members of the Database Systems Lab for providing a stimulating and fun environment for work. I would like to thank my other friends at IISc who have made my stay at IISc memorable.

I thank my parents for their continued support throughout my career.

Abstract

With the total energy usage of computing systems increasing at a steep rate, the traditional *performance and price/performance only* perspective of the designers of computing systems is changing and much attention has been paid towards designing *power* and *energy efficient* systems. Increasing energy usage increases the powering cost of computer systems while overheating caused by a high power usage increases the probability of thermal failures so it increases the cooling cost of the computer systems. Data centers are widely popular and major power consuming computing systems and a DBMS is their major power eater. So a power and energy efficient database engine can lead to significant economic savings in powering and cooling costs.

Motivated by this, in this work, we do power profiling of four popular relational database engines viz. Microsoft SQL Server, Oracle, IBM DB2 and PostgreSQL on TPC-DS decision support benchmark. We found that CPU dynamic power dominates the database server machine's dynamic power and the dynamic power is roughly directly proportional to % CPU utilization. We do the comparison of the Peak Power and Energy Efficiency of the four engines and rank them based on the above metrics. DB2 is found to be most Peak Power Efficient as well as most Energy Efficient. SQL Server is least Peak Power Efficient and PostgreSQL is least Energy Efficient. We also analyzed the temporal power consumption behavior of these engines for TPC-DS queries and found that *sort, aggregate* are the most power eating database operators because of their CPU intensive nature. Higher the CPU utilization higher is the power consumption.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Historically, performance improvement has been the main objective of designing computing systems. But due to the rising electricity costs and steep increase in the energy consumption of computers, system designers are paying critical attention towards power and energy efficiency. Data centers are one of the most widely spread and popular computing systems. Due to rapid increase in their power and energy usage a high fraction of their budget is spent of powering the server and cooling infrastructure. Increasing energy usage negatively impacts density, scalability, reliability and the also environment. A high power usage causes overheating which increases the probability of thermal failure and hence incurs high cooling costs. So power and energy conservation is attracting attention from industrial sectors and research communities. The benchmarks of industrial standard are now paying attention to power and energy as well. SPEC recently announced the industry's first power benchmark SPECpower_ssj2008 [14] to evaluate the power and performance characteristics of servers. The Transaction Processing Council [15] which has been defining benchmarks to measure the performance of large scale transactional as well as decision support systems, has formed an energy specification subcommittee [16] in December 2007 and it is developing its energy specification that will supplement the existing benchmarks.

A DBMS is an important component in the three tier architecture of today's business environments. So a majority of computing resources in a typical data center are dedicated to database servers. This makes DBMS the largest energy consumer among all

the software applications. Hence the DBMSs running in these data centers should be as peak power and energy efficient as possible. Due to high competition, traditionally the DBMS vendors have focused on only performance. So a database engine which is best with respect to performance may not necessarily be best the with respect to peak power and energy efficiency.

Hence the objective of this work is to do power profiling of current database engines. We do a comparative study of the peak power and energy consumption of current database engines and rank the engines with respect Peak Power Efficiency and Energy Efficiency. Peak Power and Energy are two orthogonal metrics. We give Peak Power more importance, as it corresponds to worst-case power and a high Peak Power for even a small duration can cause huge damage. So we also aim to study the temporal power behavior of the TPC-DS benchmark queries on the database engines and try to identify major power-eating database operators.

1.1 Our Contributions

- We compared the Peak Power and Energy consumption of three commercial database engines viz. Microsoft SQL Server, Oracle, IBM DB2 and one public domain database engine PostgreSQL on TPC-DS benchmark and ranked the engines with respect to Peak Power Efficiency and Energy Efficiency. We found DB2 as most Peak Power efficient as well as most Energy Efficient. But other engines fare differently with respect to these metrics. Moreover the best engine with respect to performance is not the best with respect to either of these metrics. To our knowledge, ours is the first study of this kind.

- We found out that CPU power is the most dominant component of dynamic power of our test machine and dynamic power increases in proportion with % CPU utilization. We analyzed the temporal power behavior of the TPC-DS benchmark queries on the four engines. The analysis shows some interesting power patterns and *sort* and *aggregate* are found to be the major power eating operators because of their CPU intensive nature.

1.2 Report Organization

The remainder of the report is organized as follows. In Chapter 2, we summarize the related work in the problem area. Chapter 3 gives a survey of various power measurement methodologies. Chapter 4 describes our experimental setup. In Chapter 5 we find out CPU as the most dominant power eater. In Chapter 6 we do the comparison of four database engines on Peak Power and Energy metrics followed by the analysis of temporal power behavior of the TPC-DS queries on the four engines in Chapter 7. Finally we conclude with the summary of work and directions for possible future work in Chapter 8.

Chapter 2

Related Work

To the best of our knowledge there is no published work on comparing the Peak Power and Energy consumption of various database engines. There is some work done on comparing various hardware configurations based on Energy Efficiency. [2] developed an external sort benchmark and used it for evaluating the Energy Efficiency of wide range of computer systems from clusters to handheld devices. The metric for Energy Efficiency used by them is the number of records sorted per Joule of Energy. TPC [15] has been publishing its results on the comparison of various database system configurations based on performance . The results are for various hardware and software (operating system and database engine) configurations on TPC benchmarks like TPC-C [18], TPC-H [19] etc.. Recently TPC has started developing an Energy Specification [16] which augments energy metrics to all its existing benchmarks. But the specification is still under development and there are no published results on the energy metric. [3] did a Peak power consumption analysis of best TPC-C results published over a duration of 6 years and compared the performance and power trends for these results. Thus they compare how the Peak power of the best performing configurations varied in these 6 years.

[4] studied the power consumption of TPC-H queries running on PostgreSQL and categorized the queries into two classes. One corresponding to those having power saving potential while other corresponding to those not having such potential. But their work is with respect to average power and on only one database engine. Peak power is more important and no comparison of various database engines based on it has been done. There

is no prior work done on analyzing the temporal power behavior of the TPC benchmark queries on the various database engines.

Chapter 3

Power Measurement Methodologies

There are broadly three ways in which one can do the power and energy profiling of the programs running on the computer systems. The first way is *simulation*, in which a cycle accurate simulation of the program execution is done on a simulator like *Wattch* [8]. The simulator estimates the power consumption of the program using an inbuilt power model. The second approach is by using a power model developed using hardware performance counters. [9, 10] used this method to do power estimation. But to use this approach, we need to develop our own power model using existing hardware performance counters to estimate the power consumption of database engine. And even after that we get only an estimation. To get real power numbers we decide to follow the third approach of using a hardware device to do power measurement. This approach gives the real power numbers which help in direct comparison between power and energy usage of two alternative hardware or software approaches. There are various power measurement devices which were used in the past by research community which we describe in the next paragraph.

[7] used a Data Acquisition Card (DAQ) [13] manufactured by National Instruments. The DAQ card can measure up to 16 components like hard disk, CPU etc. simultaneously. But this instrument can measure only voltages. So one has to measure current drawn by a component to get its power consumption , which is simply the product of voltage to current. But to measure current one has to break the circuit and add a resistor called current sensing resistor in series with the measured component. The voltage drop across

the resistor (measured by DAQ) divided by its resistance will give the current. But the resistance of the resistor should be sufficiently small so that it won't change the current flowing through the measured component by a large extent. This approach is *invasive* as it requires breaking the circuit of computer. So we also looked at *non-invasive* power measuring instruments. Two such digital instruments are the Brand Electronics [11] Power Meter of model 20-1850/CI used in [2, 3] and the *Watts up? PRO* [12] Power Meter used in [4]. Both these meters can measure the power of the whole system. So to get the power consumption only due to running queries on database engines, we need to take the difference between the measured power when the query is running and the measured power when the system is in idle state. Both these meters have a resolution of 1W and are capable of logging the power values to a computer at sampling interval of 1 second. We used the The Brand Electronics Power Meter for all the measurements in our work. We experimented on TPC-DS [17] database of size 100 GB, on which the TPC-DS benchmark queries run for several minutes. So the sampling interval of one second is sufficient for us. We assume that the power consumption remains constant for one second period between the two consecutive instants of sampling and this constant value is equal to the power value corresponding to the former instant.

