Dieses Dokument ist eine Zweitveröffentlichung (Postprint) / This is a self-archiving document (accepted version):

Thomas Kissinger, Marcus Hähnel, Till Smejkal, Dirk Habich, Hermann Härtig, Wolfgang Lehner

Energy-Utility Function-Based Resource Control for In-Memory Database Systems LIVE

Erstveröffentlichung in / First published in:

SIGMOD/PODS '18: International Conference on Management of Data, Houston 10.-15.06.2018. ACM Digital Library, S. 1717–1720. ISBN: 978-1-4503-4703-7

DOI: https://doi.org/10.1145/3183713.3193554

Diese Version ist verfügbar / This version is available on: https://nbn-resolving.org/urn:nbn:de:bsz:14-qucosa2-803739







Energy-Utility Function-Based Resource Control for In-Memory Database Systems LIVE

Thomas Kissinger^{*} Marcus Hähnel[#] Till Smejkal[#] Dirk Habich^{*}

Hermann Härtig[#] Wolfgang Lehner^{*}

*Database Systems Group / #Operating Systems Group

Technische Universität Dresden

Dresden, Germany

{first.last}@tu-dresden.de

ABSTRACT

The ever-increasing demand for scalable database systems is limited by their energy consumption, which is one of the major challenges in research today. While existing approaches mainly focused on transaction-oriented disk-based database systems, we are investigating and optimizing the energy consumption and performance of data-oriented scale-up in-memory database systems that make heavy use of the main power consumers, which are processors and main memory. In this demo, we present energy-utility functions as an approach for enabling the operating system to improve the energy efficiency of scalable in-memory database systems. Our highly interactive demo setup mainly allows attendees to switch between multiple DBMS workloads and watch in detail how the system responds by adapting the hardware configuration appropriately.

CCS CONCEPTS

Information systems → Main memory engines; Relational parallel and distributed DBMSs;
Software and its engineering → Power management;

KEYWORDS

in-memory, energy efficiency, adaptivity, elasticity

ACM Reference Format:

Thomas Kissinger, Marcus Hähnel, Till Smejkal, Dirk Habich, Hermann Härtig, Wolfgang Lehner. 2018. Energy-Utility Function-Based Resource Control for In-Memory Database Systems LIVE. In *SIGMOD'18: 2018 International Conference on Management of Data, June 10–15, 2018, Houston, TX, USA*. ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3183713.3193554

1 INTRODUCTION

To keep pace with the ever-increasing data volume, modern stateof-the-art in-memory database systems need to scale up on server hardware featuring an increasing amount of main memory and compute resources usually implementing a *non-uniform memory access* (*NUMA*). The power draw of this server hardware in general

DOI: https://doi.org/10.1145/3183713.3193554



Figure 1: Architecture of Energy-utility function-based resource control of a data-oriented in-memory DBMS.

and especially database servers as a fundamental component of mostly every service became a severe issue in the recent years. Hardware vendors have become more and more aware of this problem and focus on increasing the energy efficiency (performance per Watt) and energy proportionality (proportionality between supplied performance and power draw) of processors and servers as a whole by adding a rich set of energy-control knobs (e.g., sleep states, separate core and uncore clocks). From the software perspective, data-oriented in-memory database systems [2, 4, 5] were proposed that employ worker threads, which exclusively operate on local data partitions and communicate via a message passing layer (cf., Figure 1), exhibiting a superior scalability, which is a major prerequisite for achieving energy proportionality and thus energy efficiency. However, those modern in-memory database systems do all the thread scheduling on their own (e.g., to ensure NUMA awareness) and threads are rarely blocked while waiting on secondary storage devices. Hence, hardware and operating system have almost no chance to appropriately configure the energy-related tuning knobs of the hardware due to their limited DBMS insights. Thus, it is an essential step to add more sophisticated energy-control mechanisms to the operating system, which requires additional interfaces for pushing down database-specific runtime information.

