Annett Ungethüm, Patrick Damme, Johannes Pietrzyk, Alexander Krause, Dirk Habich, Wolfgang Lehner

Balancing Performance and Energy for Lightweight Data Compression Algorithms

Erstveröffentlichung in / First published in:

New Trends in Databases and Information Systems: ADBIS 2017 Short Papers and Workshops. Nicosia, 24.-27.09.2017. Springer, S. 37-44. ISBN 978-3-319-67162-8.

DOI: http://dx.doi.org/10.1007/978-3-319-67162-8_5

Final edited form was published in "New Trends in Databases and Information Systems: ADBIS 2017 Short Papers and Workshops. Nicosia 2017", S. 37-44, ISBN 978-3-319-67162-8
http://dx.doi.org/10.1007/978-3-319-67162-8_5

Balancing Performance and Energy for Lightweight Data Compression Algorithms

Annett Ungethüm(✉), Patrick Damme, Johannes Pietrzyk, Alexander Krause, Dirk Habich, and Wolfgang Lehner

Database Systems Group, Technische Universität Dresden, Dresden, Germany
{annett.ungethuem,patrick.damme,johannes.pietrzyk,alexander.krause,
dirk.habich,wolfgang.lehner}@tu-dresden.de

Abstract. Energy consumption becomes more and more a critical design factor, whereby performance is still an important requirement. Thus, a balance between performance and energy has to be established. To tackle that issue for database systems, we proposed the concept of work-energy profiles. However, generating such profiles requires extensive benchmarking. To overcome that, we propose to approximate work-energy-profiles for complex operations based on the profiles of low-level operations in this paper. To show the feasibility of our approach, we use lightweight data compression algorithms as complex operations, since compression as well as decompression are heavily used in in-memory database systems, where data is always managed in a compressed representation. Furthermore, we evaluate our approach on a concrete hardware system.

Keywords: Energy efficiency · In-memory databases · Compression

1 Introduction

Database systems constantly adapt to new hardware features to satisfy performance demands [7,8,14]. However, these features do not only influence the performance, but also the energy consumption [3]. As energy consumption becomes more and more a critical factor [5], a balance between performance and energy has to be established [12]. To tackle this balancing challenge in a fine-grained way for in-memory database systems, we proposed the concept of *work-energy profiles* in a previous work [13]. A *work-energy profile* exposes a relation between reachable performances and the resulting energy-efficiency, thereby a profile covers a wide range of CPU features, such as different vector extensions, multithreading, CPU pinning, and frequency scaling [13]. Thus, these *work-energy profiles* can be used to determine the most energy-efficient hardware configuration for any required performance demand. However, generating *work-energy profiles* requires extensive benchmarking and must be done once for every database operator and every possible hardware configuration [12,13]. Therefore, the number of necessary benchmark tests can quickly add up to thousands or millions. Even if every

Final edited form was published in "New Trends in Databases and Information Systems: ADBIS 2017 Short Papers and Workshops. Nicosia 2017", S. 37-44, ISBN 978-3-319-67162-8
http://dx.doi.org/10.1007/978-3-319-67162-8_5

test needs only a one second micro benchmark, a full benchmark for a database system on a specific hardware system would need hours or days.

Our Contributions and Outline. To overcome this benchmarking overhead, we present a novel approach to approximate work-energy profiles of *complex operations* from work-energy profiles of *low-level operations* in this paper. In detail, our contributions are: (i) We briefly summarize the core concept of our *work-energy profiles* in Sect. 2. In particular, we state which low-level profiles are necessary for our approximation approach. (ii) To illustrate our approximation approach, we use a lightweight data compression algorithm and we describe the concrete algorithm in Sect. 3. (iii) Then, our approximation approach for complex operations is introduced in Sect. 4. Thereby the approximation is based on linear combination of work-energy profiles of low-level operations. (iv) We evaluate our approach on a concrete hardware system to show the feasibility in Sect. 5. (v) We conclude the paper with related work and a summary in Sects. 6 and 7.

