Sebastian Götz, Thomas Ilsche, Jorge Cardoso, Josef Spillner, Thomas Kissinger, Uwe Aßmann, Wolfgang Lehner, Wolfgang E. Nagel, Alexander Schill

# Energy-Efficient Databases Using Sweet Spot Frequencies

**Erstveröffentlichung in / First published in:**

*2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing.* London, 08.-11.12.2014. IEEE, S. 871-876. ISBN 978-1-4799-7881-6.

DOI: http://dx.doi.org/10.1109/UCC.2014.142

Final edited form was published in "2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing. London 2014", S. 871-876, ISBN 978-1-4799-7881-6
http://dx.doi.org/10.1109/UCC.2014.142

# Energy-Efficient Databases using Sweet Spot Frequencies

Sebastian Götz*, Thomas Ilsche†, Jorge Cardoso*‡, Josef Spillner*, Thomas Kissinger*,
Uwe Aßmann*, Wolfgang Lehner*, Wolfgang E. Nagel*†, Alexander Schill*
* Technische Universität Dresden, Faculty of Computer Science, Germany
Email: sebastian.goetz@acm.org, {*firstname.surname*}@tu-dresden.de
† Technische Universität Dresden, Center for Information Services and High Performance Computing, Germany
Email: thomas.ilsche@tu-dresden.de
‡ University of Coimbra, Department of Informatics Engineering, Portugal
Email: jcardoso@dei.uc.pt

***Abstract*—Database management systems (DBMS) are typically tuned for high performance and scalability. Nevertheless, carbon footprint and energy efficiency are also becoming increasing concerns. Unfortunately, existing studies mainly present theoretical contributions but fall short on proposing practical techniques. These could be used by administrators or query optimizers to increase the energy efficiency of the DBMS. Thus, this paper explores the effect of so-called sweet spots, which are energy-efficient CPU frequencies, on the energy required to execute queries. From our findings, we derive the *Sweet Spot Technique*, which relies on identifying energy-efficient sweet spots and the optimal number of threads that minimizes energy consumption for a query or an entire database workload. The technique is simple and has a practical implementation leading to energy savings of up to 50% compared to using the nominal frequency and maximum number of threads.**

## I. Introduction

A large amount of research done in the field of database management systems (DBMS) has been looking into how new methods and techniques could improve the performance, throughput, and scalability of systems [1], [2], [3], [4], [5]. Nonetheless, increasing concerns about energy costs and increasingly large data sets that need to be processed are calling for the development of energy-efficient DBMS. The energy consumption caused by running analytical queries over large data sets – especially in the era of Big Data – is not negligible when compared to the overall consumption of data centers despite an increase of the number of transactions per Watt [6]. In 2010, the electricity used in global data centers accounted for between 1.1% and 1.5% of total electricity use. Data center traffic is expected to quadruple by 2016. This calls for the development of new energy-efficient approaches to reduce their consumption [7].

Existing studies on the energy efficiency of DBMS suffer from three restrictions: they are limited, occasionally contradictory, and fall short on providing techniques which can be easily applied to most existing DBMS. For example, the study described in [8] indicates that power savings in the range of 11%–22% can be achieved by equipping DBMS with a query optimizer that selects query plans based on both estimated processing time and power requirements. In [9], the authors describe a scenario where the use of uncompressed tables results in a more energy-efficient query execution. On the other hand, Tsirogiannis et al. [10] have found no concluding evidence that these techniques can indeed lead to a more energy-efficient DBMS. Their observation is that computing energy efficiency without taking into consideration the power of peripheral components and the idle power of the CPU may lead to a reduced energy consumption of one component (e.g., the CPU), but does not contribute to the energy reduction of the overall system. The final conclusion is that the highest CPU frequency is the most energy-efficient one.

In this paper, we revisit the critiques by Tsirogiannis et al. [10] and Xu et al. [8] to investigate the effect of frequency scaling on the energy efficiency of a DBMS. In particular, we take a closer look at the query-level energy efficiency in relational database systems. Read-only queries (i.e., SELECT and FETCH) account for the bulk of database operations, around 97% according to IBM's REDWAR analysis [11]. Our aim is to reproduce the results of previous work and, at the same time, to consider the impact of additional configuration options: multi-threading, turbo mode, and the use of a column-oriented database. We measured the energy consumption of the SELECT queries that are part of the TPC-H benchmark against the columnar database system MonetDB.