Chapter 4

Experimental Setup

In this section we describe the setup under which all our experiments were made. We start with description of our power measurement setup in Section 4.1, followed by the description of test environment in Section 4.2.

4.1 Power Measurement Setup

As already mentioned in Section 3, we are using a digital power meter model 20-1850/CI manufactured by Brand Electronics for our power measurements. The meter has power measurement resolution of 1 Watt with an accuracy of $\pm 1.5\%$. Figure 4.1 shows how to make the connections to do the power measurements. The Power Meter has to be connected to an electricity supply outlet and it has a socket to which the power cord of any electrical appliance can be connected. The power to the appliance comes directly via the meter and the meter records the power drawn by the appliance. A computer whose power consumption is to be measured is called the *monitored* computer and has to be connected to the socket of the meter. The power value recorded by the meter at any instant is the power consumption of the monitored computer at that instant. The meter has a computer interface cable which has a serial port connector at the other end. Through this cable it is capable of transmitting the measured power values to any computer having a serial port. We call this computer as the *monitor* because it will monitor the power consumption of the monitored computer. The monitor runs the Power Logging Program of the power meter. This program logs the power readings transmitted by the meter to

a file at a sampling rate of one reading per second. For each logged power value, the timestamp at which it was measured is also logged.

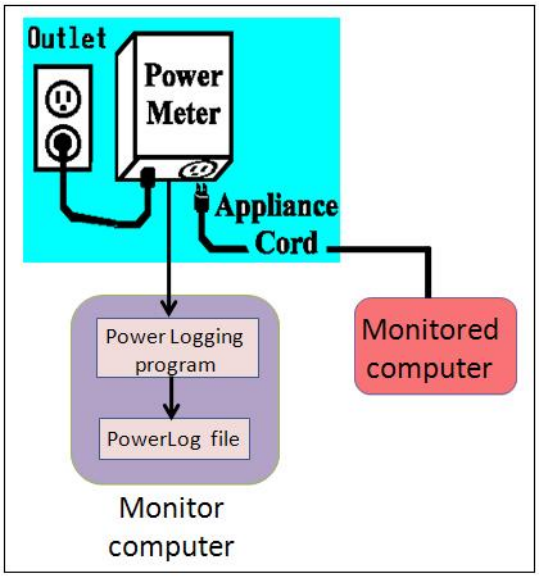


Figure 4.1: Power Measurement of a computer using Brand Electronics Power Meter

In our experiments the *monitored* computer is the database server computer which runs the database engine. We disconnected this computer from network to avoid the unpredictable power consumption of network interface card due to input traffic and in addition to that used a separate *monitor* computer to ensure that we get the power consumption due to activities of database engine and not due to monitoring activities.

Peak Power and Energy Calculation Since we want to determine the power and energy consumed by the database server only due to the execution of the query, we first determine the *ambient* power of the database server. The *ambient* power is the power consumption of the database server when no query is being executed. The server still consumes some power during this idle state because of the idle state power consumption of processor, memory, rest of the mother board and hard disk. But this power is nearly constant. Before the start of every execution we note the ambient power. The start and end times of the query execution are also noted down. To get the dynamic (active) power of the database server only due to query execution we subtract the ambient power from the logged power values. Once we get the dynamic power samples corresponding to the duration of query

execution, its energy and average power can be calculated easily. Energy as we know is the integration of the power with respect to time. Since our sampling interval is one second, summing up all the active power samples (in Watts) for the query, we get its energy cost in Joules. The Average Power is then simply energy divided by the execution time. The Peak Power is determined as the maximum value of an active power sample.

4.2 Test Environment

In this section we describe the configuration of the test machine used as database server, the database engines under study and the workload on which they are studied.

Workstation	Sun Ultra 24
Processor	Intel Core 2 Extreme Quad Core: 3.0 GHz
RAM	8 GB
Hard Disks	4*300GB SAS: 15000 RPM
Operating System	Windows Vista Business: 64 bit

Table 4.1: Database Server Configuration

4.2.1 Database Server Configuration

Table 4.1 shows the configuration of the machine used as the database server in our experiments. It is a quad core machine with 8 GB RAM and 1.2 TB disk space and thus a reasonably good database server.

4.2.2 Database Engines

The Peak Power and Energy comparison of a suite of three popular commercial relational database engines viz. **Microsoft SQL Server (2008 Edition)**, **Oracle (Version 11g)** and **IBM DB2 (Version 9.7)** and one public domain relational database engine-**PostgreSQL (Version 8.4.2)** is done in this work. Some of these engines offer a range of query optimization levels that tradeoff result latency versus response time. All our experiments are done at the highest level of optimization on all the engines.

Engine	**Setup Time (days)**	**Avg. Loading Power (Watts)**
Microsoft SQL Server 2008	14	2
Oracle 11g	7	1
DB2 9.7	14	2.6
PostgreSQL 8.4	45	1

Table 4.2: The setup time and average loading power for TPC-DS 100 GB database on different engines

4.2.3 Workload

All our experiments are done on TPC-DS [17] benchmark which is the new generation industry standard decision support benchmark. It models the decision support functions of a multi-channel retails product supplier. The database schema is highly complex with a total of 24 tables. There are three channels *store*, *catalog* and *web*, each of which has one *sales* and one *returns* table. These six plus the INVENTORY table, constitute the *fact* tables of the schema. The other tables are mainly the *dimension* tables which store critical business data like information about customers, items, orders etc. The scale 1 size of the benchmark database is 100 GB and we used this size for our experiments. All the engines were loaded with the 100 GB database and the statistics were created on all the columns of all tables. The setup time for an engine includes the data loading and statistics creation time. The setup times and the average power consumption during data loading for the four engines is listed in Table 4.2. As it can be seen from the Table, PostgreSQL takes the maximum setup time while Oracle takes the minimum setup time. The average power consumption during loading is very small for all the engines. The power consumption is found to remain very steady for the whole loading process and the value in table shows average power over the duration of an hour.

The TPC-DS benchmark has a set of 99 highly complex queries with various operational requirements (e.g. reporting, data extraction, ad-hoc etc.). In all our experiments we used the TPC-DS benchmark queries in *as-is* manner.

4.2.4 Main Memory and Cache Configuration

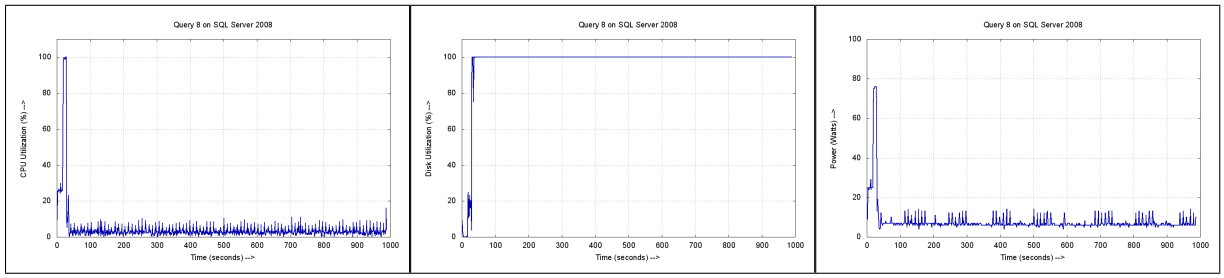
Main Memory In order to do a fair comparison of the power consumption of the various engines, an equal amount of physical memory (RAM) is given to each of them. On most of the engines there is a single parameter which specifies the amount of global memory it can use and the engine will distribute this memory among its various sub-components. We configured all the engines to have an upper bound of 6 GB out of 8 GB RAM that the test database server machine has (Table 4.1).