In this demo, we address this issue by extending our recent approach of a DBMS-integrated hierarchical *energy-control loop (ECL)* [3], which is organized in multiple *socket-level ECLs* and a *system-level ECL*. While the first ones are responsible for configuring the socket-local hardware components using an adaptively

^{©2018} Copyright held by the owner/author(s). Publication rights licensed to ACM. This is the author's version of the work. It is posted here for your personal use. Not for redistribution. The definitive Version of Record was published in *SIGMOD'18, June* 10–15, 2018, Houston, TX, USA



Figure 2: Energy-utility functions (EUF) for a compute-bound, memory bandwidth-bound, and shared hash table insert workload for a single socket. Circles are hardware configurations. Inner color shows uncore frequency; outer color the average core frequency; diameter indicates the number of active cores.

maintained *energy-utility function (EUF)*, the latter one keeps track of current query latencies and uses a best-effort strategy to stay within a user-defined latency limit. As shown in Figure 1, our extension comprises (1) the decomposition of EUFs into a *hardware energy model* and a *software model* to reduce EUF maintenance efforts [1] and (2) the outsourcing of the energy-control mechanism to the operating system to pave the way for multi-DBMS instance energy management on a single machine.

2 ENERGY-UTILITY FUNCTION-BASED RESOURCE CONTROL

In this section, we discuss our overall approach for adaptive energycontrol of data-oriented in-memory database systems running on NUMA-based scale-up hardware architectures and cover the foundations of EUF-based hardware resource control. Figure 1 visualizes the architecture of our approach. The application under control is a data-oriented in-memory DBMS that implicitly partitions all data objects and stores them in the local memory of the individual sockets. Each hardware thread runs a worker that processes requests on the local partitions, which are exclusively locked during the processing phase to eliminate the need of non-scaling fine-grained data structure latches. During query processing worker threads communicate via a message passing layer. The operating system leverages and maintains workload-dependent energy-utility functions that reflect the trade-off between a certain utility (e.g., query throughput) and energy consumption. An EUF is maintained per energy domain (i.e., per socket) to select the most energy-efficient hardware configuration for a specific system load. The operating system pulls monitoring information from the DBMS (e.g., query throughput and worker utilization) and from the hardware performance counters (e.g., memory bandwidth) for its decision making. Energy-Utility Functions (EUF). An EUF consists of configurations representing a specific system state in terms of hardware energy-control settings for a single socket. Configurations itself are workload-agnostic, but exhibit different energy and utility characteristics when being evaluated in the context of a specific workload. Hence, a configuration is expressed as:

$$c_x = (\{\text{core}_{\text{HyperThread}}\}, \{(\text{core}, f_{\text{core}})\}, f_{\text{uncore}})$$
(1)

For instance a configuration c_1 can be instantiated as:

$$c_1 = (\{1_1, 1_2, 2_1\}, \{(1, 1.2 \text{ GHz}), (2, 2.1 \text{ GHz})\}, 3 \text{ GHz})$$
 (2)

This configuration activates the first physical core and both HyperThread siblings as well as the second physical core with one HyperThread. The core frequency of the first physical core is set to 1.2 GHz and the clock of the second core is set to 2.1 GHz. The uncore clock (operating frequency of caches and memory controllers) is pinned to 3 GHz. During the evaluation process of a configuration, it is enriched with the utility metric and a power consumption. A covering set of configurations create an EUF as shown in Figure 2. The color of the outer circles encodes the average core frequency (ranging from green to red), the inner color the uncore frequency, and the diameter the number of active cores. The x-axis indicates the *utility* as the utility metric normalized to the peak utility. The y-axis shows the energy efficiency of a configuration normalized to the measured peak energy efficiency. Database systems without energy-control mechanisms usually use all available cores at the highest frequency as long as enough work is available, which is known as race-to-idle (RTI). Therefore, the baseline in the figure shows the respective energy efficiency that is achieved for different performance demands using this approach. Obviously, a more energy-efficient way is using a RTI strategy that switches between idle mode and the most energy-efficient configuration, which is depicted as the ECL RTI line. In general, only the skyline configurations are of interest for our energy-control mechanism, because they are the most energy-efficient ones for a specific utility.

As shown by Figure 2, EUFs are workload-dependent. For instance, Figure 2(a) show the EUF for a compute-bound workload (all threads increment a local counter). However, the shape of an EUF starts to change as soon as hardware resource contention is created as shown in Figure 2(b) and 2(c). In the first figure, memory controller contention occurs (i.e., a parallel column scan) resulting in configuration clusters with the same uncore frequency or cacheline contention that is generated in the latter EUF. As shown, hardware resource contention usually widens the gap between the configurations regarding their energy efficiency.