Our findings reveal two important aspects to consider when developing an energy-efficient DBMS. Energy-efficiency can be improved by running queries in a specific configuration. This is confirmed with AC measurements that include the static and dynamic power consumption of the whole system. The energy efficiency of a query can be maximized in sweet spots with high performance-power consumption ratios due to the non-linear nature of processor power curves. Based on these findings, we propose the *Sweet Spot Technique*, which can guide database administrators or query optimizers to benchmark DBMS following the same steps we carried out in our experiment to identify the most energy-efficient configuration for their workload.

This paper is structured as follows. Sect. II identifies the research questions to be answered by this study. Sect. III describes the context (hardware, software, and configurations)

Final edited form was published in "2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing. London 2014", S. 871-876, ISBN 978-1-4799-7881-6
http://dx.doi.org/10.1109/UCC.2014.142

of the experiments. We discuss the results in Sect. IV and describe, in Sect. V, a technique that developers can use to improve the energy efficiency of their database applications. Finally, Sect. VI and VII present related work and our conclusions, respectively.

## II. Research Questions

The work presented in this paper is motivated by our previous work (see [12]), which showed that combinations of algorithms and processors have so-called *sweet spots*, CPU frequencies, at which the execution of a computational task is more energy-efficient. Our objective is to discover to which extent the existence of sweet spots is usable to design more energy-efficient database systems.

Our research questions (RQ) are the following:

- RQ1 (Measurement Setup). How to instrument a system to obtain per-query energy measurements?
- RQ2 (Sweet spots). Can sweet spot frequencies be identified when running queries against a database?
- RQ3 (Threads). What is the influence of the number of threads used to run a query on sweet spot frequencies?

To answer these questions, we follow an experimental and empirical approach to study the energy impact of running queries against a DBMS by varying database parameters and CPU settings (configurations).

## III. Experiment Description

For each configuration, we measure the time and energy consumed by analytical queries and store these results as comma-separated values in files. The following sections outline the hardware and software used in our experiments. Additionally, the configurations used are precisely described to encourage replications and extensions of our experiments on different systems.

### A. Hardware Setup

The system under test is equipped with two Intel Xeon E5-2690 (Sandy Bridge) processors. Each processor has 8 physical cores and supports Hyper-Threading. The core frequencies can be set from 1.2 to 2.9 GHz and turbo mode with up to 3.8 GHz. All cores of one socket use the same frequency and voltage configuration. The server is equipped with an internal SSD (Intel 520 series, 120 GB).

The total real power consumption of the machine is measured with a calibrated ZES Zimmer LMG450 power analyzer at the AC input of the power supply unit (PSU). A dedicated separate measurement system reads 20 values per second from the power analyzer to avoid any perturbation of the system under test. Furthermore, all DC outputs of the PSU are instrumented with custom-built, shunt-based sensors at the respective Molex adapters. These measurements separately capture the two 12V connectors to the sockets, the different ATX voltages, the SATA/SSD power and all the fans in the system. In addition to these, one of the DDR3 DIMMs is measured at a DIMM-riser.

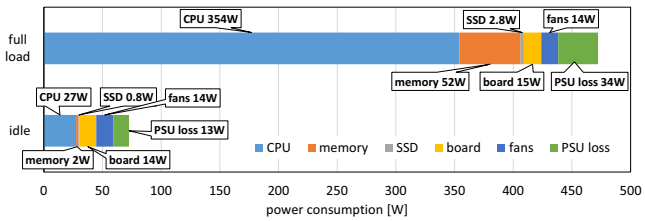



Fig. 1: Breakdown of the component power consumption. The full load configuration was measured at sustained turbo mode with FIRESTARTER [13] and "cp" for the SSD.

As done in related work [10], we first characterize the power consumption of the machine. For this, we use the DC instrumentation of the system. The memory power consumption for all 8 DIMMs is estimated from the measurements of one DIMM and translated to the 12V consumption assuming a voltage regulator efficiency of 85% [14]. To estimate the CPU power consumption, which is not instrumented directly, the memory power is substracted from the 12V input to each socket. The PSU losses are computed from the difference between the AC measurement and the sum of all DC measurements. They are load dependant but also include a base consumption and a fan within the PSU.

