Generating Power-Efficient Query Execution Plan

Xiaowei Liu
Massive Data Computing Research Lab
Harbin Institute of Technology, HIT
Harbin, China
liuxiaoweilight@163.com

Jinbao Wang
Massive Data Computing Research Lab
Harbin Institute of Technology, HIT
Harbin, China
wangjinbaosky@gmail.com

Haijie Wang
Internet & Information Center
Harbin Institute of Technology, HIT
Harbin, China
1979.9.4WHJ@hit.edu.cn

Hong Gao
Massive Data Computing Research Lab
Harbin Institute of Technology, HIT
Harbin, China
honggao@hit.edu.cn

***Abstract*—As more and more power is consumed in servers, growing attention has been paid to power management. However, little work has been dated on the power-efficiency of query execution plans generated by the DBMS. Traditional optimal query execution plan does not take power-efficiency into account. This paper tries to generate a power-efficient query execution plan by redesigning the query optimizer of the database. We build a power model to estimate the power consumption of a query execution plan and propose an algorithm to generate the power-efficient query execution plan. Using two experiments based on the TPC-H benchmark, we evaluate the accuracy of our power model and compare the power-efficient query execution plan with the traditional optimal query execution plan.**

Keywords-query execution plan; power-efficiency; power model;

I. INTRODUCTION

In recent years, more and more energy is consumed in servers. Energy management technology has attracted the attention of both academia and industry. Data Centers in United States consumed 1.2 % of the annual energy in 2005, valuing 2.7 billion US dollars [1]. In 2007, the US Environmental Protection Agency (EPA) report showed that data center servers consumed 1.5% of the nation's electricity, up to 61 billion kwh, worth $ 4.5 billion in 2006 [2]. As many works suggest, energy costs in data centers will continue to increase [3, 4].

The growth trend of energy consumption in servers not only means the energy costs increase but also arouses other practical problems. One of these problems is power saving. Power being too high may lead to the breakdown of the blades in the data centers. The development trend of modern blades is to be faster and smaller. These trends bring challenges to the server cooling system. If the power consumption of the server is high, this will lead to the environment of the server temperature being high. Research shows that up to 50% of the hardware faults are aroused by the temperature of hardware being too high [5]. It is a promising way to control the temperature of the server by reducing the power consumption.

In a typical data center environment, the DBMS occupies most of the resources of the server. Meanwhile, the DBMS is dedicated most of the tasks in the data center. Most of the power is consumed by the DBMS. It is a promising way to research power saving strategy in the DBMS. As traditional query optimizer of the DBMS only concerns the time cost of a workload, but pays no attention to the power consumption. This gives us opportunity to redesign query optimizer to make it generate the power-efficient query execution plan.

To generate the power-efficient query execution plan, the query optimizer should be able to estimate the power consumption of a query execution plan. We first build a power model to estimate the power consumption of query execution plans. Then we introduce the possible query plan set which the power optimal query plan must be in and the algorithm to generate the possible query plan set. Then we can just search the possible query set for the power-efficient query plan.

Our main contribution is as follows:

1) A power model of query execution plan is designed in our paper. We build this power model to estimate the power consumption of query execution plan.
2) An algorithm to generate the power-efficient query plan is proposed in our paper.
3) Experiments based on the TPC-H benchmarks have proved the effectiveness of our algorithm.

The remainder of this paper is organized as follows. Related works are introduced in Section II. We introduce our framework in Section III; Section IV introduces our power model; Section V introduces our algorithms to generate power-efficient query execution plan. Our experimental evaluation is displayed in Section VI; Conclusions and future work are presented in Section VII.

II. RELATED WORK

Existing works to explore energy efficient computing can be concluded to two directions, including hardware-based approaches [9] and software-based approaches [6, 8, 9,

© 2013. The authors - Published by Atlantis Press

11]. We only talk about the software-based approaches as our work belongs to the software-based approach.

Harizopoulos et al. summarized three kinds of software-based approaches for reducing energy in data center [6]. The first type is energy-aware optimizations. Xu et al. proposed a strategy to estimate the power cost of query plans by estimating the power cost of each basic database operation in query processing [10, 11]. The second strategy is resource use consolidation. This strategy tried to reducing the numbers of powered up nodes by resource scheduling and data placement, so that the energy consumption of the data center can be reduced. J. Leverich et al. reduce the resource scheduling problem to the Covering Set (CS) problem in Hadoop clusters [7]. The third strategy is Redesign for Max Energy Efficiency. Lang el at. proposed the QED (Improved Query Energy-efficiency By Introducing Explicit Delays) strategy to save energy [8]. Xu el at. proposed a strategy by redesigning the query optimizer because existing query optimizer only concerns the time cost of a query plan but not concerns the power cost [11].