EUF Decomposition. To evaluate the energy and utility metric of a configuration, we need to map all possible workload types and configurations to their respective energy draw. Due to the vast configuration space, this becomes quickly unfeasible to benchmark



Figure 3: Web-based demo user interface.

for all imaginable workloads. We argue that re-benchmarking all possible configurations upon a workload change is too invasive and costly for a practical application. To overcome this issue and be able to quickly adapt a socket's EUF, we take the step to decompose an EUF into a calibrated software model and hardware energy model. This approach allows us to generate the software and hardware models separately. In particular, we benchmark the hardware once, using a wide range of benchmarks that are independent of the actual workload, but are able to trigger many different usage characteristics of the CPU. This step yields a hardware energy model that maps various CPU resource usage patterns to the power draw of the CPU. Accordingly, to create the software model it is no longer necessary to actively apply various hardware configurations, since this configuration space is already covered by the hardware model. Instead, we choose a fixed hardware configuration and monitor the behavior of the changed DBMS workload. This step provides us with a software model that maps the DBMS utility (e.g., query throughput) to the corresponding CPU resource usage pattern. Combining both models allows accurate prediction of power consumption and utility.

The main challenge using the aforementioned approach is to find an abstraction between the hardware energy model and the software model that is versatile enough to express different workloads and at the same time abstract enough to avoid hardware-specific metrics in the software model. For instance, the key feature set for CPU power draw are frequencies, active cores (hardware configuration) and the instruction mix issued by the application. Memory bandwidth-intensiveness and the types of computations used are key factors that define the instruction mix and thus the power draw of a specific hardware configuration. Vector instructions for instance use more power than basic ALU instructions, while the CPU draws considerably less power when mostly waiting for memory. A CPU hardware energy model is thus a mapping from these metrics to power described by the following equation:

 $P_{CPU} = f(c_x, \text{memory bandwidth}, \text{vector instruction ratio})$ (3)

The hardware energy model needs to be trained for each configuration (c_x) with a set of workloads, which adequately capture the memory bandwidth and vector instruction ratio metrics.

The corresponding software model maps the software-specific performance metric, such as query throughput, to the characteristics of the instruction stream needed by the hardware model. We found that a good - hardware-independent - metric for the software to specify the two key features is the number of memory-accesses per retired instruction and the number of vector instructions per retired instruction. These metrics are only dependent on the programs instruction stream and the cache-size, which are both constant for a given workload in our setup as we neither recompile our software nor migrate to a different machine. So a software model that characterizes the database can be described by the following metrics which are functions of the workload and the database configuration: (1) instructions per query, (2) memory accesses per instruction, and (3) vector instructions per instruction. These two models then can be combined to compute the EUFs seen in Figure 2. If the workload composition of the DBMS changes during runtime, the operating system monitors the software model metrics and the DBMS executions statistics and re-adjusts the software model accordingly without the need to re-benchmark all configurations.

3 DEMO DESCRIPTION

In this section, we describe our overall demo setup and especially focus on the interaction points, visualizations, and interesting scenarios for demo attendees. The database system is executed on a 2-socket Intel server system running a Linux with our EUF-based power management component and features the web-based demo interface for interaction shown in Figure 3. The main objective of the demo is to show the audience how the system responds to a changing workload type and system load by visualizing internal structures such as calculated energy-utility functions, the current hardware configuration, and important metrics such as power draw and query latency. In the following, we describe the individual interaction points and visualizations in detail:

