Annett Ungethüm, Thomas Kissinger, Willi-Wolfram Mentzel, Dirk Habich, Wolfgang Lehner

Energy Elasticity on Heterogeneous Hardware using Adaptive Resource Reconfiguration LIVE

Erstveröffentlichung in / First published in:


DOI: https://doi.org/10.1145/2882903.2899390

Diese Version ist verfügbar / This version is available on:

https://nbn-resolving.org/urn:nbn:de:bsz:14-qucosa2-803698
Energy Elasticity on Heterogeneous Hardware using Adaptive Resource Reconfiguration LIVE

Annett Ungethüm, Thomas Kissinger, Willi-Wolfram Mentzel, Dirk Habich, Wolfgang Lehner
Database Technology Group
Technische Universität Dresden
01062 Dresden, Germany
{firstname.lastname}@tu-dresden.de

ABSTRACT

Energy awareness of database systems has emerged as a critical research topic, since energy consumption is becoming a major limiter for their scalability. Recent energy-related hardware developments trend towards offering more and more configuration opportunities for software to control its own energy consumption. Existing research so far mainly focused on leveraging this configuration spectrum to find the most energy-efficient configuration for operators or entire queries. In this demo, we introduce the concept of energy elasticity and propose the energy-control loop as an implementation of this concept. Energy elasticity refers to the ability of software to behave energy-proportional and energy-efficient at the same time while maintaining a certain quality of service. Thus, our system does not draw the least energy possible but the least energy necessary to still perform reasonably. We demonstrate our overall approach using a rich interactive GUI to give attendees the opportunity to learn more about our concept.

Keywords
Energy, Adaptivity, Heterogeneity, Energy Elasticity

1. INTRODUCTION

Energy consumption is becoming a major limiter for scalability on today’s server systems and thus for the data management software running on such hardware. For that reason, energy awareness of database systems was identified as a critical research topic some years ago [1], but the absence of energy-aware hardware strongly limited the optimization potential of the software [5]. Recently, energy-related technology that originates from the mobile devices sector started to make its way into desktop and server systems. Those hardware developments, especially a more fine-grained dynamic voltage and frequency scaling (DVFS) as well as sleep states, allow an energy-proportional behavior of the hard-

©2016 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 ’16 June 26–July 1, 2016, San Francisco, CA, USA.
DOI: http://dx.doi.org/10.1145/2882903.2899390

Figure 1: Architectural Overview.

ware. However, to fully utilize these energy control knobs offered by the hardware, software has to make appropriate use of them. Following the current hardware development trends, even an ARM big.LITTLE-like design, which adds stripped versions of traditional fat cores to the system, will appear in future server systems opening up an additional dimension for the energy optimization process.

In this demo, we propose the energy-control loop, which addresses the topic of software-controlled hardware configurations at runtime for data management systems running on a single heterogeneous server system optionally employing a big.LITTLE design. While existing works [3, 4] mainly try to find the most energy-efficient configuration for a given workload, our approach not only aims at achieving energy-proportionality, but also considers energy-efficiency as a secondary optimization target. This way, our energy-control loop is able to run the system in an energy-efficient configuration while still being able to maintain certain query latency constraints, especially in times of heavy load. Since the workload is a moving target, the loop is continuously running and adapts the hardware configuration. We call this concept energy elasticity.

Figure 1 shows the overall architectural overview of the system. The platform that is running the DBMS is exposed to a dynamic workload and has to fulfill certain soft constraints in terms of maximum query response time respectively query latency. While processing the workload, the energy-control loop is continuously monitoring the DBMS to detect the required performance demand that is necessary to meet the latency constraints. If this performance demand increases or decreases, the loop applies another energy-efficient configuration suitable for this case at runtime. This config-
2. ENERGY ELASTICITY

With the term energy elasticity, we refer to the ability of software to (1) behave energy-proportional and (2) energy-efficient at the same time while (3) maintaining a certain quality of service. Up to now, research mostly focused on finding the most energy-efficient configuration for a specific query offering a binary decision between a responsive performance mode or a slower energy-efficient query processing mode.