Cache State Every query execution in our experiments is done on cold cache. This is enforced by two steps before every execution. First the database process cache is cleaned up restarting the database engine's server process and then the operating system cache is also cleaned up by execution of a sequential scan query on a large table which is not present in the current query. We used four large tables for the latter step and they are STORE_SALES, CATALOG_SALES, WEB_SALES and INVENTORY.

Chapter 5

Where Does Power Go?

Any computer system's power consumption has two components viz. ambient (idle state) power and dynamic (active state) power. The *ambient* power is the idle state power consumption of its hardware which consists of CPU, memory, rest of the mother board and hard disks. For our test system the ambient power was found to be around 130 Watts and the system power range was found to be from around 130 Watts to 210 Watts. So the dynamic power range is from 0 Watts to 80 Watts. The dynamic power of memory and the mother board is negligible [6] while CPU and hard disk are the major contributors of the dynamic power. So the dynamic power of the computer is practically the sum of dynamic power values of CPU and hard disks.



(a) % CPU Utilization (b) % Disk Utilization (c) Dynamic Power Consumption

Figure 5.1: Variation in % CPU Utilization, % Disk Utilization and Dynamic Power Consumption of test system during the execution of TPC-DS Query 8 on SQL Server

The dynamic power of any hardware device is directly proportional to its % utilization. There are many hardware performance counters in modern hardware which count and record various hardware events. These counters can be used to get a good approximation of % utilization of the device. The counter [*Processor(_Total)\% Processor Time*] gives a good approximation of the % *CPU utilization*, which is the average utilization over all cores (4 in our case) collectively. Another counter [*PhysicalDisk\% Idle Time*] gives the approximate % *Disk Idle Time*. The % *disk utilization* is simply 100 minus this number. We used a tool called *Perfmon*, provided by the Windows operating system for monitoring and logging these counters. We executed some TPC-DS benchmark queries on the test machine (Table 4.1) and logged % CPU utilization, % disk utilization and the dynamic power consumption of the machine during the query execution. Figure 5.1 shows the variation of this metrics over the duration of execution for Query 8 of TPC-DS benchmark on SQL Server. As it can be seen there is a very strong correlation between % CPU utilization (Figure 5.1a) and dynamic power of the machine (Figure 5.1c). When CPU utilization reaches near 100%, the dynamic power of the system jumps to 76W and the overall shapes of the two plots are very similar. There is no such correlation between % disk utilization (Figure 5.1b) and dynamic power (Figure 5.1c). In fact after first 40 seconds of execution the % disk utilization is 100% but the dynamic power during the period is very low. Thus disk dynamic power contributes very little to the machine's dynamic power, and hence the CPU dynamic power dominates the dynamic power of the machine. Due to strong correlation between dynamic power of machine and % CPU utilization, dynamic power of the machine is practically directly proportional to % CPU utilization. Thus high CPU utilization contributes to high dynamic power and to reduce the Peak Power one should reduce the Peak CPU utilization.

Chapter 6

Peak Power and Energy Comparison of Engines

We executed the TPC-DS benchmark queries on all the engines on the 100 GB database. For major part of our experiments the queries were executed *serially* i.e. only one query at a time. In Section 6.1 we present Peak Power and Energy comparison results based on *serial* execution of queries. We also experimented with *concurrent* execution of multiple queries. Section 6.2 discusses the effect of concurrent execution of queries on Peak Power and Energy.

6.1 Comparison Based on Serial Execution of Queries

6.1.1 Peak Power Comparison

We tried to execute all the 99 TPC-DS benchmark queries on all the four engines in serial (one at a time) manner. But there were some issues while executing all the queries on all the engines. For example PostgreSQL does not have an implementation of data cube and so the 11 queries involving that operator couldn't be executed on it. Some queries didn't finish execution even after waiting for long times in the tune of 5-10 hours. This happened for 2 queries on SQL Server, 12 on Oracle, 13 on DB2 and 8 on PostgreSQL. Figure 6.1 shows the Peak Power Comparison Graph for the 99 queries on the four engines. All the Peak Power values shown in the Graph are dynamic power values (above ambient) and

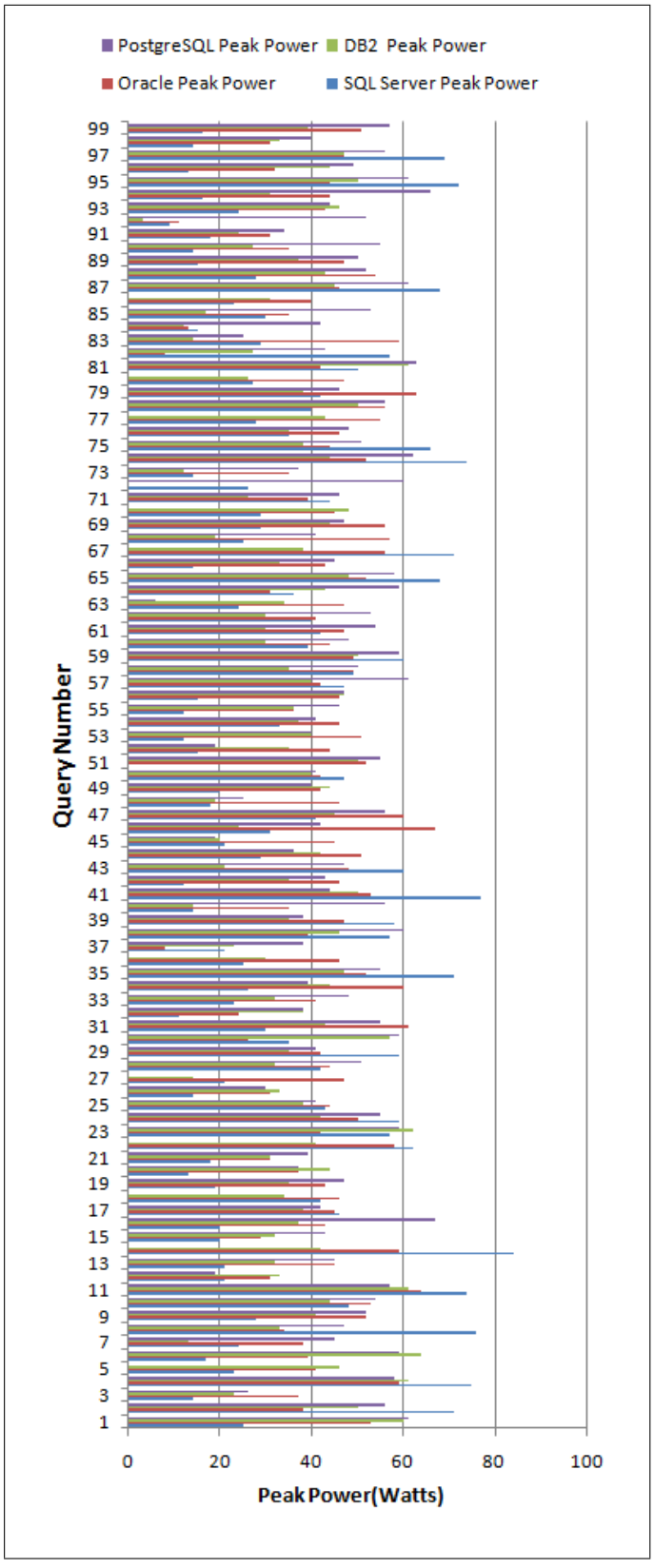


Figure 6.1: Peak Power Comparison Graph for 99 TPC-DS benchmark queries

hence only due to query execution on the corresponding database engine. For queries which didn't finish on a particular engine, the graph shows the Peak Power for only the duration of execution.