The breakdown of power consumption in Figure 1 shows that the CPU power consumption is the dominating factor, especially under high load. This confirms the ability to preserve energy by adapting the CPU frequencies. Nevertheless, the constant power consumption of the fans[1] and mainboard should not be neglected when considering reductions of the power consumption that increase the overall runtime. We therefore use the total real power measured by the LMG450 for all further energy efficiency considerations.

### B. Software Setup

On the test system, which runs Linux, the MonetDB column-oriented DBMS is installed. MonetDB is a well-known open source software to be used primarily for online analytical processing tasks[2]. As input for the database, the industry standard benchmark TPC-H designed by the Transaction Processing Performance Council[3] was used. TPC-H includes a set of 22 SQL SELECT queries to be used to evaluate relational DBMS. TPC-H was configured with a scale factor of 50 to have long-running queries for more accurate measurement, but the database should still fit into memory. We initiate the queries from the system under test and run only one query at the same time. The latter is necessary to correlate the energy consumption caused by individual queries.

During the initial experiments, we found out about MonetDB's result caching causing a lot of write operations to disk even when operating on a read-only database. Therefore, we patched MonetDB to forcibly disable this caching. Results with unpatched versions may differ considerably.

[1]The fans were configured to run at constant speed.
[2]MonetDB website: https://www.monetdb.org/, Version11.17.13
[3]TPC-H benchmark website: http://www.tpc.org/tpch/, Version2.17.0

Final edited form was published in "2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing. London 2014", S. 871-876, ISBN 978-1-4799-7881-6
http://dx.doi.org/10.1109/UCC.2014.142

### C. Configurations

We measured energy consumption for various database and hardware configurations. Each configuration was characterized by 3 variables:

1) $TPC-H_{query\#}$: {1, ..., 14, 16, ..., 22}.[4]
2) $CPU_{thread\#}$: {1, 2, 4, 8, 16, 32}
3) $CPU_{frequency}$: {1.2, 1.3, 1.4, 1.6, 1.7, 1.8, 1.9, 2.0, 2.2, 2.3, 2.4, 2.5, 2.7, 2.8, 2.9, and turbo mode}.

To account for variance, each query was repeated 8 times, but the first repetition was excluded from analysis to allow the database to warm up. The result of the queries is not cached for successive executions. Among the various TPC-H benchmarks, we selected a subset of queries with a small time and energy variance over repetitions to allow a significant comparison. Good candidates are those which scan large amounts of data. We tested all queries but then focused our investigation to queries 9, 16 and 21.

## IV. Results of the Experiments

Figures 2 and 3 show the results (energy and runtime) for the three selected queries. As expected, the runtime is shortest with turbo mode for the three queries[5]. Query 9 is fastest with 8 threads, while query 16 and 21 do benefit from Hyper-Threading. Figure 3b exposes another interesting effect: Going from a single thread to two threads increases the runtime due to parallelization overhead, but further threads actually improve the runtime even up to Hyper-Threading. However, as seen in Figure 2b, the energy consumption is optimal at only one thread as a result of the low parallel efficiency and increased power requirements for more active cores. With only one thread being active for query 16, turbo mode actually results in the lowest energy consumption. In our case with only one query being executed at the same time, the other cores of the system are in sleep states and therefore largely unaffected by the frequency setting. Consequently, *race-to-idle* is a good strategy for this query. Results for query 9 and 21 indicate a lower thread count and frequency as energy-optimal setting (compared to the fastest setting). Their energy consumption does not benefit from Hyper-Threading. The results show that there is not one optimal frequency - it really depends on the specific workload characteristics that are being executed, i.e. memory requirements, thread synchronization bottlenecks, etc.

A total overview for all queries is displayed in Figures 4a, 4b and 4c, and shows the sweet spots with respect to energy, time and the energy delay product (EDP). The EDP, computed as the product of energy and time, is a metric to summarize the compromise between performance and energy and was discussed by Laros et al. in [15]. Similarly to time, the EDP is optimal at the turbo frequency setting, but does not benefit from utilizing all available cores.