III. FRAMEWORK

For traditional performance-driven query optimizers, time cost is the first reference of a query execution plan. The best query execution plan is the query plan that costs the least time. But if we concern both power and time, the time optimal query plan maybe not best in power consumption. The time cost of a query plan is the summation of the holding times of all system resources including CPU, disks, and communication channels [11]. As modern hardware develops trend, the CPU processing speed is much higher than the speed of I/O. Compared to I/O time, the CPU time of a query plan can be ignored in most cases. When query optimizer calculates the query plan time cost, it only concerns the I/O time. The optimal query execution query plan is a query plan with the smallest I/O operations. However, the power consumption of CPU is much greater than that of the storage systems [11]. Concerning power-efficiency, traditional optimal query execution may be not the best query plan.

We expect that the query optimizer can generate a power optimal query plan. To do this, the query optimizer should have knowledge about the power consumption of the query plan. As we know, traditional query optimizer cannot predict the power consumption of query execution plans. We have to build a power model to estimate the power consumption of query execution plan. When we have built a power model of query execution plan, we could estimate the power consumption of any query plan.

To get the power-efficient query plan, we should search the query plan space. For any query plan in the query plan space, we can estimate its time cost and power consumption. We can compare the power consumption of these query plans. Then we can find the power-efficient query plan. But there are many redundancy query plans in the query plan space. In order to reduce redundant queries, we introduce the definition of the possible query plan set in section IV. To get the power-efficient query execution, we first generate the possible query plan set. Then we can just search the possible query plan set for the power-efficient query execution plan. We summarize the process to get the power-efficient query execution in figure 1.

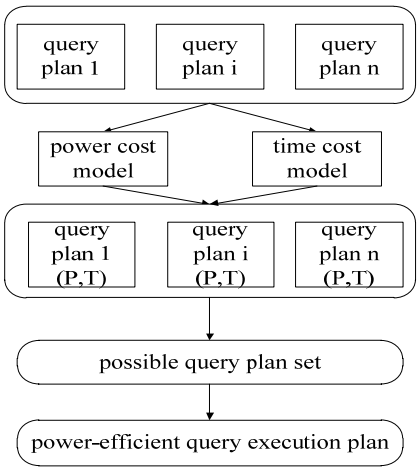


Figure 1 the process to get power-efficient query execution plan

IV. POWER MODEL

To estimate the power cost of a query execution plan, we have to build a power cost model. Different from the method by measuring power consumption of the whole system in [12], we measure the power consumption of the various components of the system. We measure the power consumption of CPU, memory and disk separately. So we can have detailed knowledge about the power consumption of CPU, memory and disk components. As the power consumption of memory component changes little, we regard the power consumption of the memory component to be a constant. We mainly measure the power consumption of CPU and disk. We use the PM 1000+ to measure the power consumption of each component.

We build the power model by imitating that the database builds the time model. The process that how we get the power model is as follows. For a specific query execution plan, we are able to get the basic operations that the query execution plan contains. We can also record power consumption of the query plan using the PM 1000+ machine; we design query plans that only obtain simple basic query operations. We then run these query plans along with measuring the power consumption of these query plans with different data sets. We train our power model to be accurate enough to estimate the power consumption of the query plans. We then design more complex query plans that are composed of basic query operations. We measure the power consumption of these query plans with different size data sets. We train our power model until it is accurate enough to estimate the power consumption of any query plans.

The power consumption of any query plan can be calculated because that each query plan is consist of the basic operations. Our main task is to build power model of these basic operations. We train our power model with those designed query plans. To get more accurate power model,

we train our power model with different data sets. We update our power model when the power consumption which our power model estimates differs much from the power consumption which the query plan actually cost. We will train our power model until we are satisfied with the error between power which the power model estimates and the power which the machine measures.

Table I basic operation power cost function

Operation	Power Cost Function
Seq Scan	$F_1(cT)+f_1(N)+C$
Index Scan	$F_2(cT)+f_2(N)+C$
Bitmap Scan	$n（F_2(cT)+f_2(N)）+C$
Sort	$F_3(cT)+f_3(N)+C$

The basic query operations that we measure include Seq Scan, Index Scan, Bitmap Scan and so on. They are shown in table I. In table I, we show the power cost function of basic query plans. T stands for the number of tuples to retrieve; c stand for the number of column to retrieve; N is the pages of data which need to read from the disk; Function F_i stands for the power consumption of CPU; Function f_i stands for the power consumption of disk; All C is a constant and it stands for the power consumption when the system is idle.