It can be seen from the Peak Power Graph that for many queries the current database engines differ a lot w.r.t. the Peak Power. In fact for 57 queries the standard deviation of the Peak Power values for the four engines is greater than 10W. For example, the peak power values for SQL Server, Oracle, DB2 and PostgreSQL for Query 94 are 16W, 44W, 31W and 66W respectively, with the corresponding standard deviation being about 21W. For this query PostgreSQL's peak power is 50W more than that of SQL Server, which is the maximum difference between the peaks for any two engines for any query. The highest Peak Power observed among all the engines and all queries is 84W for Query 14 on SQL Server. The highest Peak Power observed for Oracle, DB2 and PostgreSQL are 67W, 64W, 67W respectively.

Engine	**Low-Peak Queries**	**Medium-Peak Queries**	**High-Peak Queries**	**Total Number of Queries Categorized**
SQL Server	42	33	23	98
Oracle	5	79	14	98
DB2	16	75	7	98
PostgreSQL	6	58	24	88

Table 6.1: Peak Power distribution for all the engines

Based on the Peak Power values we classified the queries into three categories for each engine. The three categories are Low-Peak queries (Peak $\leq$ 25W), Medium-Peak queries (25W $<$ Peak $\leq$ 55W) and High-Peak queries (Peak $>$ 55W). These three categories correspond to low peak CPU utilization, medium peak CPU utilization and high peak CPU utilization. Table 6.1 shows for each engine, the distribution of queries into above three categories. Only the queries which could be executed (not necessarily to completion) on the respective engine are considered in the counts. SQL Server has its highest number of queries in Low-Peak category, but it also has a large number (23) of queries in high peak category. The other three engines have their highest number of queries in Medium-Peak category. 56 queries on SQL Server and more than 80 queries on the other engines have peak power values in Medium or High-Peak category. PostgreSQL and SQL Server have about 25% of queries in High-Peak category.

Since all 99 queries could not be executed to completion on all engines, for ranking of engines based on Peak Power we decide to select a representative subset of the set of all queries, such that all queries in this subset could be executed to completion on all the engines. Even for ranking based on Energy in Section 6.1.2, we use the same representative subset.

Ranking Engines Based On Peak Power We singled out a subset containing 16 queries which is a good representative of the set of all 99 queries. The choice of these queries was motivated from [5], in which the authors classified the TPC-DS benchmark queries according to schema coverage (coverage of fact and dimension tables). The classification is as follows:

- Queries involving only dimension tables (6 in number)
- Queries involving single fact table (54 in number)
- Queries involving multiple fact tables
  - With join of subqueries (22 in number)
  - With union of subqueries (17 in number)

The 16 queries we singled out are the queries numbered 8, 16, 24, 41, 49, 57, 58, 59, 61, 64, 66, 76, 82, 83, 88 and 98. This set contains 1 query from first category, 9 from second and 3 each from two sub-categories of the third category. These queries exhibit a wide range of SQL features ranging from huge aggregates to CASE statements.

Engine	**Max[Individual Peak Power]** (Watts)	**Number of High-Peak queries**
SQL Server	77	5
Oracle	59	1
DB2	50	0
PostgreSQL	67	4

Table 6.2: Peak Power metrics for 16-Query subset for four engines

We could execute each of these 16 queries to completion on all the four engines and hence have the exact Peak Power values for each of them for each engine. Figure 6.2 shows the Peak Power Graph for these 16 queries. Now we rank the engines based on Peak

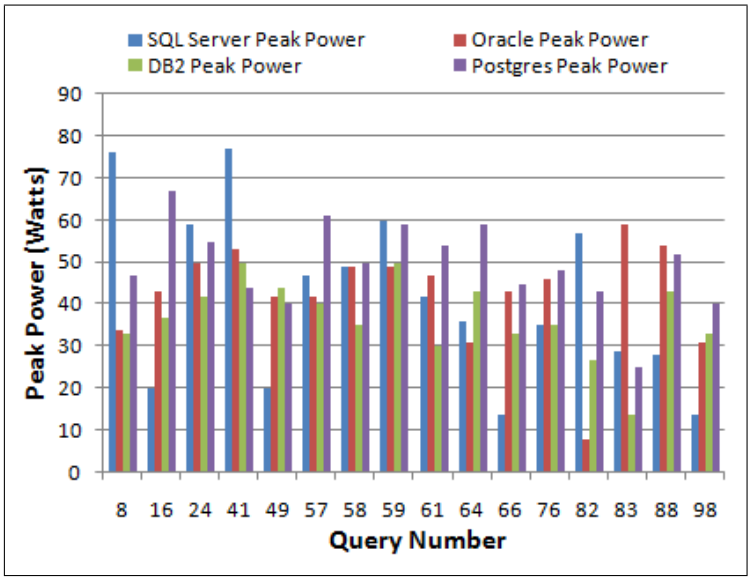


Figure 6.2: Peak Power Graph for Representative Subset

Power Efficiency. The metric we used for this is **Max [Individual Peak Power]** which is the maximum of individual Peak Power values for 16 queries. This metric corresponds to the worst-case power consumption of the database server running any of these queries. Since high Peak Power leads to high probability of system failure due to overheating and it also incurs a high cooling cost, we want to minimize the metric. Lesser the value of the metric higher is the Peak Power Efficiency. Table 6.2 shows the value of the metric for the four engines. So the ranks with respect to the Peak Power Efficiency are DB2 first, Oracle second, PostgreSQL third and SQL Server last (fourth).

Table 6.2 also lists the number of queries (among the 16 representative queries) in the High-Peak category (Peak $>$ 55W) for each engine. One may argue that higher the number of queries in the High-Peak category for an engine, higher are the chances of system failure and higher cooling cost and lesser should be the Peak Power Efficiency of the engine. The ranks of engines based on this metric for Peak Power Efficiency come exactly the same as for the the metric Max [Individual Peak Power].

Metric	SQL Server	Oracle	DB2	PostgreSQL
Total Execution Time (hh:mm:ss)	07:53:37	04:02:37	05:08:50	11:14:46
Average Power (Watts)	10.06	26.30	11.17	35.14
Total Energy (Joules)	285878	382896	206997	1422490

Table 6.3: Total Execution Time, Average Power and Energy for the Serial Execution (one at a time) of 16 Query Subset on all engines

6.1.2 Energy Comparison

Since not all the 99 queries from TPC-DS benchmark set could be executed to completion on all the engines, we don't have total energy figures for many of the queries and hence we can't compare engines based on this incomplete information. So we used the same 16-Query Representative Subset used for ranking based on Peak Power Efficiency, to do rank the engines based on Energy Efficiency.

Ranking Engines Based on Energy Efficiency Table 6.3 shows the Total Execution Time, Average Power and Total Energy for the serial execution of 16 queries in the Representative Subset. The Average Power here is the average over all the 16 queries as if they were executed one after another in serial manner. PostgreSQL has the highest Total Energy while DB2 has the lowest Total Energy. The three commercial engines take less than 30% the Total Energy of PostgreSQL. SQL Server takes about 50% more energy than DB2 while Oracle takes about double the energy as DB2. Lesser the Total Energy, higher is the Energy Efficiency. If we rank the engines based on Energy Efficiency then DB2 comes first, SQL Server second, Oracle third and PostgreSQL fourth. Thus Oracle, the best engine w.r.t. performance (Total Execution Time), is not the best w.r.t. Energy Efficiency. DB2 whose Total Execution Time is about 25% more than that of Oracle, consumes about half the Energy as Energy of Oracle, because its Average Power is very less compared to that of Oracle.