Table I summarizes the results for all queries by showing the respective energy-optimal settings and the energy savings as well as performance penalty. As a baseline, we use 32 threads (all logical threads used) and 2.9 GHz (nominal frequency), which is a common default setting. In addition, when using an optimal configuration per query the overall energy and time savings are 16.8% and 3.4%, respectively. For the queries 9 and 17, the energy saving is >50% - they are parallelized inefficiently and benefit with respect to energy and time from using less threads. Using a single optimal configuration for all queries results in 12% energy and 29.7% time savings.

| Query | Frequency [MHz] | Threads | Energy Savings [%] ▫ Time Savings [%] ▫ |
|---|---|---|---|
| 1 | 1900 | 32 | |
| 2 | 2400 | 16 | |
| 3 | 2800 | 16 | |
| 4 | 2700 | 16 | |
| 5 | 2700 | 16 | |
| 6 | 2400 | 16 | |
| 7 | 2200 | 32 | |
| 8 | turbo | 16 | |
| 9 | 2800 | 4 | |
| 10 | 2800 | 32 | |
| 11 | 2500 | 32 | |
| 12 | 2000 | 16 | |
| 13 | 2700 | 16 | |
| 14 | 2500 | 8 | |
| 16 | turbo | 1 | |
| 17 | 2900 | 8 | |
| 18 | 2400 | 32 | |
| 19 | 2300 | 32 | |
| 20 | turbo | 8 | |
| 21 | 2400 | 16 | |
| 22 | 2700 | 8 | |

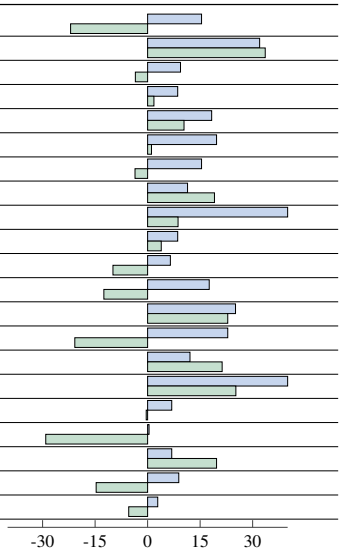



TABLE I: Energy savings and time savings for all investigated queries using sweet spots and a varying number of threads.

The answers to our research questions were the following:

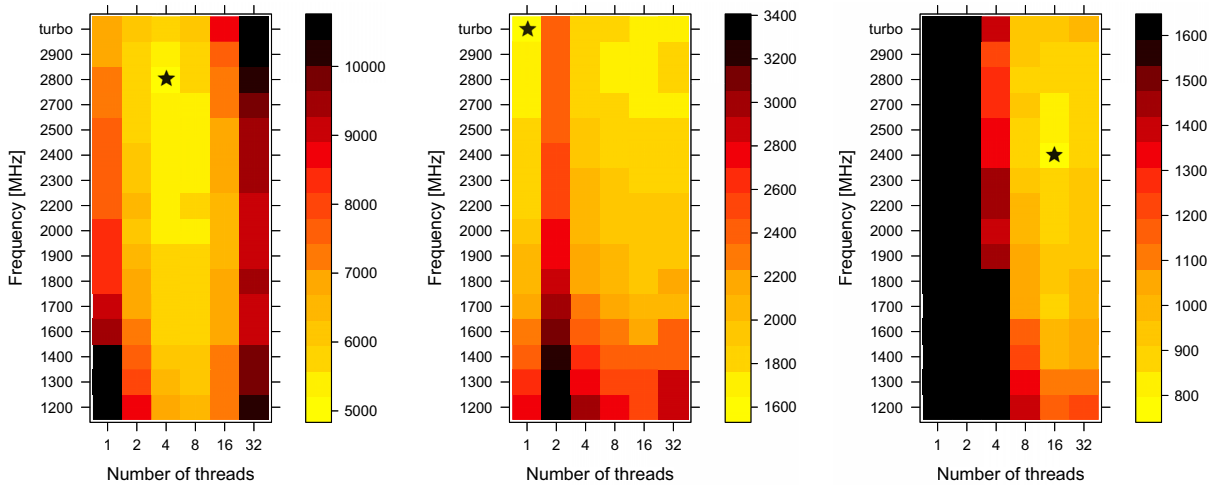
*RQ1 (Measurement Setup).* To analyze the energy efficiency of an complete system, the total AC power consumption should be taken into account. However, the real AC power consumption can only be measured at relatively low update intervals, in our case one value every 50 ms. Xu et al. [8] even use only one value per second. Therefore, the runtime of the measured task (e.g., one query execution) should be significantly longer than the update interval.