Our approach breaks with this binary decision and allows the database system to scale between both processing modes and is thus primarily enabling fine-grained energy proportionality meaning that system load and energy consumption strongly correlate with each other. The main indicator for the selection of the proper hardware configuration is the tolerated average query latency and thus the amount of performance that is necessary to hold this constraint. Besides solely aiming for energy proportionality, our energy-control loop applies the most energy-efficient hardware configuration that is able to meet the given query latency respectively performance demands.

2.1 Energy Profile

The operation of the energy-control loop depends on the existence of an energy-profile for the target platform. Such an energy profile has to be produced once and usually behaves equally for the same hardware platform. An energy profile basically consists of exploration points that contain the specific configuration as well as energy and performance measurements. A configuration itself usually comprises:

(1) A list of active cores with benchmark/operator type mapping.

(2) The frequencies for all available cores, because the frequency of sleeping cores sometimes influences the performance and power consumption of others.

To produce an energy profile for a platform, all reasonable configurations have to be generated and benchmarked. For instance, the ODROID-XU3 as our demo platform features one cluster of four LITTLE ARM Cortex A7 (frequency range from 200MHz to 1.4 GHz) and one cluster of four big ARM Cortex A15 (frequency range from 200MHz to 2 GHz) as well as a Mali GPU, which will not be subject of the demo. Both core clusters implement the same instruction set architecture (ISA) but differ from each other in their instruction execution pipeline, last level cache size, and frequency range. This platform allows 24 active core combinations and 70 frequency combinations (200 MHz steps; shared clock per core cluster) which amounts to a total configuration space of 1680 exploration points. However, the quality of the energy profile also depends on the characteristics of the benchmark that is executed for the different configurations.

Figure 2 visualizes the full energy profile for our demo setup using the mentioned benchmark respectively operator types. The configurations are ordered by their overall performance (measured in the number of performed scans respectively computations), which is the sum of the compute and sequential read performance (normalized to the single LITTLE core performance to make both benchmarks comparable). An example configuration for the ODROID-XU3 platform looks like this:

(1) Cluster 1 (2x seq. read, 1x compute, 1x sleep),
Cluster 2 (2x compute, 2x sleep)

(2) Cluster 1 (800 MHz),
Cluster 2 (1,000 MHz)

The energy consumption was measured using the sum of all built-in power sensors. Using the normalized benchmark performance result and the measured energy consumption,
the energy efficiency is calculated as performance/energy. The chart shows a nearly linear performance scaling and a high variety regarding the energy efficiency of the configurations.

2.2 Energy-Control Loop

The energy-control loop is designed following the principle of a reactive control loop. Every time the loop is triggered, it firstly collects the current monitoring information that basically includes the current latencies for the different query classes (read or compute intensive on our case). Afterwards, it compares the latencies with the given latency constraints for the query classes and either demands less or more read respectively compute performance. Using this reduced or increased performance demands, it searches the energy profile for configurations that fulfill those demands for both query classes and selects the configuration that requires the least energy.

Finally, the energy-control loop applies the new hardware configuration using the facilities exposed by the operating system. Additionally, the loop instructs the database operator scheduler to schedule queries of a respective query class on the cores as specified by the configuration. Since the loop executes a pretty lightweight process, it can be triggered multiple times per second and is thus very responsive to cope with highly volatile workload characteristics.

3. DEMONSTRATION

In this section, we explain our overall demonstration setup and discuss the interactive opportunities that our demo GUI offers to the attendees of the demo.

3.1 Setup

For this demonstration we employ an ODROID-XU3, which is based on the ARM big.LITTLE design, running an Ubuntu 14 Linux. As already mentioned, the chip features one cluster of four LITTLE ARM Cortex A7 (maximum frequency of 1.4 GHz) and one cluster of four big ARM Cortex A15 (maximum frequency of 2 GHz) as well as a Mali GPU, which will not be subject of the demo. The SoC containing both core clusters is a Samsung Exynos 5422 operating in the heterogeneous multi-processing mode (HMP), which allows us to simultaneously schedule threads on both kinds of cores and thus to selectively power specific core combinations.