6.2 Effect of Concurrent Execution of Multiple Queries

Peak Power and Energy comparison in Chapter 6.1 was done based on serial execution of queries. In a real world decision support system multiple queries can be submitted

Metric	**SQL Server**	**Oracle**	**DB2**	**PostgreSQL**
Max [Individual Peak Power] (Watts)	60.00	49.00	50.00	61.00
Peak Power for Concurrent Execution (Watts)	65.00	51.00	50.00	69.00
Sum of Individual Execution Times (hh:mm:ss)	03:12:30	00:48:53	02:08:54	01:30:54
Total Time for Concurrent Execution (hh:mm:ss)	02:48:00	00:45:03	02:05:11	01:12:07
Average Power for Serial Execution (Watts)	12.83	12.99	13.95	27.31
Average Power for Concurrent Execution (Watts)	11.89	12.50	12.40	34.95
Sum of Individual Energies (Joules)	148175.00	38114.00	107825.00	148967.00
Total Energy for Concurrent Execution (Joules)	119751.00	33740.00	93026.00	150980.00

Table 6.4: Metrics for Serial and Concurrent Execution of the set of 5 queries: 57, 58, 59, 61, 64

simultaneously by different clients causing a huge load on the system. So we also experimented with concurrent execution of multiple queries. This was achieved by opening multiple client instances and connecting them to the single instance of the database server. The queries we used for these experiments are from our representative subset of 16 queries. We did these experiments for all the four engines and compared the metrics obtained for concurrent execution of a set of queries with the gross numbers based on serial execution of the same set of queries. Table 6.4 shows gross numbers for four metrics (Peak Power, Execution Time, Average Power and Energy) for a set of 5 queries which are the queries numbered 57, 58, 59, 61 and 64. For each metric we show two values viz. the aggregate of the values for the serial query execution and the value for concurrent execution.

As it can be seen from the Table 6.4, the Peak Power for concurrent execution is greater than or equal to the maximum of serial peak power values. The Peak Power has increased due to concurrent execution for all engines except DB2. The maximum increase observed is 8W for PostgreSQL. The increase in Peak Power was due to increase in peak CPU utilization during concurrent execution. Due to the inter-query parallelism the execution time has also reduced for all the engines, but the reduction is very marginal. The Total Energy has reduced for all engines except PostgreSQL. The Total Energy has increased for PostgreSQL despite reduction in Execution Time and this is due to increase in Average Power. The relative order of four engines w.r.t. Peak Power and Energy Efficiency for this 5-Query set is same for both serial and concurrent execution. We found similar results for other sets of queries tried as well.

Chapter 7

Temporal Behavior of Power Consumption

Since we have the logs of the instantaneous power consumption for the whole query execution at the granularity of one power sample per second, we plotted the temporal power graphs for all the queries on all the engines. All the temporal power graphs presented in this section plot the *dynamic* (above *ambient*) power consumption of the queries. We found some frequently occurring patterns and a few interesting power patterns with respect to appearance of Peak Power. We also tried to determine the database query processing operators in the plan of a query given by the optimizer responsible for such patterns. For the ease of mapping of operators in the plan of a query to patterns in the temporal power graph we generated plan trees of the optimal plans for the queries given by the optimizer of the engines. Picasso [1], a database query optimizer visualizer tool, was used to generate these plan trees. The patterns we found and the results of our investigation on the operators responsible for them are given below.

7.1 Fluctuating Pattern

In this pattern the power value fluctuates frequently within a small range of 10 Watts for the whole execution and it rarely goes out of this range. This is the most frequent pattern observed on SQL Server while very rare on the other three engines. The pattern is observed for 52 queries on SQL Server. Query 68 whose temporal power graph on SQL

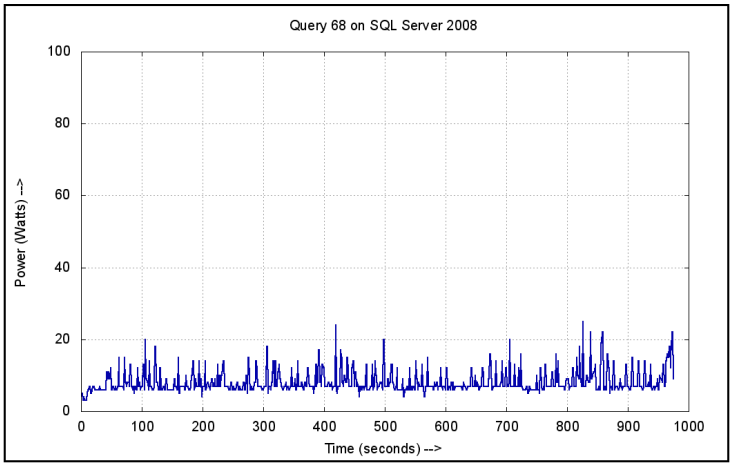


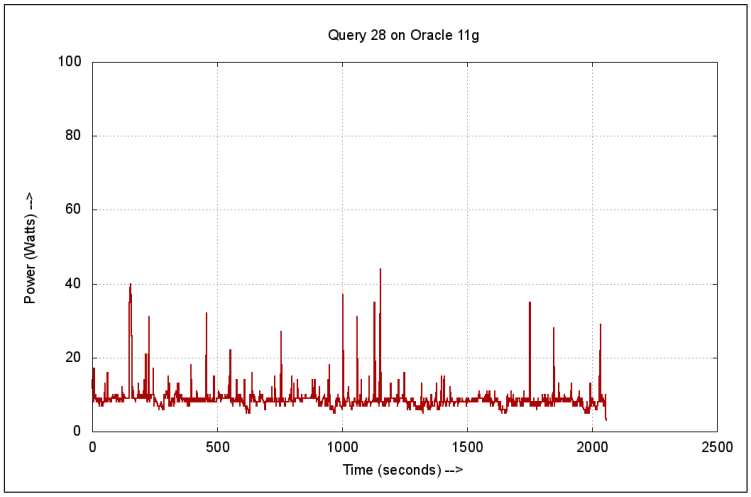
Figure 7.1: Fluctuating temporal power behavior for Query 68 on SQL Server

Server is shown in Figure 5 is a typical representative of such pattern. We could not however identify the reason for such behavior.

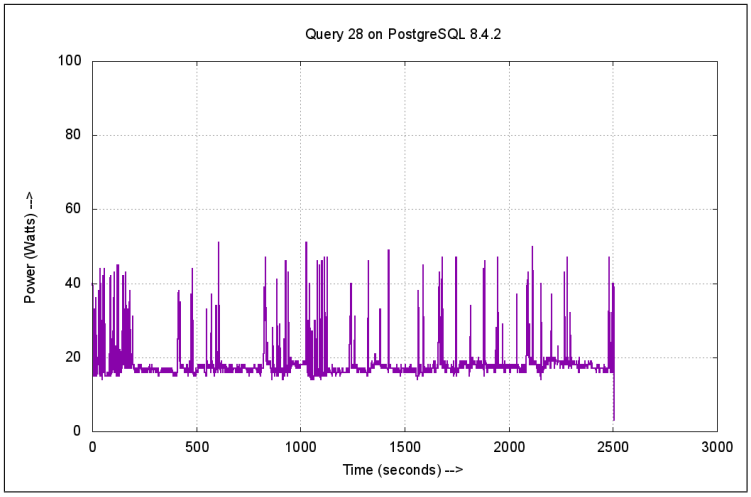
7.2 Big Power Jumps of Short Duration over Near Stable Base

In this pattern the power value typically ranges within a baseline power range of width 5W-10W and at irregular intervals there are big power jumps each of which is of a very short duration (one or two seconds). The Peak Power for the query can be any one of these jumps and can occur at any instant. This is the most frequent pattern found on Oracle, DB2 and PostgreSQL while not found on SQL Server. 52 queries on Oracle, 36 queries on DB2 and 42 queries on PostgreSQL show this pattern.