*RQ2 (Sweet spots).* When using a low number of threads, all queries show the optimal energy consumption is either at nominal maximum frequency or turbo mode. Due to the high relative impact of the constant base power consumption, the energy in these cases is almost proportional with time. This results in an efficient *race-to-idle* strategy. Interestingly, with an increasing number of threads, varying the CPU frequency has less impact on energy consumption. Notably, the data for eight and more threads show sweet spots at different frequencies ranging from 1900 MHz to turbo mode. Therefore, we argue that processor- and application-specific configurations exist, which improve energy efficiency.

[4]Query 15 is not used because it does not operate in read-only mode.

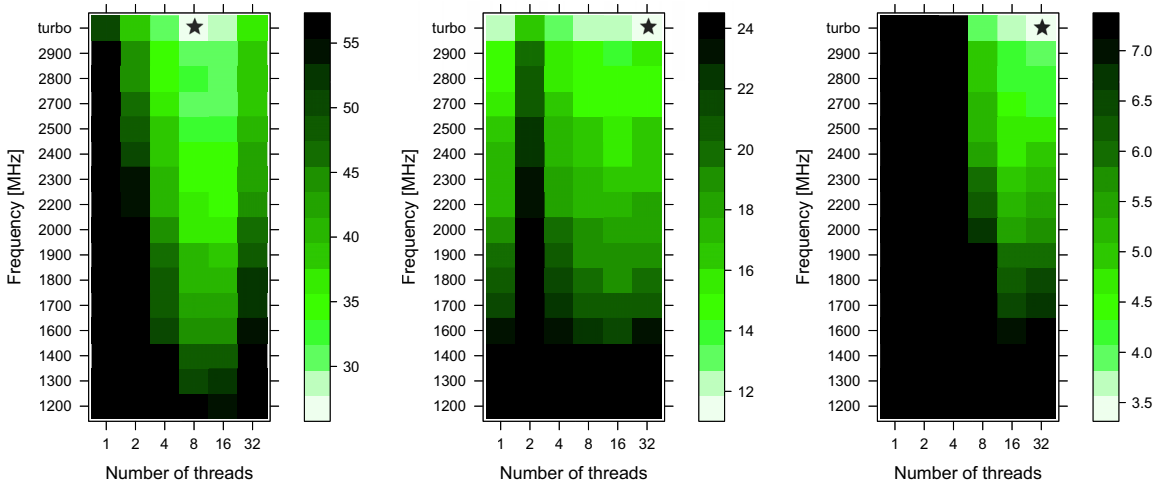
[5]For queries 1, 6 and 11, these results indicate a shortest execution for a frequency other than turbo mode. However, these queries show a high variance, thus, this observation is not statistically significant.

Final edited form was published in "2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing. London 2014", S. 871-876, ISBN 978-1-4799-7881-6
http://dx.doi.org/10.1109/UCC.2014.142




(a) Query 9 (b) Query 16 (c) Query 21

Fig. 2: AC energy consumption of different queries based on CPU frequency and the number of threads. Three different selected TPC-H queries are shown. The lowest energy value is marked with a star (⋆), all energy values more than twice the optimal value are shown in black. Each value represents the average energy consumption of 7 repetitions.




(a) Query 9 (b) Query 16 (c) Query 21

Fig. 3: Runtime of different queries based on CPU frequency and the number of threads. Three different selected TPC-H queries are shown. The shortest time is marked with a star (⋆), all times larger than twice the shortest time are shown in black. Each value represents the average runtime of 7 repetitions.