V. QUERY EXECUTION PLAN GENERATION

The traditional DBMS query optimizer only generates the time optimal query plan. We need to redesign the query optimizer if we want to get a power-efficient query plan. As shown before, the lowest time cost query plan maybe not the lowest power consumption query plan. We expect to get power-efficient consumption query plan. To do this, we introduce some definitions.

Definition 1: query plan cost: if time cost of query plan A is T, power consumption is P. then the cost of query plan A is defined as (P_A, T_A).

Definition 2: query plan A dominate query plan B: query plan A and query plan B are different query plans for one query task. If the cost of query plan A is (P_A, T_A), and the cost of query plan B is (P_B,T_B).We define query plan A dominate query plan B if $P_A<=P_B$ and $T_A<=T_B$;

If query plan A dominate query plan B, then the query optimizer will choose query plan A to execute the query task because query plan A is better than query plan B both in time cost and power consumption. For the same reason, the optimal query plan generated by the query optimizer for query task should not be dominated by any query plans for the same query task. We introduced the possible query plan set for a query task.

Definition 3: possible query plan set for a query task: If query plan A is not dominated by any other query plan in the query plan space, query plan A is a possible query plan to execute the query task. We call all the possible query plans for a query task the possible query plan set.

The possible query plan set for a query task stand for the query plans those are not dominated by any other query plans in the query plan space. We introduce definition 3 to get the optimal query plan in the query plan space.

Theorem 1: The optimal query plan must be in possible query plan set.

We can easily prove theorem 2 by contradiction. To get the optimal query plan, we can first get the possible query plan set. Then we can search the possible query plan set for the optimal query plan. We concentrate on redesign the query optimizer to generate the possible query plan set. To generate the possible query plan set by the query optimizer, we should take both the time cost and power consumption into account. For a query node in the query logical tree, there exceeds one physical way to execute the query node in most cases. For different physical ways to execute the query node, the time cost and power consumption are different too. Assume that there are two physical ways to execute the query node, including physical way A and physical way B. The time cost of physical way A is t_A and the power consumption is p_A, we say the cost of physical A is (p_A, t_A). The cost of physical way B is (p_B, t_B). If physical way A dominates physical way B, the query optimizer will prefer to choose the physical way A to execute the logical node.

Theorem 2: Query plans in the possible query plan set must be composed of physical query ways that are not dominated by other physical query ways.

We can also easily prove theorem 2 by contradiction. Theorem 2 reveals us a way to generate the possible query plan set. For each query node in the logical query tree, the query optimizer can just generate the physical ways that are not dominated by other physical ways. After each node in logical query node has been generated, we get a superset of the possible query plan set. Detailed ways are displayed in the following algorithm.

Algorithm power optimal query execution plan generation
Input: the logical query plan tree
Output: the power optimal query plan

```
1: Initialize the possible query plan set M
2:    for each query node in the logical query plan tree
3:       Initialize the possible logical way set N
4:       for each physical way of the query node
5:          estimate (p,t)
6:          for each (p',t') in N
7:             if (p,t) dominate (p',t') then
8:                update N'
9:  get the (P,T) of these query plan
10:    for each (P',T') in M
11:       if (P,T)  dominate (P',T')
12:          update M
13: Search M for the power optimal query plan J
```

The query optimization work is to get the optimal physical query plan from the logical query tree. To get the optimal query plan, traditional query optimizer has to search from the query plan space to get the best query plan. But if we can first get the possible query plan set, then we can just search the possible query plan set for the optimal query plan.

The possible query plan set is smaller than the query plan space because query plans in possible query plan set are query plans that cannot be dominated by other query plans. So we can find the optimal query plan faster.

VI. Experimental Evaluation

The purposes of our experiments are to test the accuracy of our power model and to verify the effectiveness of the algorithm to generate power-efficient query execution plan. We run all our experiments in a single database server. The system has the following main components: Intel(R) Core(TM) i5-2400 CPU, 2*2GB Kingston main memory, and a WDC WD5000 500G disk. The DBMS is the open source PostgreSQL 9.1.8, and the Linux is the Ubuntu 12.04 LTR. To avoid other applications affecting the load of CPU and I/O equipment, we only run the experiment workload in our system. The power consumption of our experiments is recorded by the PM 1000$^+$, running on its highest sampling frequency –one hertz. We run the TPC-H workloads in our experiments. Our experiments are as follows.

Experiment 1. Evaluate our power model in different database sizes running different TPC-H query workloads. We train the power model to be accurate enough to estimate the power consumption of our query execution plan.