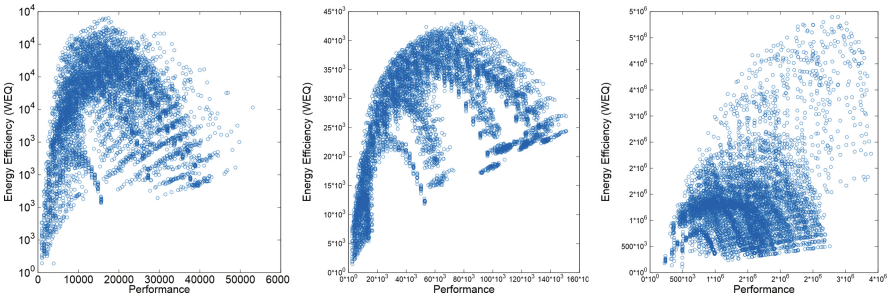
2 Work-Energy-Profiles

Modern hardware, especially CPUs, offers a lot of features like vectorization [3], multithreading, or frequency scaling [9]. These features usually have an influence on performance as well as energy consumption. However, the mapping between hardware configurations – meaning which features to which extend should be used in which way –, performance, and energy-efficiency is not trivial [13]. To capture these effects for all possible hardware configurations, we proposed the concept of *work-energy profiles* in [13].

A *work-energy profile* is a set of the useful work done during a fixed time span and the required energy for this work for all possible hardware configurations [13]. Thus, the *work-energy profile* for a specific application has to be benchmarked on a concrete hardware system [13]. Figure 1 shows three different example *work-energy profiles*, which we measured on a concrete hardware. While the performance is plotted on the x-axis, the y-axis shows the energy-efficiency. Each dot in this graph represents a specific hardware configuration. We measured the performance as *work done* per *second* and the energy-efficiency as *work done* per *Joule* (work energy quotient – WEQ). Hence, a work-energy-profile is a set of (performance,WEQ)-tuples, each of them representing one specific hardware configuration. As we can see in Fig. 1, different hardware configurations offer the same performance range with a high variance in the energy-efficiency. To balance performance and energy, the hardware configuration with the highest *energy-efficiency* within a desired performance range should be used for application execution. This hardware configuration can be extracted from our *work-energy profile*.

Generally, our main focus is on energy-efficient in-memory database systems [12]. Here, the performance and energy-efficiency of a hardware configuration depends on a multitude of factors (e.g., data characteristics and size, operator types, etc.). Moreover, main memory bandwidth and latency are limiting factors that could cause a non trivially predictable hardware behavior.

Final edited form was published in "New Trends in Databases and Information Systems: ADBIS 2017 Short Papers and Workshops. Nicosia 2017", S. 37-44, ISBN 978-3-319-67162-8
http://dx.doi.org/10.1007/978-3-319-67162-8_5



(a) Read (b) Write (c) Lookup

Fig. 1. ODROID-XU3-based work-energy profiles for different memory access patterns.

To get a deeper understanding and a specific foundation, we propose to benchmark *work-energy profiles* only for fine-grained memory access patterns – database primitives – which are highly utilized in in-memory database systems: (i) read-primitive, (ii) write-primitive, (iii) lookup-primitive, (iv) compute-primitive, and (v) processing-primitive, whereas the processing is a combination of read and compute, since for data processing a set of data has to be read first and then some kind of computation is triggered. The ratio between read and compute can vary, which we consider in our benchmark.

