Some Heuristic Approaches for Reducing Energy Consumption on Database Systems

Miguel Guimarães
ALGORITMI R&D Center
University of Minho
Portugal
pg22800@alunos.uminho.pt

João Saraiva
HasLab R&D Center
INESC/University of Minho
Portugal
jas@di.uminho.pt

Orlando Belo
ALGORITMI R&D Center
University of Minho
Portugal
obelo@di.uminho.pt

Abstract—**Today, one of the major concerns of administrators and managers of data centers is related with the cost of the energy that each database component consumes when involved in activities and processes they manage. In fact, it is not necessary to conduct a detailed assessment to realize that the cost of energy consumed in this type of systems is really great. So, it is not surprising the significant growing interest that researchers have in this domain. Various techniques have been developed to assess the energy consumption on database systems, demonstrating their utility in managing the power they use to consume. Basically, they come to confirm the paradigm shift on the issue of energy concern in database systems towards the reduction of its consumption. In this paper, we present and discuss a set of heuristics that we suggest to reduce, in particular, the energy consumption on the execution of a given query inside a relational database system. With this work, we intend to contribute to design and implement more efficient queries in terms of energy, i.e., green queries, based on the analysis of the various components that are used in their physical implementation, reducing as much as possible their energy consumption, taking into consideration the characteristics of the database operators used and the querying execution plans established for them.**

Keywords-data centers; database management systems; querying execution plans, database queries consumption, green queries; consumption heuristics.

I. INTRODUCTION

The assessment of energy consumption of any component is not an easy task. To carry out appropriately this kind of evaluation, it is necessary to study in detail how the component behaves and how it is used in practice. The same may be applied to the study of power consumption of a *DataBase Management System* (DBMS) or, in particular, of any query that can be performed in its environment [1]. Any process for the establishment of energy-efficient queries, usually recognized as green queries, and not its optimization in terms of processing time or usage of computing resources, requires a fairly deep knowledge of how queries are processed and optimized in the environment of a DBMS.

Database querying processing [2][3] is one of the most important activities of a DBMS, which involves a well-defined set of processes for supporting the way the system responds to users' queries. Optimizing querying processing is something that has been worked over the years by researchers in the databases field. See, for example, the work of Ceri and Gottlob [4] that presented a way to transform a SQL statement into relational algebra expressions representing equivalent SQL sub statements, a method revealed by Taniar [5] especially oriented to add additional instructions (e.g., optimizer hints, access method hints, or table joins hints) into SQL statements to instruct the SQL optimizer for executing the statement in an alternative better way, or how Li et al. [6] suggested a manner to improve querying performance with configuration options – e.g., table partitioning, materialized views, or storing plan outlines – increasing as well the efficiency of the code application.

However, as the number and capacity of data centers increases, beside the so usual issues of performance and querying processing abilities – always critical aspects for any DBMS –, the issue of power consumption is appearing very clear in their cost operational reports, year after year [7]. This caught the attention of data centers' managers all over the world up sharply their concerns related to energy consumption, not only because of the cost of electricity itself but also because of relevant environmental issues. In general, database servers are the biggest customers of computational resources of a conventional data center, which makes them also one of its biggest energy consumers [8]. Although these systems are very well equipped today, with powerful tools for querying optimization, quality of service, or overall performance, usually in terms of energy consumption DBMS do not have any means especially oriented to the management and control of consumption power. With the current trends and needs of the markets and DBMS users, this lack is considered quite serious. The non-availability of data about the energy consumption of a DBMS has been motivating a large diversity of research initiatives aiming to create means so that we can in addition

 ISBN: 978-1-61208-486-2

to defining query optimization plans in DBMS environments also create energy consumption plans for executing queries.