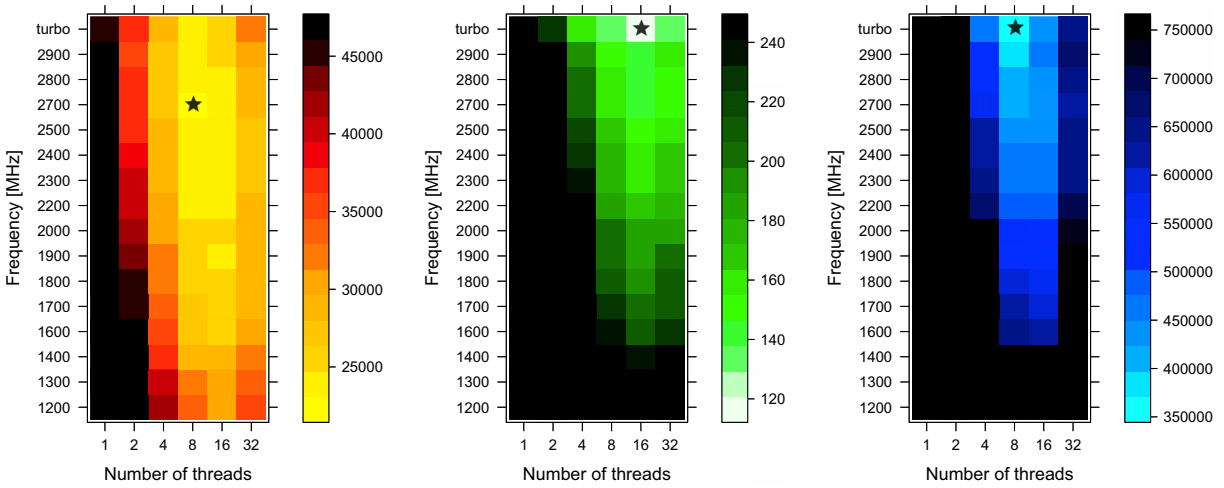
*RQ3 (Threads).* Another interesting observation is the energy consumption in relation to the number of worker threads and the use of Hyper-Threading. Primarily, the number of threads influences the runtime and, therefore, the energy. In several cases more threads actually result in a slowdown - which as a consequence also worsens the energy consumption. An example is query 9: the fastest execution is with eight threads. Whenever the runtime scales very well (inversely proportional to the number of threads), it is also clear that a large number of threads is beneficial for energy consumption. The most interesting cases are when increasing parallelism does improve runtime, but not to the extent that energy consumption is increased. This is why it is important to take parallelism into account when considering the trade-off between performance and energy.

## V. Sweet Spot Technique

Based on the experiments we conducted, we propose the following 3-step technique for database administrators to optimize the energy efficiency of database systems:

1) *Workload Recording*. Record a representative workload of the production DBMS and calculate the probability of occurrence of each query (type or class).

Final edited form was published in "2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing. London 2014", S. 871-876, ISBN 978-1-4799-7881-6
http://dx.doi.org/10.1109/UCC.2014.142




(a) Energy [J] (b) Time [s] (c) EDP [Js]

Fig. 4: Average AC energy consumption, runtime and EDP over all queries based on CPU frequency and the number of threads. The lowest consumption, runtime and EDP are marked with a star ($\star$), all values larger than twice the lowest value are shown in black. Each value represents the average of 7 repetitions.

2) *Query Benchmarking*. Benchmark all queries to collect their run time and energy consumption in correlation to the number of threads and frequency used.
3) *Query Aggregation*. Select the optimal configuration among all queries using weighted benchmark results.

Today's database systems are usually faced with a specific workload that changes over time. This workload consists of a set of application-specific precompiled queries and ad-hoc queries. Thus, the first step of our technique is to record the workload of the running DBMS. Based on the recorded workload, we are able to compute probabilities of occurrence for each query that is part of the workload.

In the next step, all queries are benchmarked for all combinations of CPU frequencies and threads, and the execution time as well as energy consumption for each configuration is stored in a matrix. Since – especially for long-running queries – this is the most time consuming part of the process, experimental design methods can be applied to limit the configuration space and queries can be executed on a data sample.

These measurement results will then be aggregated to reflect the overall effect of each configuration option on runtime and energy consumption. The aggregation is a weighted sum, where the probability of each query to be executed in the production system (relatively to the other queries) is the weight. This results in two matrices ($M_E$ and $M_t$) with a size of $F \times T$, where F denotes the number of available frequencies and T the number of threads to be used. The values of the two matrices are the total runtime and total energy consumption, respectively.

To identify the optimal configuration, the two matrices can be normalized by dividing them with the optimal value respectively. We denote these matrices as $M_E^1$ and $M_t^1$. The EDP being the product of both matrices (i.e., $M = M_E^1 \cdot M_t^1$), then forms an additional basis to select the optimal configuration, which is the lowest value of $M$. At this point it can be decided to optimize for EDP, energy or time.

Finally, either the database administrator or an automated system has to deploy this configuration to the running system. Since the workload is a "moving target" and physical data structures of the DBMS are also subject to change, which affects the energy-performance profile of queries. The entire process should be periodically repeated preferably at times of a low database load.