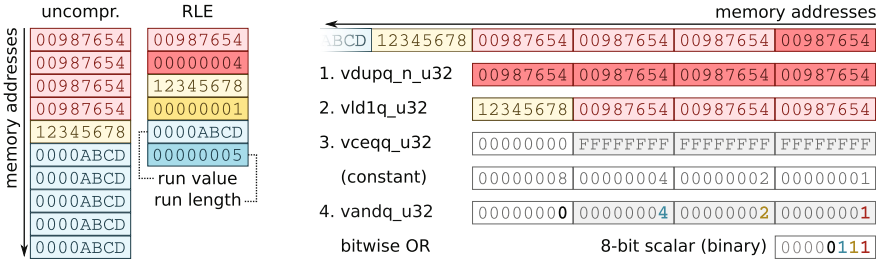
Figure 1 depicts the resulting *work-energy profiles* for these primitives on an ARM big.LITTLE hardware system. Concretely, we used an ODROID-XU3, which consists of a big and a little cluster, each of them featuring 4 cores. Additionally, the ODROID-XU3 is equipped with on-board power sensors allowing us to measure the power level of individual core clusters and the main memory separately. The different combinations of cores and their frequencies add up to roughly 6000 different configurations [13] and each dot in Fig. 1 represents a specific configuration. As we can see, the shapes of the *work-energy profiles* of our primitives are different.

3 Example Operation RLE

To describe our approximation approach, the physical operation must be precisely known. In this paper, we focus on run-length encoding (RLE), since RLE is a heavily applied compression technique in in-memory database systems [1,6]. RLE tackles uninterrupted sequences of occurrences of the same value, so called runs. In its compressed format, each run is represented by its value and length. Therefore, the compressed data is a sequence of such pairs as illustrated in Fig. 2(a). In addition to the simplicity of RLE, there are two other advantages: (i) RLE can be easily parallelized by data partitioning, so that the parallelization itself does not produce any mentionable communication overhead between the processing cores and (ii) RLE can be vectorized [2].

Our vectorized implementation consists of four steps as shown in Fig. 2(b) using the ARM NEON implementation of SIMD. In step one, four copies of

Final edited form was published in "New Trends in Databases and Information Systems: ADBIS 2017 Short Papers and Workshops. Nicosia 2017", S. 37-44, ISBN 978-3-319-67162-8
http://dx.doi.org/10.1007/978-3-319-67162-8_5



(a) Basic concept of RLE (b) RLE using 128-bit vector registers

Fig. 2. The basic idea of RLE compression is to not store every value individually, but only once followed by the number of sequential elements with the same value.

the current input element are loaded into one 128-bit vector register using the `vdupq_n_u32()` NEON operation. In Step two, the next four input elements are loaded into a second 128-bit vector register using the `vld1q_u32()` intrinsic. In step three, these four values are compared in parallel with the current run value using `vceqq_u32()`. The result is stored in a third vector register. In each 32-bit element of this vector register, either all bits are set or all bits are not set, depending on whether the corresponding elements were equal or not. In step four, from this register we extract a bit mask by ANDing it with the constant vector $(1, 2, 4, 8)$ using `vandq_u32()`, storing the result to memory using `vst1q_u32()`, and sequentially ORing the lowest bytes of all four elements. The number of trailing ones in this bit mask tells us for how many elements the current run continues. We look up this number in a table created offline and indexed with the 16 possible masks. If the obtained number is four, then we have not seen the run's end yet, and continue at step 2. Otherwise, we have reached the run's end and append the run value and run length to the output and continue with step 1 at the next element after the run's end.

4 Work-Energy-Profile Approximation

Unfortunately, the benchmarking of a *work-energy profile* for a specific primitive on the ODROID-XU3 takes about eight hours. To do this for all physical database operators or even queries would be way too much overhead. However, the profiles are necessary to determine the most energy-efficient hardware configuration for a demanded performance. To tackle this challenge, our idea is to approximate the *work-energy profiles* of complex operations from the profiles of these low-level primitives. In this section, we want to demonstrate that our idea is feasible using RLE as an example.