In a previous work [9], we developed a study that allowed us to develop an energy consumption plan for a query that is usually executed in a conventional data center environment. At that time, the goal was reducing, as much as possible, the energy consumption of data centers queries without affecting their usual performance. We believed that such small reduction in the consumption of a simple query could be a great help in reducing the overall consumption of a data center – that was proved easily by multiplying these tiny savings by the huge number of queries (and transactions) that are executed per minute in a data center. Continuing this work, we studied a set of specific heuristics that we suggest to reduce, in particular, the energy consumption on the execution of a given query inside a relational DBMS, which we expect that contribute to design and implement more efficient queries in terms of energy – green queries.

In this paper, we present and discuss the referred energy consumption heuristics, giving particular attention to other related works (Section II), showing how we categorized the energy consumption of a SQL query, what kind of transformation rules were applied, and how the energy was consumed by each one of the transformation rules studied (Section III). Finally, we finish this paper with some brief conclusions and pointing out some future research lines (Section V).

II. Related Work

Today, energy efficiency is a trend topic in terms of researching and development. Researchers of different fields of expertise work and discuss possible solutions to solve the energy crisis that we are facing today. On the one hand, saving energy allow us to reduce the energy billing costs; and on the other hand, by doing that, we are also preserving environment resources. Thus, saving energy and creating policies to develop green software is beneficial for everybody. Steps towards that direction have been already made. At software level, for instance, the works presented in [10] and [11] are some good examples of techniques and methods used to detect energy consumption. In [10], the authors adapted a technique known as Spectrum-based Fault Localization to identify parts of code responsible for a higher energetic consumption. The work done in [11] focused more on finding and detecting anomalous energy consumption in Android systems. Both works show us that reducing energy consumption has been tackled in a variety of systems by several researches, and its popularity in computer science domains has increased.

Database systems have also taken small steps towards green guidelines. The Claremont report was one of the first approaches concerning energy consumption in database systems [12]. The main goal of this report was to take into consideration, during the devise and implementation stages of a database system, the energy consumed by different tasks. Reinforcing such concerns, the work presented in [13] provided us a clear survey of how to control efficiently energy in data management operations. Later, other studies emerged approaching the same topic [14][15][16]. However, most of them have focus essentially on hardware questions. In terms of software, in [9] it was redesigned the execution plan of a DBMS in order to include, not only the default estimative values for query execution, but also an estimative of energy that will be consumed to run a specific query. Later, it was proposed a solution to redesign a DMBS kernel, in order to reduce energy consumption [15]. Afterward, in [17] other alternatives were suggested to reduce the high levels of energy consumption in DBMS, in general terms, while other works, were concerned about the prediction of the consumption of large join queries [18], or how to optimize queries to reduce global consumption of energy within a DBMS [19]. However, as far as we know, there are no works approaching the effect of regular querying optimization heuristics on the consumption of energy of a DBMS. Thus, we selected some of the most used heuristics on relational querying processing and studied their effect in terms of energy consumption.

III. Energy Consumption Categorization

The energy consumption paradigm has been increasing its importance over the last few years, slowly replacing the performance paradigm, in terms of main concerns, to take into account when developing any kind of database querying task. Query processing is one of the most important activities performed by a DBMS. Today, it is possible to analyze and optimize the cost of a database query in terms of performance, establishing better execution plans and reducing querying processing time. Having access to these plans, we can also measure the energy that database operations consume in a similar way as we can measure their processing time. We only need appropriate tools.

A. Data and Test Configuration

In this work, we developed a tool with the ability to measure the energy consumption of SQL queries, categorizing which ones are green and which ones are not. We used the tool *gSQL*, shorten for *greenSQL*, to categorize the energy consumption of SQL queries. This tool uses as support the jRAPL framework, which allows for monitoring the energy consumption of different hardware levels for a certain code block [20]. In order to use jRAPL, there are certain conditions that must be fulfilled. The processor has to be from Intel architectures and support *Machine-Specific Registers* (MSR). The later are the registers used for storing the energy consumption information for code block that was monitored. Therefore, the role played by the jRAPL framework is exclusively dedicated to categorize and analyze the energy consumption of SQL queries. Regarding the *gSQL* tool, even though its use is simple, it gives us the required information to devise hypothesis and thus create

heuristics to reduce the queries energy consumption. To run the tests it is necessary to specify three different parameters:

1) an input file, with all the queries that going to be tested;

2) the number of times each query will be repeated;

3) the number of times each test will be repeated.