The results of experiment 1 are shown in figure 2. We can conclude that the accuracy of our power model have negative correlation with the database size from figure 2. Because when the size of database becomes bigger, the error of the power model will be amplified. The difference of our power model varies form 5.2% to 12.1%, which is better than 14.5% reported in [12].

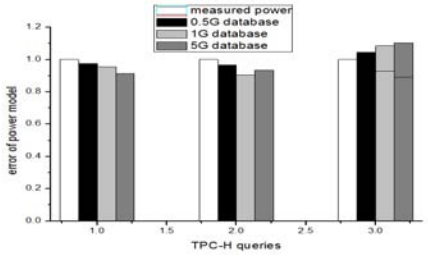


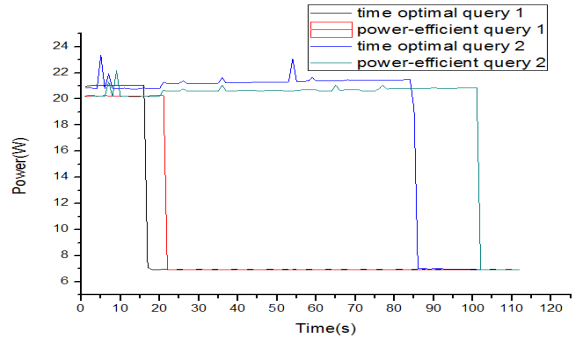
Figure 2 The error of our power model

Experiment 2. We redesign the query optimizer of the PostgreSQL to generate the power-efficient query execution plan. To evaluate the effectiveness of our algorithm, we run the algorithm under different database size with different TPC-H workloads. The results are shown in figure 3.

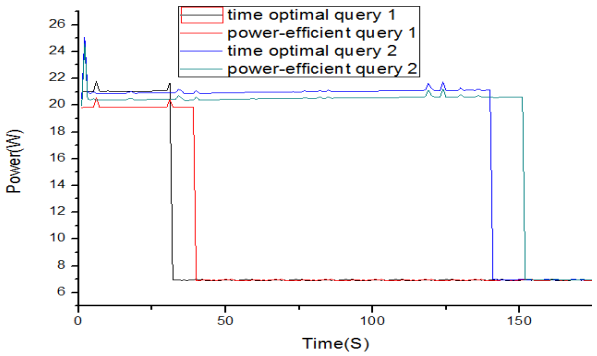
The x-axis in figure 3 represents the time cost of the query execution plan; the y-axis represents the power consumption of the query execution plan. We compare the time optimal query execution plan with the power-efficient query execution plan. We can see that there usually a tradeoff between the time cost and the power consumption. To get the power-efficient query execution plan, we have to bear the increase of the execution time. The results of task 2 prove the effectiveness of our algorithm to generate the power-efficient query execution plan.

VII. Conclusions and Future Work

In this paper, we explore the power-efficient query execution plan in current DBMS system. We first build a power model of query execution plan, then we propose an algorithm to generate power-efficient query execution plan. Experiments in section VI have proven the accuracy of our power model and the effectiveness of our algorithm. We can conclude that the power-efficient query execution plan really exists.



a 500M database



b 1G database

Figure 3 Comparision between power-efficient query plan and traditional query plan

As little attention has been paid to the power-efficient query execution plan, much future works can be done in this area. One of these is to build a dynamic power model to estimate the power consumption of query plans instead of static power model.

References

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

[2] J. Wang, L. Feng, W. Xue, and Z. Song. A survey on energy-efficient data management. SIGMOD Rec. 40(2):17–23, Sept. 2011.

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

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

[5] L.-T. Yeh and R. C. Chu. Thermal Management of Microelectronic Equipment: Heat Transfer Theory, Analysis Methods, and Design

Practices. ASME Press, 2002.

[6] S. Harizopoulos, M. Shah, J. Meza, and P. Ranganathan. Energy efficiency: The new holy grail of data management systems research. In CIDR, 2009.

[7] J. Leverich and C. Kozyrakis. On the energy (in) efficiency of hadoop clusters. SIGOPS Oper. Syst. Rev., 44(1):61–65, 2010.

[8] W. Lang and J. Patel. Towards eco-friendly database management systems. In CIDR , 2009.

[9] D. Tsirogiannis, S. Harizopoulos, and M. Shah. Analyzing the energy efficiency of a database server. In SIGMOD, pages 231–242. ACM, 2010.

[10] Z. Xu. Building a power-aware database management system. In IDAR, pages 1–6. ACM, 2010.

[11] Z. Xu, Y. Tu, and X. Wang. Exploring Power-Performance Tradeoff s in Database Systems. In ICDE, pages 485–496. IEEE, 2010.