4.1 RLE and Low-Level Operations

As just described, our vectorized RLE algorithm is fixed, but the input data determines the execution behavior, thereby two extremes arise. One extreme is

Final edited form was published in "New Trends in Databases and Information Systems: ADBIS 2017 Short Papers and Workshops. Nicosia 2017", S. 37-44, ISBN 978-3-319-67162-8
http://dx.doi.org/10.1007/978-3-319-67162-8_5

obtained if there is no run in the input data at all (average run length equals 1). In this case, RLE-compressed data is twice as large as the original data, because for each array element, a run length of 1 is additionally stored. This means for the processing, that our vectorized compression algorithm performs essentially random reads and random writes with a ratio of 1:1. Random reads, since we always read 4 elements in each iteration, with 3 of them already being read in the previous iteration, producing overlapping reads. The second extreme occurs when each element in the uncompressed data array equals a single value (average run length equals number of elements). In this case, RLE-compressed data consists of two values, the single value and the number of elements as run length and these two values are written once by the compression algorithm. Thus, the read/write ratio approaches 1:0, while the read accesses are still random. Furthermore, between the reads and writes, there is also the actual compression which is a computation bound work. If this computation is slower than the I/O accesses, the memory access pattern is not the bottleneck for the performance anymore, but the computation itself.

Therefore, three low-level operations are used within our vectorized RLE compression as well as decompression algorithms: (i) read, (ii) write, and (iii) compute. Depending on the input data, these operations are composed differently. While the number of compute operations per read operation is constant, the ratio between read and write operations changes depending on the average run length. Hence, the profile for a specific average run length is within the spectrum between read-bound and write-bound operations.

4.2 Approximation Using Linear Combination

To approximate *work-energy profiles*, we propose to combine low-level primitives in a linear way. The new profile, containing (performance, WEQ)-tuples for i different configurations, is obtained from j low-level profiles and the tuples of the new profile are determined by

$$(Performance, WEQ)_{i,new} = f^{-1}(\sum_{0}^{j-1} w_j * f(Performance, WEQ)_{i,j}) \quad (1)$$

where w_j is a weighting factor which describes the influence of the profile j. The adjustment function f modifies the performance- and WEQ-values for the combination. This is necessary if the limiting factors, i.e. the low-level-profiles, do not scale linearly when the parameters of the operation are changed.

For RLE compression, the only available parameter is the average run length. This parameter defines the number of read and write accesses, which define the scaling of the *work-energy profile*, i.e. the adjustment function f. Whereas the ratio between the read and write accesses defines the weighting factors w_j. The number of read or write accesses as a function of the run length rl, can be extracted from the vectorized algorithm presented in Sect. 3. As shown in Eqs. 2 and 3, we denote to the number of reads and writes as $count_{reads}$ and $count_{writes}$ respectively. A sequence of run lengths is described as $RL = [rl_0, rl_1, rl_{|RL|-1}]$

Final edited form was published in "New Trends in Databases and Information Systems: ADBIS 2017 Short Papers and Workshops. Nicosia 2017", S. 37-44, ISBN 978-3-319-67162-8
http://dx.doi.org/10.1007/978-3-319-67162-8_5

with $count_{runs} = |RL|$ runs, and k is the vector width. For a constant run length, $count_{runs}$ can be computed by $count_{elements}/rl$.

$$count_{reads} = \sum_{i=0}^{|RL|-1} (2 + \lfloor \frac{rl_i - 1}{k} \rfloor) \tag{2}$$

$$count_{writes} = 2 * count_{runs} \tag{3}$$

Both functions are rational. Hence, our function f must be rational, too. Since there are no polynoms or exponential parts in either $count_{reads}$ or $count_{writes}$, it is safe to use the most simple rational function $f(Performance, WEQ) = (1/Performance, 1/WEQ)$ for Eq. 1.