## VI. Related Work

Tsirogiannis et al. [10] found that unlike previous studies had suggested, the highest performing configuration is the most energy-efficient. In the few cases where this did not hold, the improvements in energy efficiency were less than 10%. This result is mainly due to the large, up-front power costs of their investigated systems. Our work shows that these findings do not hold for current state-of-the-art systems. The ratio of maximum system power vs idle system power was less than two for the system they tested. In our experiments, which used a more recent system, it is more than six. Naturally, *race-to-idle* is a much less efficient strategy for modern systems.

Lang and Patel [16] have explored Processor Voltage/Frequency Control (PVC) to explicitly reduce the processor voltage and frequency parameters when the processor is idle, or underutilized, or current system settings require a lower performance level. They have shown that PVC can reduce the processor energy consumption of an (unnamed) commercial BDMS by 49% while increasing the response time by 3%. On MySQL, the approach reduces energy consumption by 20% with a response time penalty of 6%. The limitation of this work is that only TPC-H Q5 has been tested. Nonetheless,

Final edited form was published in "2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing. London 2014", S. 871-876, ISBN 978-1-4799-7881-6
http://dx.doi.org/10.1109/UCC.2014.142

the TPC-H contain 22 queries (Q1-Q22). They also explore workload management by delaying the execution of queries to find sets of queries with common components.

The work of Rodriguez-Martinez et al. [17] developed a cost model to estimate the cost of database queries. The approach used multiple-linear regression on data describing SQL queries (TPC-H data sets), the characteristics of the tables, columns, cardinalities, the number of DBMS servers (PostgreSQL on Ubuntu), and energy measurements to derive a mathematical estimation function. The cost models are derived from internal sensors, or power meters.

Harizopoulos et al. [9] argue for focusing on database management software as a primary target for reducing energy consumption in data centers. They offer two experiments, with TPC-H and relational scans, but do not analyze queries, threads or sweet spots in detail. Their work is not concerned with user choices over the trade-off between energy efficiency and performance.

The more recent work of Xu et al. [8] gives quantitative results for a setup with PostgreSQL similar to the work of Rodriguez-Martinez et al. [17]. Their encouraging results of 11-22% power savings are surpassed by our detailed analysis which yields up to 50% savings when optimizing the configuration for specific queries.

Livingston et al. [18] discuss energy savings at runtime with a novel hybrid hardware-software approach called REST which uses profiling and dynamic CPU/memory usage pattern driven frequency scaling. They show that sweet spots can be found for almost any algorithm under test. Our approach currently considers static frequencies per query instead. Their work is based on older hardware on which memory bandwidth does not depend on CPU frequency.

While most researchers use TPC-H and we followed this approach for easier comparison, alternative approaches for determining database energy efficiency are now becoming feasible. JouleSort [19][6] is an external sort benchmark for evaluating the energy efficiency of a wide range of computer systems. SPECpower is another benchmark which produces a server-side Java operations per Watt ($ssj\_ops/W$) metric.

## VII. Conclusion and Future Work

Database queries can be tuned to address energy efficiency requirements by using appropriate hardware and software configurations and using sweet spot frequencies (i.e., frequency scaling). Our experiments have confirmed the presence of sweet spot frequencies. Furthermore, we have also shown how to determine the optimal number of threads for the MonetDB system and that multithreading up to the number of physical cores yields decreasing but always worthwhile efficiency gains. Both results enabled to tailor the Sweet Spot Technique, to be used by developers to reduce the energy consumption of their DBMS.

In the future, we plan to extend the investigation by dividing queries into individual operators and determining the operator-level energy efficiency to optimize arbitrary query plans.

[6]See also: http://sortbenchmark.org/

## Acknowledgements

This work has been partially funded by the German Research Foundation (DFG) under project agreements SFB 912/1 2011 and SCHI 402/11-1.

## References

[1] R. Tudoran, O. Nano, I. Santos, A. Costan, H. Soncu, L. Bougé, and G. Antoniu, "JetStream: Enabling High Performance Event Streaming across Cloud Data-Centers," in *Proceedings of the 8th ACM International Conference on Distributed Event-Based Systems (DEBS)*, Mumbai, India, May 2013, pp. 23–34.

[2] Y. Li and J. M. Patel, "BitWeaving: Fast Scans for Main Memory Data Processing," in *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data (SIGMOD)*, New York City, New York, USA, June 2013, pp. 289–300.