Figures 7.2a and 7.2b show this pattern for Query 28 on Oracle and PostgreSQL respectively. Since the jumps are for very short duration, it is difficult to identify what are the operators responsible for them. The plans for Query 28 on both Oracle and PostgreSQL contain many *sort* and *aggregate* operators and they are responsible for the jumps in power. The plans for many other queries which show this pattern contain lots of *nested loop join* operators in them and we think they are responsible for the sudden jumps.



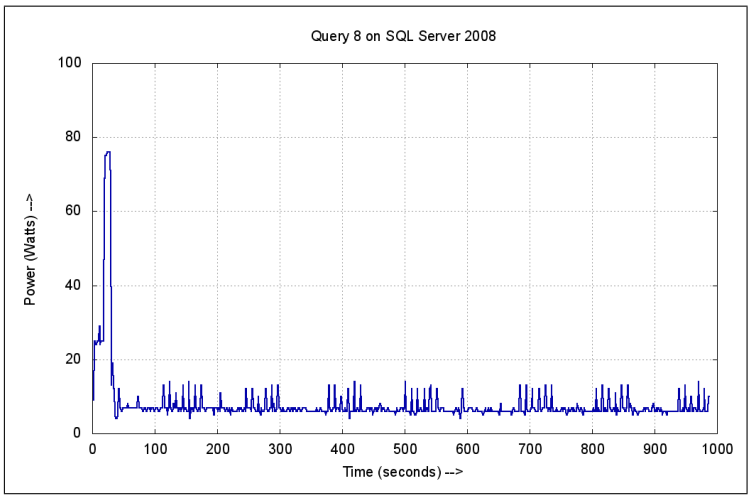
(a) Temporal Power Graph for Query 28 on Oracle



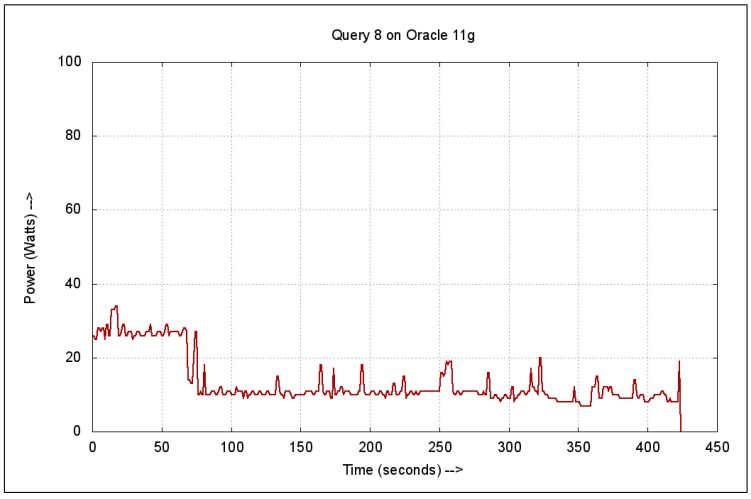
(b) Temporal Power Graph for Query 28 on PostgreSQL

Figure 7.2: Big Power jumps over a near stable base for Query 28 on Oracle and SQL Server

We couldn't identify exactly why the jumps in this pattern are for a very short duration, but they are due to sudden increase in CPU utilization for a short duration. Although there are many jumps, they are of very short duration. Hence if the CPU utilization is reduced at these instants, the time won't increase significantly but it will lead to Peak Power reduction. Thus these queries have opportunities for lowering Peak Power usage without sacrificing much performance.



(a) Temporal Power Graph for Query 8 on SQL Server



(b) Temporal Power Graph Query 8 on Oracle

Figure 7.3: Peak occurring towards the beginning of execution for Query 8 on SQL Server and Oracle

```
select ca_zip
from (

 (
  select substr(ca_zip,1,5) ca_zip
  from customer_address
  where substr(ca_zip,1,5) IN (
           '99148','14140','53002','21895',
           ..............................
           '49413','92304','61489','56839') //400 entries
 ) x

 intersect

 (
  select ca_zip
  from ( select substr(ca_zip,1,5) ca_zip,count(*) cnt
         from customer_address, customer
         where ca_address_sk = c_current_addr_sk
               and c_preferred_cust_flag='Y'
         group by ca_zip
         having count(*) > 10) A1
       )
 ) y

)
```

Figure 7.4: Subquery responsible for Peak in the beginning for Query 8

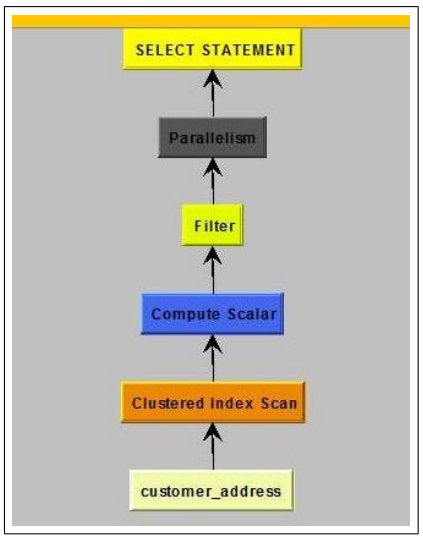


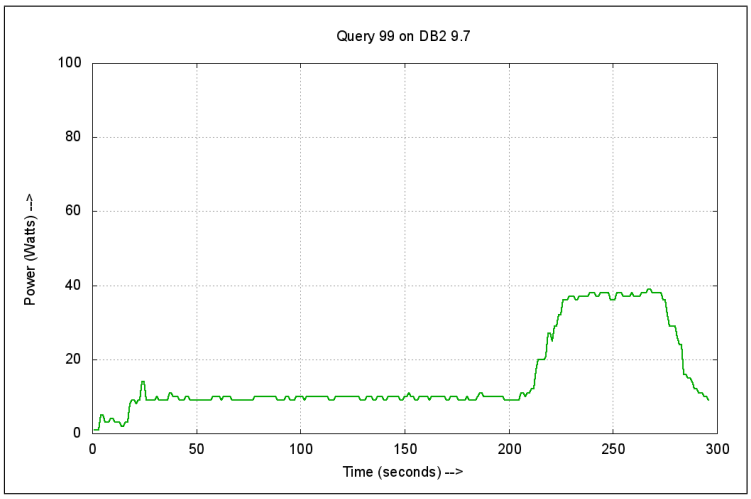
Figure 7.5: Plan for the subquery **x** (see Figure 7) in Query 8 on SQL Server responsible for 76W Peak