The ratio between the read and write operations, and therefore the weighting factors w_j, follow from the Eqs. 2 and 3 as well:

$$w_0 : w_1 = 2 * count_{runs} : \sum_{i=0}^{|RL|-1} (2 + \lfloor \frac{rl_i - 1}{k} \rfloor).$$

5 Evaluation

To validate the results of our profile approximation approach, we also benchmarked the profiles on the ODROID-XU3. For comparing the quality of an approximated profile to a benchmarked profile, we filtered the configurations which are part of the Pareto front of the measured and the approximated profile. Then, we divided the profile into ten performance ranges and compared the approximated and measured configurations which are the closest to the middle of these performance ranges. As a result, there are two measures to quantify the quality of the approximated profile: (1) the chance that the approximated configuration actually is in the performance range, in which we expect it to be, and (2) the mean deviation of the WEQ from the measured configuration in the middle of this performance range.

Figure 3 shows the results for our RLE compression algorithm applied on synthetic data containing values with an average run length of 45. The figure shows that the match of the approximation and the benchmarked profile is not exact but close to the optimal solution. To quantify the difference, we calculated the two measures as described above: (1) The chance that a configuration, which we expect to be close to the middle of a specified performance range, is actually in this performance range, is 100% in our example. This means, that all ten configurations, we filtered for the ten performance ranges were actually within a 10% radius of this performance range. (2) The mean deviation of the WEQ from the measured configuration was only 3%.

We conducted the evaluation on different data sets as well as on different hardware systems. In almost all cases, we observed a similar behavior, so that we are able to conclude that our approximation approach is well-suited for the considered operations of RLE compression and decompression.

Final edited form was published in "New Trends in Databases and Information Systems: ADBIS 2017 Short Papers and Workshops. Nicosia 2017", S. 37-44, ISBN 978-3-319-67162-8
http://dx.doi.org/10.1007/978-3-319-67162-8_5

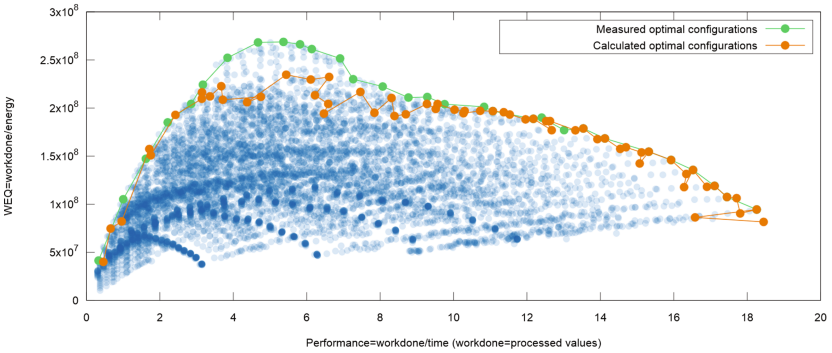


Fig. 3. A benchmarked work-energy profile for RLE compression on the ODROID-XU3 with average run length of 45. The approximated optimal configurations are highlighted in orange, the actual optimum is highlighted in green. (Color Figure Online)

6 Related Work

Lossless compression techniques play an important role in in-memory database systems [1,6]. They reduce not only the amount of needed space, but also the time spent on i/o instructions. Thus, they have already been investigated for query performance [1]. Further works on vectorized compression techniques also focus on performance [10], but not on energy efficiency. An extensive experimental evaluation on lightweight compression techniques has been done by Damme et al. [2]. However, to the best of our knowledge, none of these works explicitly regards energy efficiency. Vice versa, the works regarding energy-efficiency (1) focus on query execution, usually evaluated by running a TPC benchmark, rather than on compression itself [11], and (2) treat the performance-energy-tradeoff as a binary decision between energy-efficiency and performance [15]. In general, there are analytical models to estimate and reduce the energy consumption [15] and benchmark-based approaches [4]. The latter has only been applied for homogeneous systems and analytical models become more complex the more complex the hardware becomes. Regarding the ever-growing world of heterogeneous hardware, we developed an approach which is mostly benchmark based and applicable to various systems.