[3] G. Liu, A. Suchitra, and L. Wong, "A Performance Study of Three Disk-based Structures for Indexing and Querying Frequent Itemsets," *Proceedings of the VLDB Endowment*, vol. 6, no. 7, pp. 505–516, 2013.

[4] P. A. Boncz, M. L. Kersten, and S. Manegold, "Breaking the Memory Wall in MonetDB," *Commun. ACM*, vol. 51, no. 12, pp. 77–85, 2008.

[5] V. Leis, P. A. Boncz, A. Kemper, and T. Neumann, "Morsel-Driven Parallelism: A NUMA-Aware Query Evaluation Framework for the Many-Core Age," in *SIGMOD Conference*, 2014, pp. 743–754.

[6] M. Poess and R. O. Nambiar, "Energy Cost, The Key Challenge of Today's Data Centers: A Power Consumption Analysis of TPC-C Results," *Proceedings of the VLDB Endowment*, vol. 1, no. 2, pp. 1229–1240, 2008.

[7] J. Koomey., "Growth in Data center electricity use 2005 to 2010," Analytics Press, 2011. [Online]. Available: http://www.analyticspress.com/datacenters.html

[8] Z. Xu, Y.-C. Tu, and X. Wang, "Exploring power-performance tradeoffs in database systems," in *Data Engineering (ICDE), 2010 IEEE 26th International Conference on*, March 2010, pp. 485–496.

[9] S. Harizopoulos, M. A. Shah, J. Meza, and P. Ranganathan, "Energy Efficiency: The New Holy Grail of Data Management Systems Research," in *4th Biennial Conference on Innovative Data Systems Research (CIDR)*, Asilomar, California, USA, January 2009, pp. 1–8.

[10] D. Tsirogiannis, S. Harizopoulos, and M. A. Shah, "Analyzing the energy efficiency of a database server," in *Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD '10. New York, NY, USA: ACM, 2010, pp. 231–242.

[11] P. S. Yu, M.-S. Chen, H.-U. Heiss, and S. Lee, "On Workload Characterization of Relational Database Environments," *IEEE Transactions on Software Engineering*, vol. 18, no. 4, pp. 347–355, April 1992.

[12] S. Götz, T. Ilsche, J. Cardoso, J. Spillner, U. Aßmann, W. E. Nagel, and A. Schill, "Energy-efficient data processing at sweet spot frequencies," in *4th Intl. Symposium on Cloud Computing, Trusted Computing and Secure Virtual Infrastructures (CTC14)*, 2014, accepted for publication.

[13] D. Hackenberg, R. Oldenburg, D. Molka, and R. Schöne, "Introducing FIRESTARTER: A processor stress test utility," in *Green Computing Conference (IGCC), 2013 International*, 2013, pp. 1–9.

[14] W. Kim, M. S. Gupta, G. yeon Wei, and D. Brooks, "System level analysis of fast, per-core DVFS using on-chip switching regulators," in *in International Symposium on High-Performance Computer Architecture*, 2008.

[15] J. Laros III, K. Pedretti, S. Kelly, W. Shu, K. Ferreira, J. Vandyke, and C. Vaughan, "Energy delay product," in *Energy-Efficient High Performance Computing*, ser. SpringerBriefs in Computer Science. Springer London, 2013, pp. 51–55.

[16] W. Lang and J. M. Patel, "Towards Eco-friendly Database Management Systems," in *4th Biennial Conference on Innovative Data Systems Research (CIDR)*, Asilomar, California, USA, January 2009, pp. 1–8.

[17] M. Rodriguez-Martinez, H. Valdivia, J. Seguel, and M. Greer, "Estimating power/energy consumption in database servers," *Procedia Computer Science*, vol. 6, no. 0, pp. 112 – 117, 2011, complex adaptive sysytems.

[18] K. Livingston, N. Triquenaux, T. Fighiera, J. Beyler, and W. Jalby, "Computer using too much power? give it a rest (runtime energy saving technology)," *Computer Science - Research and Development*, 2012.

[19] S. Rivoire, M. A. Shah, P. Ranganathan, and C. Kozyrakis, "Joulesort: A balanced energy-efficiency benchmark," in *Proceedings of the ACM SIGMOD Intl. Conference on Management of Data (SIGMOD)*, 2007, pp. 365–376.