In Figure 1, it is possible to see a brief description of the overall behavior of the *gSQL* tool, written in pseudo code. After all the tests have been executed, we obtain as result the energy and the time consumption for each query tested. The calculated aggregated values were maximum, minimum, average, and standard deviation. This set of aggregated values allows for us to find out if a certain test need to be executed again, by analyzing the standard deviation as well as the amplitude between maximum and minimum values. For the test environment, we choose the PostgreSQL DBMS, populated with data based on the TPC-H benchmark. Depending on the scaling factor, the database can assume different dimensions. In this case study, we used a scale factor of two, which means that we multiplied by two the cardinality of the tables that depends from the scale factor.

```
begin
resultsList ← initializeResults()
  for each query in queriesList do
    begin
      for each execution in executionsList do
      being
        for each repetition in repetitionList do
        begin
          initialEnergy ← getEnergy()
          initialTime   ← getTime ()
          executeQuery(query)
          finalEnergy ← getEnergy()
          finalTime ← getTime()
          energy ← initialEnergy - finalEnergy
          time ← finalTime - initialTime
          storeValues(resultsList, energy, time)
        end
      end
    end
    aggregate ← aggregateResults(results)
    writeFile(aggregate)
  end
```

Figure 1. A pseudo code excerpt describing the behavior of the gSQL tool

B. Transformation Rules

Often, we start a querying optimization process by analyzing the structure of the query, trying to see if it is well designed and use the most appropriated resources. In this kind of processes, it is common to see if some practical querying heuristics can be applied at a certain stage of the process, in order to improve the way the query is processed, having the goal to reduce its execution time. There are a set of heuristics well establish in the literature to improve querying processes [21]. In some particular application cases, such heuristics give us clear processing advantages, reducing the resources involved with and the response time of the query. The question now is: do those querying heuristics also help in reducing querying energy consumption?

The transformation rules used for the relational algebra operations suit well the requirements presented in [22] and posteriorly in [21]. The first six rules were tested using the *gSQL* tool. The results we got were analyzed in order to create the heuristics to optimize querying energy consumption. Each transformation rule (1-6) that was used will be explained and illustrated with a specific SQL query example. The SQL queries were devised specifically for each transformation rule. Due to the variety of tables in the TPC-H benchmark, there are plenty of options that could be used to design queries for each different transformation rule. However, TPC-H benchmark has a set of queries which, in this case study, were adapted to better represent transformation rules. Results can be consulted later in Table 1.

Transformation Rule 1 – this rule states that conjunctive selection operations can be separated into individual selection operations. To demonstrate this transformation rule, we create two SQL queries that are presented in Figure 2.

```
a)  select * from lineitem where l_quantity>40
        and l_discount>0.03;
b)  select * from ( select * from lineitem
        where l_discount>0.03) as sub
        where sub.l_quantity>40;
```

Figure 2. The SQL queries for testing rule 1.

The first query (Figure 2a) represents the conjunctive selection operations whereas the second one (Figure 2b) represents individual selections with the application of some filtering conditions.

Transformation Rule 2 – this second rule shows us how the selection operations have commutative proprieties. This means that doing the selection of a given predicate *p* followed by a predicate *q* have the same result as doing first the selection of the predicate *q* followed by the predicate *p* (Figure 3).

```
a)  select * from ( select * from lineitem
       where l_discount>0.03) as sub
       where sub.l_quantity>40;
b)  select * from ( select * from lineitem
       where l_quantity>40) as sub
       where sub.l_discount>0.03;
```

Figure 3. The SQL queries for testing rule 2.