7 Conclusion and Outlook

As energy consumption becomes more and more a critical design factor, a balance between performance and energy has to be established. To tackle this challenge for in-memory database systems, we proposed the concept of work-energy profiles in [13]. A *work-energy profile* is a set of the useful work done during a fixed time span and the required energy for this work for all possible hardware configurations [13]. In this paper, we proposed to approximate work-energy-profiles for complex operations based on the work-energy profiles of low-level operations

Final edited form was published in "New Trends in Databases and Information Systems: ADBIS 2017 Short Papers and Workshops. Nicosia 2017", S. 37-44, ISBN 978-3-319-67162-8
http://dx.doi.org/10.1007/978-3-319-67162-8_5

and demonstrated the feasibility on a concrete example. In future work, we are going to generalize this approximation approach to other physical database operators. This is essential to achieve our main goal of integrating these work-energy profiles into query optimization.

Acknowledgments. This work is partly funded within the DFG-CRC 912 (HAEC) and by the DFG-project LE-1416/26.

References

1. Abadi, D.J., et al.: Integrating compression and execution in column-oriented database systems. In: SIGMOD (2006)
2. Damme, P., et al.: Lightweight data compression algorithms: an experimental survey (experiments and analyses). In: EDBT (2017)
3. Firasta, N., et al.: Intel AVX: new frontiers in performance improvements and energy efficiency. Intel White Paper (2008)
4. Götz, S., et al.: Energy-efficient databases using sweet spot frequencies. In: UCC 2014 (2014)
5. Harizopoulos, S., et al.: Energy efficiency: the new holy grail of data management systems research. In: CIDR (2009)
6. Hildebrandt, J., Habich, D., Damme, P., Lehner, W.: Compression-aware in-memory query processing: vision, system design and beyond. In: Blanas, S., Bordawekar, R., Lahiri, T., Levandoski, J., Pavlo, A. (eds.) IMDM/ADMS - 2016. LNCS, vol. 10195, pp. 40–56. Springer, Cham (2017). doi:10.1007/978-3-319-56111-0_3
7. Karnagel, T., et al.: Adaptive work placement for query processing on heterogeneous computing resources. PVLDB **10**(7), 733–744 (2017)
8. Kissinger, T., et al.: ERIS: a numa-aware in-memory storage engine for analytical workload. In: ADMS@VLDB, pp. 74–85 (2014)
9. Le Sueur, E., et al.: Dynamic voltage and frequency scaling: the laws of diminishing returns. In: Proceedings of the 2010 International Conference on Power Aware Computing and Systems, pp. 1–8 (2010)
10. Lemire, D., Boytsov, L.: Decoding billions of integers per second through vectorization. Softw. Pract. Exper. **45**(1) (2015)
11. Mühlbauer, T., Rödiger, W., Seilbeck, R., Kemper, A., Neumann, T.: Heterogeneity-conscious parallel query execution: getting a better mileage while driving faster! In: DaMoN@SIGMOD (2014)
12. Ungethüm, A., et al.: Energy elasticity on heterogeneous hardware using adaptive resource reconfiguration LIVE. In: SIGMOD, pp. 2173–2176 (2016)
13. Ungethüm, A., Kissinger, T., Habich, D., Lehner, W.: Work-energy profiles: general approach and in-memory database application. In: Nambiar, R., Poess, M. (eds.) TPCTC 2016. LNCS, vol. 10080, pp. 142–158. Springer, Cham (2017). doi:10.1007/978-3-319-54334-5_10
14. Willhalm, T., et al.: Simd-scan: ultra fast in-memory table scan using on-chip vector processing units. PVLDB **2**(1), 385–394 (2009)
15. Xu, Z., et al.: Dynamic energy estimation of query plans in database systems. In: 2013 IEEE 33rd International Conference on Distributed Computing Systems (ICDCS), pp. 83–92. IEEE (2013)