- (1) Workload. This interaction point allows the attendee to chose from a set of predefined workloads. This workload set includes corner case workloads such as the indexed and non-indexed keyvalue store workloads generating an either memory latency-bound workload or a memory bandwidth-bound workload. Moreover, the set includes typical database benchmarks that generate OLTP or OLAP-like workloads. Changing the workload during the demo causes an adaptation of the EUFs on both sockets (shown in 7), usually resulting in a hardware reconfiguration (shown in 6).
- (2) Load Profile. While the workload describes the types of queries executed by the DBMS, the load profile describes the system load in number of queries per second. The load profile allows us to measure the energy efficiency for different loads, which effectively shows the energy proportionality of the database system. The attendee is able to switch between two static load profiles (low and high load) as well as two dynamic load profiles executing either a spike load to show the energy proportionality or the twitter load profile to simulate a real-life load scenario. Changing the system load typically results in a changing hardware configuration (6) that is chosen from the socket-local EUFs (7).
- (3) Load & Power Monitor. This visualization shows the live power and system load measurements over time. Attendees are able to observe the correlation between system load and power draw effectively showing the energy proportionality as well as the adaptation speed to a changing load. Moreover, this area shows the power savings of our approach if used in combination with energy-control master switch (5).
- (4) Query Latency Monitor. Since energy savings are usually a result of trading energy for query execution latency, this monitor visualizes the current average query latency relative to the user-defined latency limit. While the green zone indicates a good and acceptable latency, the red zone indicates a violation of this limit. The demo audience can observe how the latency fluctuates during sudden system load peaks, while our energy-control mechanism tries to keep the query latency below the preset limit.
- (5) Energy-Control Master Switch. The energy-control master switch either enables our approach or disables it by switching back to traditional operating system power management without energy-utility function assistance. Switching between both modes causes an increased power draw visible in the power monitor (3) as well as increased core and uncore frequency as it is visualized in the hardware configuration (6). Attendees can use this switch to see the benefits of our energy-control mechanism in terms of energy efficiency and energy proportionality.
- (6) Hardware Configuration. The hardware configuration monitor effectively visualizes the topology of the system as well as its configuration in terms of energy-control knobs. In particular, we schematically show both sockets of the systems including the

available hardware threads. Each column shows the hardware threads that share the same physical core (simultaneous multithreading). The bar within each hardware thread visualizes the utilization of a hardware thread (height of the bar) and the frequency setting of the underlying physical core (color of the bar). Hardware threads without a visible bar are currently turned off and reside in a sleep state (C-state). The border color of a socket visualizes the current uncore clock of the socket. In general, the hardware configuration monitor visualizes the current hardware configuration as it was picked from the EUF (7) respectively from the traditional operating system mechanism in case of a turned off master switch (5). With the help of this monitor, demo attendees are able to observe how the configuration changes in case of an increasing system load (2) or a changing workload (1).

(7) Energy-Utility Functions. The EUF monitor visualizes the energy-utility functions for each socket as they were calculated by the operating system using the decomposition approach. The EUFs of the sockets may differ, because of an asymmetric workload or due to different energy characteristics of the processors as a result of the manufacturing process. Each EUF employs an indicator to show the currently selected skyline configuration that is applied to the hardware. Additionally, the audience is able to see the shape of the current EUFs and is able to watch how the EUFs adapt in case of a changing workload (1).

4 CONCLUSIONS

Energy is the key-limiter for the scalability of scale-up database systems. This demo aimed at reducing the energy consumption of data-oriented in-memory database systems that make heavy use of the main power consumers, namely CPU cores and main memory. Our approach relies on energy-utility functions that are efficiently obtained using a combination of a calibrated software model and hardware energy model. While we acknowledge that our decompositional approach leads to a slight loss of accuracy of the EUF, results show that these accuracy losses are in the low single digit percent range. We deem the benefit of being able to directly generate EUFs without costly re-benchmarking worth the cost. Our sophisticated demo setup offers multiple interaction points and visualizations to show how our approach behaves in case of a changing workload type or system load to provide demo attendees with valuable insights into EUF-based power management.

5 ACKNOWLEDGMENTS

This work is partly funded within the DFG-CRC 912 (HAEC).

REFERENCES

- Hermann Hartig et al. 2013. The case for practical multi-resource and multi-level scheduling based on energy/utility. In RTCSA.
- [2] Thomas Kissinger et al. 2014. ERIS: A NUMA-Aware In-Memory Storage Engine for Analytical Workloads. In ADMS.
- [3] Thomas Kissinger et al. 2018. Adaptive Energy-Control for In-Memory Database Systems. In SIGMOD.
- [4] Ippokratis Pandis et al. 2010. Data-Oriented Transaction Execution. PVLDB (2010).
- [5] Danica Porobic et al. 2014. ATraPos: Adaptive Transaction Processing on Hardware Islands. In ICDE.