Transformation Rule 3 – this rule denotes that in any sequence of projection operations, only the last one in the sequence is necessary. For instance, doing in first place the projection of the attributes *a* and *b* followed by *b* is equivalent of doing only the projection of the attribute *b* (Figure 4).

```
a) select sub.l_shipmode
      from ( select l_quantity, l_discount,
      l_shipmode from lineitem) as sub;
b) select l_shipmode
      from lineitem;
```

Figure 4. The SQL queries for testing rule 3.

 ISBN: 978-1-61208-486-2

Transformation Rule 4 – this rule states that between selection and projection operations there is a commutative propriety associated as long as the predicate belongs to the attributes in the projection list (Figure 5).

```
a) select sub.l_shipmode, sub.l_quantity
     from ( select * from lineitem
     where l_quantity>40) as sub;

b) select sub.*
     from ( select l_shipmode, l_quantity
            from lineitem) as sub
     where sub.l_quantity>40;
```

Figure 5. The SQL queries for testing rule 4.

Transformation Rule 5 – according to this rule, a cartesian product and theta join operations can be commuted. Therefore, doing a theta join between two relations, *R* and *S*, it is equivalent to do the theta join between the relation *S* and the relation *R*. The same principle can be applied to the cartesian product as well as to a natural join or an equijoin (Figure 6). For this example, we set a limit of five hundred thousand records to be selected, in order to have a faster query. Without the limit constraint, the difference between energy consumptions of both queries will be the same, but limiting the number of records to be selected, instead of the full length of the relation, allows us to save time when running tests.

```
a) select * from orders
     inner join customer
     on o_custkey = c_custkey limit 500000;

b) select * from customer
     inner join orders
     on o_custkey = c_custkey limit 500000;
```

Figure 6. The SQL queries for testing rule 5.

Transformation Rule 6 – Rule number six states that between selection and theta join operations there is a commutative propriety associated, if the selection predicate involves only attributes of one of the relations being joined (Figure 7).

```
a)   select sub.* from (select * from orders
     inner join customer
     on o_custkey = c_custkey) as sub
     where sub.c_mktsegment='BUILDING'
            and sub.o_orderpriority='2-HIGH';

b)   select * from
     (select * from customer
         where c_mktsegment='BUILDING') as t1
         inner join (select * from orders
         where o_orderpriority='2-HIGH') as t2
            on t1.c_custkey = t2.o_custkey;
```

Figure 7. The SQL queries for testing rule 5.

C. Result Analyzes

Through the data presented in Table 1 it is possible to analyze the average energy consumed for each transformation rule and take some conclusions regarding the heuristics for optimizing the energy consumption on SQL queries. If we observe rule 2, we can see that the second query (Figure 3b) consumes less energy than the first query (Figure 3a), because the second SQL query reduces the number of tuples that are processed by the DBMS. Thus, we can infer that doing first the selection operation that discards more tuples translates into an energy saving measure. Another conclusion that can be deduced based on the *gSQL* results, is that doing only the projection of necessary attributes consumes less energy than the projection of necessary and unnecessary attributes (transformation rule 3). An obvious conclusion, since there is less computational load required, a less execution time leads to a decrease in the energy consumption. Regarding the transformation rule 4, it is possible to conclude that reducing the cardinality of the relations it is a good green practice. Hence, eliminating unnecessary tuples before doing others relational algebra operations can be seen as a heuristic to optimize the energy consumption. With the data collected from *gSQL* tool, for cartesian products and theta joins operations, we can inferred the following: if the relation on the left side of the join operation have higher cardinality than the relation on the right side, then it consumes less energy (transformation rule 5).