7.3 Peak towards the Beginning of Execution

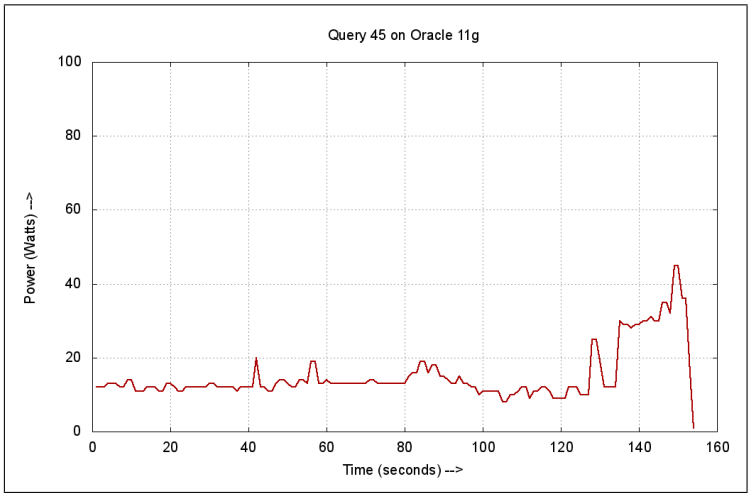
This is the pattern in which the Peak Power occurs for a longer (not for just 1-2 seconds) duration towards the beginning of execution and we are able to identify the operators responsible for it. This pattern occurs for 6 queries on SQL Server and one or two queries on other engines. Query 8 whose temporal power behavior for SQL Server and Oracle is shown in Figure 7.3a and 7.3b respectively, is the only one instance of a query which shows this behavior for all the four engines. The Peak Power towards the beginning of execution for Query 8 is due to a subquery of it shown in Figure 7. This subquery itself contains INTERSECT of two subqueries x and y. The 76W peak in Figure 7.3a for 10 seconds, which occurs after around 20 seconds from start of execution on SQL Server is due to subquery x in Figure 7. This was confirmed by executing only subquery x and plotting its power graph. The plan only for this subquery on SQL Server is shown in Figure 9. The *Filter* operator in this plan was responsible for the 76W peak. The Filter operator corresponds to the condition in *where* clause of subquery x in Figure 7. The temporary relation (colored red in the Figure) contains 400 entries. So for each tuple of CUSTOMER_ADDRESS the first five-letter substring of its CA_ZIP attribute is compared with the 400 entries in the temporary relation. This is a CPU intensive operation and the CPU utilization during the 10 second period of 76W peak, was found to be near 100%, which explains the high peak value. The 25W power consumption for first 70 seconds in Figure 7.3b on Oracle is due to the complete subquery in Figure 7. Unlike SQL Server there is no very high Peak due to subquery x. Like Oracle the complete subquery in Figure 7 is responsible for the Peak towards the beginning of execution for Query 8 DB2 and PostgreSQL as well.

7.4 Peak towards the End of Execution

For some queries we found the Peak occurring towards the end of the execution. We found this pattern for 12 queries on SQL Server, 11 on Oracle, 9 on DB2 and 4 on PostgreSQL. Query 99 on DB2 (Figure 7.6a) and Query 45 on Oracle (Figure 7.6b) are the examples of queries showing this pattern. The plan for Query 99 on DB2 is shown in Figure 11.



(a) Temporal Power Graph for Query 99 on DB2



(b) Temporal Power Graph for Query 45 on Oracle

Figure 7.6: Peak occurring towards the end of execution for Query 99 on DB2 and Query 45 on Oracle

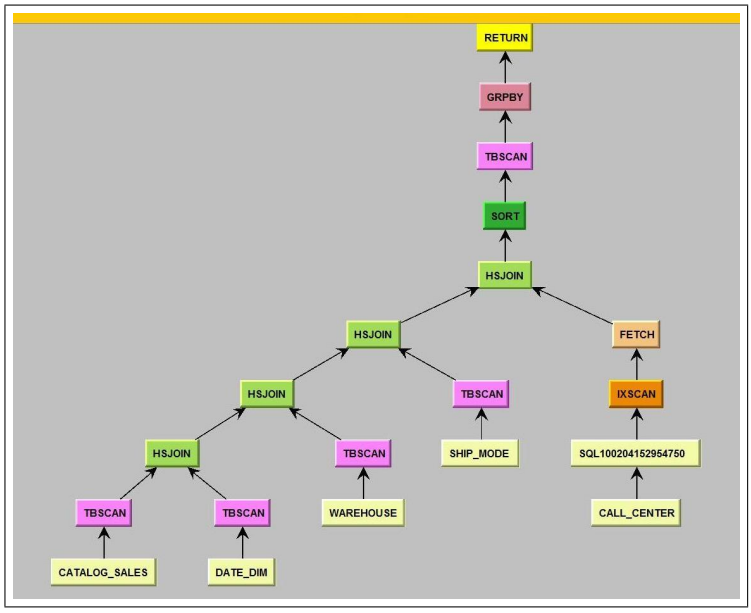
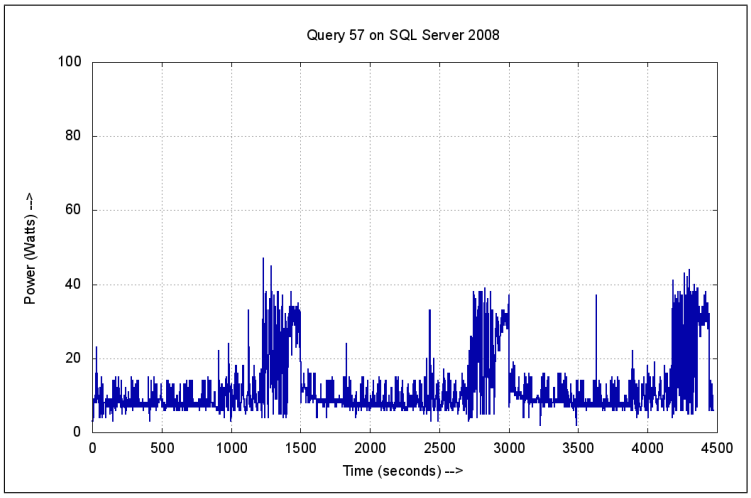


Figure 7.7: Plan for Query 99 on DB2

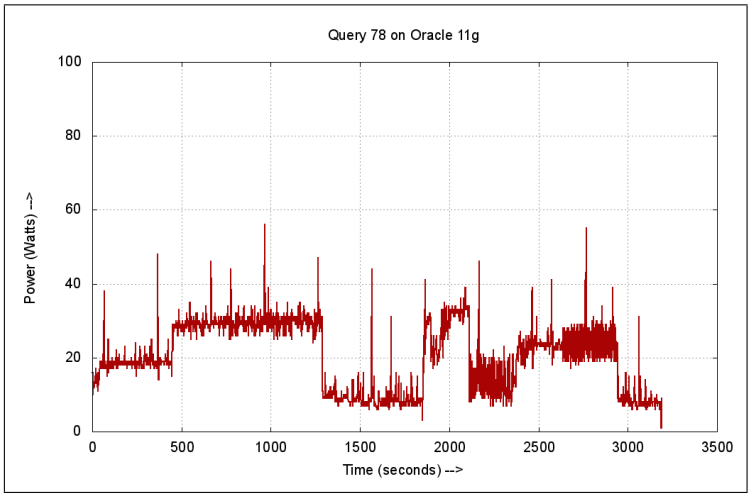
The *sort* operator in the plan is found to be responsible for the Peak. Even for Query 45 on Oracle, sort operator is responsible for the Peak in the end. This is due to CPU intensive nature of sort operator. However there are instances in which the there is no Peak towards the end of execution despite presence of sort operator. The input to the sort operator in such scenarios is of very small size and hence the CPU utilization is very low.

7.5 Pattern Repeating Similarly

This is very rare pattern in which there are three repeating instances of a similar power pattern. Query 57 on SQL Server (Figure 7.8a) and Query 78 on Oracle (Figure 7.8b) show this pattern. The pattern is periodic for Query 57 on SQL Server in Figure 7.8a as it repeats after every 1500 seconds. The plan for this query given by the optimizer reveals that there are three very similar subtrees which correspond to the three repeated patterns. We confirmed this by executing only one of the subtrees and obtained the pattern in first 1500 seconds of Figure 7.8a. The peak in the end of this 1500 second period is due to sort operator.