In contrast to its predecessors, the ODROID-XU3 also contains four current and voltage sensors. They are used in this demo for individually measuring the power consumption of the A15 and A7 clusters, the GPU, and the DRAM. The average of the measurements is periodically written to dedicated system files every 0.26 s.

Because of the absence of built-in sensors, existing works [2, 3] used a power meter between the wall socket and the AC input of the board to quantify power consumption. This approach has two major disadvantages: First, the power consumption of the A15 and A7 clusters cannot be analyzed independently from each other or from the other elements on the board (e.g. ethernet interface and USB controller) Second, the consumption of the wall plug transformer is included in the measurement. However, energy conversion efficiency varies not only between different wall plug transform-
ers, it is also dependent on the drawn power. Thus, an overhead is created which is not constant. As already mentioned, this setup does not consider the fixed base power consumption of auxiliary components which effectively shortens the absolute energy efficiency range of the power profile, however, the most energy-efficient configurations will stay the same.

The demo software itself runs a basic database system that allows column aggregations and compute-intensive queries. The user is able to interactively adjust the workload in terms of overall system load, query mix, and desired query latency. The energy control loop automatically applies the appropriate hardware configuration and thus implements energy elasticity. The monitoring information is gathered from the database system and is visualized in the demo UI.

3.2 Walkthrough

Figure 3 shows a screenshot of our demo application, which is composed of the following configuration options and visualizations.

The Power and Workload Monitor
The attendee is able to choose whether he wants to watch the detailed power monitoring information, which displays the power consumption for each component individually, or the overall power consumption in comparison to the current workload (6) to get an impression of how quickly and accurately the energy control loop adapts the hardware configuration. The selected curves are shown in (1) and are updated smoothly in real-time. In the detailed power view, the graph for each hardware component can be enabled or disabled individually. Additionally, the total energy consumption for each element is shown (7). The energy calculation begins at the start of the application and can be reset at any time. This way, the attendee is able to quantify the overall energy savings made by our approach.

Workload Selection
We provide three options for changing the workload (3). The first one is the overall system load, i.e., the number of incoming queries per second, which affects the performance that is required to meet the latency constraint. The second one is the read-compute ratio, which is the ratio of read-intensive and compute-intensive queries. Since the required performance varies depending on the workload characteristics, this option affects the resulting hardware configuration when we apply our concept of energy elasticity.

However, usually there is a limit for the response time a user regards as acceptable, i.e., he demands a minimum execution performance. For this reason the average response time can be manipulated, too, as a third option, which effectively enables energy elasticity, because otherwise the system would stay either in full performance or the most energy-efficient mode. This way it could either not maintain the required average query latency or is at risk of choosing a configuration which draws more energy than necessary. To observe the query processing state and get an impression of the current degree of adaptation, the number of active and finished queries as well as the currently measured response time is displayed (8).

Resource Configuration
According to the requested performance, our application shows the current configuration that was picked from the energy profile (2) and chooses the most energy-efficient configuration within a range of the desired performance. In the energy profile the performance is shown in ascending order. The dots represent the respective energy efficiency. A vertical line marks the configuration which is currently in use. Additionally, the energy efficiency is highlighted with a circle.

The individual core usage (5) shows the characteristics of the current configuration. The user can observe which cores are active, how the compute- and read-intensive operations are distributed and at which frequency the cores are running. For comparative purposes the energy-control loop can be disabled (4). In this case the management of the CPUs (i.e., the scheduling and frequency settings) is completely left to the operating system.

4. CONCLUSIONS

In this demo proposal, we addressed the challenge of achieving both facets of energy awareness – energy-proportionality and energy-efficiency – simultaneously while still regarding user-defined query latency constraints. We presented how to implement this concept of energy elasticity using the lightweight energy-control loop which depends on an energy profile. The energy-control loop leverages the configuration opportunities offered by modern hardware, which is reconfigured at runtime to adapt its power consumption proportionally to the current workload. For demonstration purposes, we employ a portable heterogeneous platform and a rich interactive GUI to give attendees the opportunity to gain more insight into our approach.

5. ACKNOWLEDGMENTS

This work is partly funded by the German Research Foundation (DFG) in the Collaborative Research Center 912 “Highly Adaptive Energy-Efficient Computing”.

6. REFERENCES