TABLE I. AVERAGE ENERGY COMSUMPTION FOR EACH QUERY IN A TRANSFORMATION RULE

Query	Average Energy (Joules)	Average Time (seconds)
1 a)	251.3028	12.34
1 b)	256.1062	12.5102
2 a)	256.1062	12.5102
2 b)	249.4447	12.16
3 a)	196.2550	9.86
3 b)	195.3628	9.72
4 a)	78.74333	3.98
4 b)	80.18314	4.02
5 a)	77.5720	3.8
5 b)	78.17531	3.72
6 c)	21.04923	1.04
6 d)	21.92778	1.12

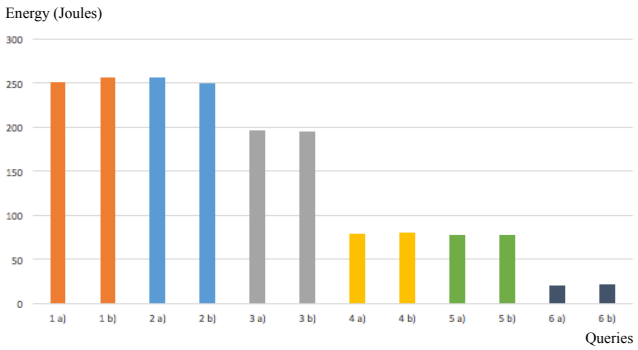

Figure 8. Energy consumed by each transformation rule version

Lastly, data from transformation rule 6 suggests that it is greener to do the selection operation before doing theta-join operations. As previously mentioned, cardinality reduction means less energy consumption. Finally, in Figure 8 we can

see a chart showing the energy consumption of each one of the queries that were used and tested in each transformation rule.

IV. CONCLUSIONS AND FUTURE WORK

In this paper, we presented some heuristics to optimize the energy consumption of relational database queries. The heuristics presented here were devised by analyzing the data from *gSQL* tool. This was a first approach towards the creation of a more refined set of energy consumption heuristics. As far as we know, nothing similar was done yet in this field of expertise. Saving energy on a query that is executed several times in a data center reduces the monthly energy bill and therefore, decreases the costs of a data center. It is interesting to notice different energy consumptions by doing simple tweaks and transformation rules. Although, we present some heuristics in this paper, some of them having a parallel that corresponds to performance optimization heuristics well defined in the area database systems. Hence, for systems with the same type of hardware optimizing a query to be greener is equivalent to optimizing a query to be faster.

In a near future, we intend to verify if the heuristics proposed here can be transposed to different DBMS. It is important to know how to rank the different DBMS, in order to offer to database administrators the possibility to adopt an eco-friendlier DBMS to support their operational systems. Another issue that we expect to explore is the impact of the established heuristics in DBMS performance structures, such as indexes, execution plans or materialized views, in order to prepare DBMS internal configuration structures regarding energy saving issues.

REFERENCES

[1] G. Graefe, "Database servers tailored to improve energy efficiency," in Proceedings of the 2008 EDBT Workshop on Software Engineering for Tailor-made Data Management, 2008, pp. 24-28.

[2] M. Jarke, "Query optimization in database systems," in ACM Computing Surveys, Vol. 16, No. 2, 1984, pp 11-152.

[3] S. Chaudhuri, "An Overview of Query Optimization". In Proceedings of the seventeenth ACM SIGACT, 1998, pp. 34-43.

[4] S. Ceri and G. Gottlob, "Translating SQL Into Relational Algebra: Optimization, Semantics, and Equivalence of SQL Queries," in IEEE Transactions on Software Engineering, Vol. SE-11, No. 4, 1985, pp. 324-345.

[5] D. Taniar, H. Khaw, T. H. Cokrowijoyo, and E. Pardede, "The use of Hints in SQL-Nested query optimization," Information Sciences, Vol. 177, No 12, 2007, pp. 2493-2521.

[6] D. Li, L. Han, and Y. Ding, "YiSQL Query Optimization Methods of Relational Database System," in Second International Conference on Computer Engineering and Applications, 2010, pp. 557-560.

[7] S. Mittal, "Power Management Techniques for Data Centers: A Survey", 2014. [online] Available at: http://arxiv.org/abs/1404.6681 [Accessed 11 March 2016].

[8] M. Poess and R. O. Nambiar, "Energy cost, the key challenge of today's data centers: a power consumption analysis of TPC-C results," in Proceedings of the VLDB Endowment, vol. 1, 2008, pp. 1229-1240.