(a) Query 57 on SQL Server



(b) Query 78 on Oracle

Figure 7.8: Similar pattern repeating

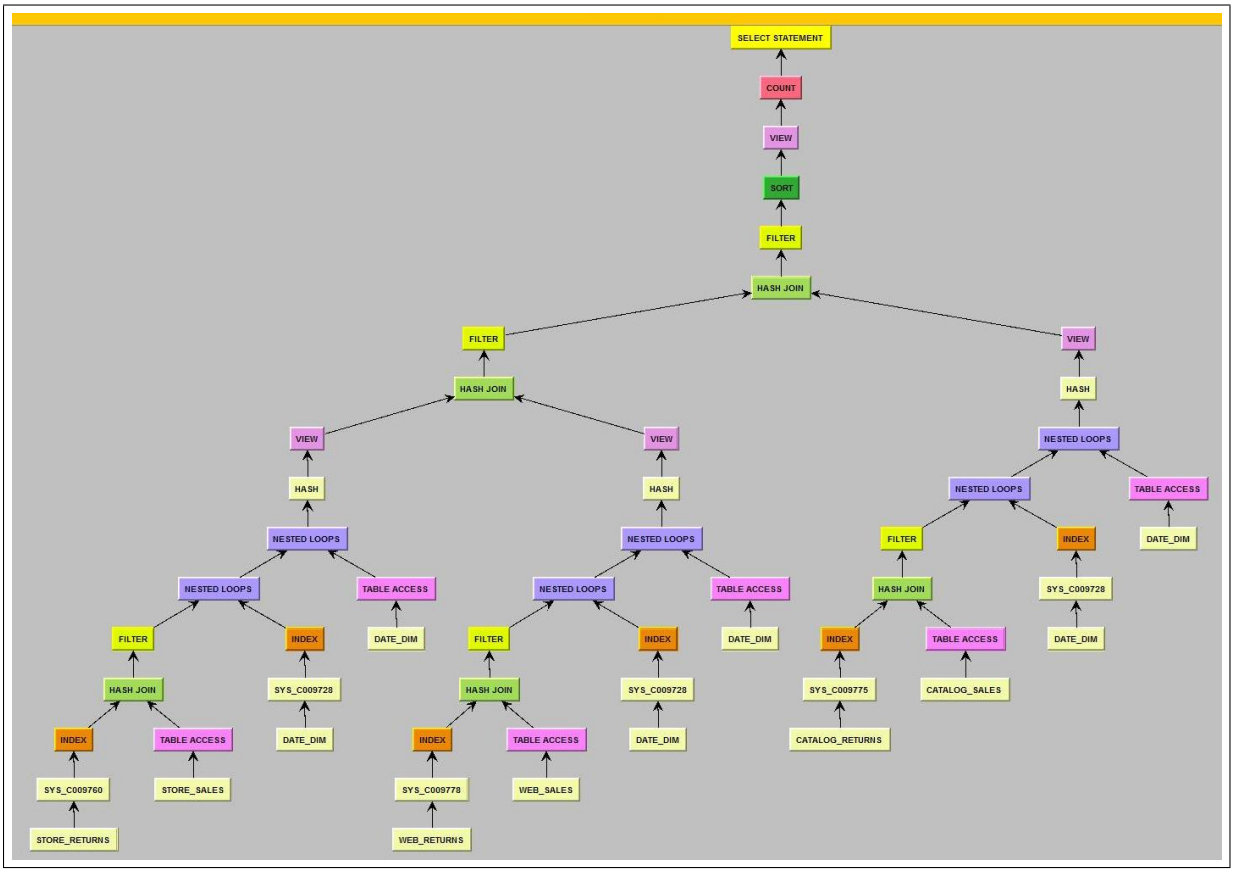


Figure 7.9: Plan for Query 78 on Oracle

The pattern for Query 78 on Oracle in Figure 7.8b is also due to similar reasons. The plan for this query shown in Figure 13 also has three similar subtrees and the three subtrees differed in two fact tables. The leftmost subtree contains STORE_RETURNS and STORE_SALES. The middle one has CATALOG_RETURNS and CATALOG_SALES in the corresponding places, while the rightmost subtree has WEB_RETURNS and WEB_SALES. The three patterns are similar but not of equal duration due to difference in them in terms of above fact tables.

Chapter 8

Summary and Future Work

In this work we compared the Peak Power and Energy consumption of four popular database engines viz. SQL Server, Oracle, DB2 and PostgreSQL on the TPC-DS decision support benchmark and ranked the engines with respect to Peak Power Efficiency and Energy Efficiency. The ranks with respect to Peak Power Efficiency are: DB2 first, Oracle second, PostgreSQL third and SQL Server fourth. The ranks with respect to Energy Efficiency are: DB2 first, SQL Server second, Oracle third and PostgreSQL fourth. DB2 is true winner with respect both Peak Power and Energy Efficiency. The other engines fare differently with respect to these metrics. Oracle which is the best engine with respect to performance comes third with respect to Energy Efficiency. SQL Server although is second best with respect to Energy Efficiency, is worst with respect to Peak Power Efficiency. PostgreSQL is third with respect to Peak Power Efficiency while worst with respect Energy Efficiency, thus it is not as good as the commercial engines.

We found that CPU power is the major component of dynamic power for our server and dynamic power is directly proportional to % CPU utilization. We analyzed the temporal power behavior of the TPC-DS benchmark queries on the four engines and found out some frequent and interesting power patterns. The major power consuming operators are found to be sort and aggregate. The other operators like nested loop also consume high power at some times. The operators take high power whenever they are able to highly utilize the CPU. Thus by carefully reducing peak CPU utilization, reduction in Peak Power can be potentially achieved.

In future we would like to do very detailed data mining based on temporal power behavior to determine the exact operators responsible for various peaks in the power graph and to determine the conditions under which these operators have high % CPU utilization and hence consume high power. This analysis can help in designing a peak power model for database operators and can be used carefully select the plans with low Peak Power but without sacrificing performance.

References

[1] Picasso Database Query Optimizer Visualizer, http://dsl.serc.iisc.ernet.in/PICASSO/picasso.html

[2] S. Rivoire, M. Shah, P. Ranganathan and C Kozyrakis. JouleSort: A Balanced Energy-Efficiency Benchmark. *Proc. of ACM SIGMOD* 2007

[3] M. Poess and R. Nambiar. Energy Cost, The Key Challenge of Today's Data Centers: A Power Consumption Analysis of TPC-C Results. *Proc. of VLDB* 2008.

[4] Z. Xu, Yi Tu and X. Wang. Exploring Power-Performance Tradeoffs in Database Systems.*Proc. of ICDE* 2010

[5] M. Poess, R. Nambiar and D. Walrath. Why You Should Run TPC-DS: A Workload Analysis

[6] D. Tsirogiannis, S. Harizopoulos and M. Shah. Analyzing the Energy Efficiency of a Database Server. *Proc. of ACM SIGMOD* 2010

[7] C. Xian, Le Cai and Y. Lu. Power Measurement of Software Programs on Computers With Multiple I/O Components. *IEEE Transactions on Instrumentation and Measurement* 2007

[8] D. Brooks, V. Tiwari and M. Martonosi. Wattch: A Framework for Architectural-Level Power Analysis and Optimizations. *Proc. of ACM ISCA* 2000

[9] W. Lloyd Bircher and Lizy K. John. Complete System Power Estimation: A Trickle-Down Approach Based on Performance Events. *Proc. of ISPASS* 2007.

[10] Isci and Martonosi. Runtime Power Monitoring in High-end Microprocessors: Methodology and Empirical Data. *Proc. of MICRO* 2003.

[11] Brand Electronics Power Meters.
http://www.brandelectronics.com/meters.html

[12] Watts up? PRO Power Meter.
http://www.wattsupmeters.com

[13] National Instruments Data Acquisition Cards, http://www.ni.com/dataacquisition

[14] SPECpower_ssj2008 Benchmark. http://www.spec.org/power_ssj2008

[15] The Transaction Processing Council.
http://www.tpc.org

[16] TPC-Energy Benchmark Development.
http://www.tpc.org/tpc_energy/default.asp

[17] The TPC-DS benchmark.
http://www.tpc.org/tpcds

[18] The TPC-C benchmark.
http://www.tpc.org/tpcc

[19] The TPC-H benchmark.
http://www.tpc.org/tpch