[9] R. Gonçalves, J. Saraiva, and O. Belo, "Defining Energy Consumption Plans for Data Querying Processes". In Proceedings of 2014 IEEE Fourth International Conference on Big Data and Cloud Computing (BdCloud 2014), IEEE computer Society, Sidney, Australia, 2014, pp. 641-647.

[10] T. Carção, "Measuring and visualizing energy consumption within software code". In: Visual Languages and Human-Centric Computing (VL/HCC), 2014 IEEE Symposium on, July, 2014, pp. 181– 182.

[11] M. Couto, T. Carção, J. Cunha, J. P. Fernandes, and J. Saraiva, "Detecting anomalous energy consumption in android applications". In Pereira, F.M.Q., ed.: Programming Languages - 18th Brazilian Symposium, SBLP 2014, Maceio, Brazil, October 2-3, 2014. Proceedings. Volume 8771 of Lecture Notes in Computer Science., Springer, 2014, pp. 77–91.

[12] R. Agrawal et al., "The claremont report on database research". SIGMOD Rec. 37(3), September, 2008, pp. 9–19.

[13] J. Wang, L. Feng, W. Xue, and Z. Song, "A Survey on Energy-efficient Data Management," in SIGMOD Rec. 40, 2, September, 2011, pp. 17-23.

[14] W. Lang, R. Kandhan, and J. M. Patel, "Rethinking query processing for energy efficiency: Slowing down to win the race". IEEE Data Eng. Bull. 34(1), 2011, pp. 12–23.

[15] W. Lang and J. M. Patel, "Towards eco-friendly database management systems". In CIDR 2009, Fourth Biennial Conference on Innovative Data Systems Research, Asilomar, CA, USA, January 4-7, 2009, Online Proceedings, www.cidrdb.org, 2009.

[16] Z. Xu, Y. Tu, and X. Wang, "Exploring power-performance tradeoffs in database systems". In Li, F., Moro, M.M., Ghandeharizadeh, S., Haritsa, J.R., Weikum, G., Carey, M.J., Casati, F., Chang, E.Y., Manolescu, I., Mehrotra, S., Dayal, U., Tsotras, V.J., eds.: Proceedings of the 26th International Conference on Data Engineering, ICDE 2010, March 1-6, 2010, Long Beach, California, USA, IEEE, 2010, pp. 485–496.

[17] M. Kunjir, P. K. Birwa, and J. R. Haritsa, "Peak power plays in database engines". In E. A. Rundensteiner, V. Markl, I. Manolescu, S. Amer-Yahia, F. Naumann, and I. Ari, eds.: 15th International Conference on Extending Database Technology, EDBT '12, Berlin, Germany, March 27-30, 2012, Proceedings, ACM, 2012, pp. 444–455.

[18] M. Rodríguez et al., "Analyzing power and energy consumption of large join queries in database systems," Industrial Electronics and Applications (ISIEA), 2013 IEEE Symposium on, Kuching, 2013, pp. 148-153.

[19] Z. Xu, Y. Tu, and X. Wang. "PET: reducing Database Energy Cost via Query Optimization," in Proc. VLDB Endow. 5, 12, August, 2012, pp. 1954-1957.

[20] K. Liu, G. Pinto, and D. Liu, "Data-oriented characterization of application-level energy optimization", in: Proceedings of the 18th International Conference on Fundamental Approaches to Software Engineering, FASE'15, 2015.

[21] T. Connoly and C. Begg, "Database Systems: A pratical Approach to Design, Implementation, and Management", 2005, Addison-Wesley Longman Publishing Co., Inc., Boston, USA

[22] A. V. Aho and J. D. Ullman, "Universality of Data Retrieval Languages", in Proceedings of the 6th ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages, POPL '79, (New York, NY, USA), ACM, 1979, pp. 110–119.

 ISBN: 978-1-61208-486-2