Energy-aware Data Stream Management

Maik Thiele and Wolfgang Lehner
Technische Universität Dresden
Database Technology Group
Dresden, Germany
{maik.thiele, wolfgang.lehner}@tu-dresden.de

***Abstract*—With the increasing use of data stream management systems (DSMS's) in energy-critical areas the energy consumption of the DSMS itself is gaining growing importance. However, processing data streams in an energy-aware manner is still an open research question. In this paper, we propose a very first concept for a DSMS architecture which treats the energy consumption as a first-class-citizen. Therefore, we rely on data synopses and anytime algorithms in order to minimize the data quantity and the number of iteration which minimizes the overall energy consumption.**

Keywords – Data stream management; Energy-awareness, Synopses, Anytime Algorithm

I. INTRODUCTION

Currently, there is a severe lack of infrastructures allowing us to monitor and control the energy consumption of a given system under observation (SUO) from the software level. This results in the need for a monitoring component to observe, analyze and predict the amount of consumed energy resources. Such a component would acquire the input for its energy calculations from a mass of information; these typically arrive in the form of continuous data streams and originate from various sensors spread across the whole SUO. The use of continuous data streams necessitates suitable techniques to handle the information flow and associated service level agreements. Therefore, we intend to build a data stream management system (DSMS) to analyze the energy-behavior of a SUO and which itself is 1) energy-adaptive and 2) minimal-intrusive in term of energy consumption.

Regarding energy-efficiency we want to refer to the so-called "need-to-know" principle as the opposite of the "ubiquity" principle. The "need-to-know" principle only provides as much information as needed by the application in only the quality that is required by the application. In contrast, traditional data management systems follow the "ubiquity" principle by always presenting an up-to-date and consistent database even if no one is currently accessing the database. Through the consistent application of the "need-to-know" design principle to the data processing, we can minimize the quantity and granularity of the data in question and we can prematurely abort the execution of operations as soon as a defined quality has been reached. Both optimization criteria directly contribute to the minimization of the system's energy requirements.

In this paper, we will describe the architecture and design principles of a DSMS based on data synopses and anytime algorithms which are both ideally suited for an energy-aware data stream processing.

II. ENERGY-AWARE DATA MANAGEMENT

Despite the increasing research efforts in power management techniques, there has been little work to date on energy efficiency from a pure data management perspective. A recent study [1] has shown that, contrary to what previous work has suggested [2, 3, 4], within a single node intended for use in shared-nothing architectures, the most energy-efficient configuration is typically the one with the highest performance. In the few cases where this correlation does not hold, the improvements in energy efficiency are less than 10%, mainly due to the large up-front power costs in current server components. A major reason for this effect is given by the lack of so-called energy-proportional hardware [5], which only uses power in constant proportion to its performance. However, there have been some recent developments in this regard (e.g. PCRAM, finer CPU power states, SSDs, etc.) which will soon allow the energy-proportional operation of the hardware as a base of an energy-efficient operation.

But hardware-only approaches are just one part of the solution, and data management and analytics infrastructures will play a key role in optimizing for energy efficiency. In [1], the authors found that common database operations can exploit the full power range of a server and further detected that the CPU power consumption of different operators, for the same CPU utilization, can differ by as much as 60%. The authors stated that the most promising software knobs are the ones that can directly trade CPU cycles for disk access time, since these are two resources with significantly different power-usage profiles. Such trade-offs exist in access methods, compression techniques, and join algorithms. Given the increase in hardware heterogeneity in combination with energy proportionality, we can expect further opportunities here to identify optimal configurations; those should be exploited via specialized data management operators. For this purpose, we need an energy model that describes the different hardware resources with their interdependencies on the one hand and that comprises the

 ISBN: 978-1-61208-136-6

resource allocation of the different algorithms and methods on the other hand. First approaches in this regard with a focus on flash memory can be found in [6].

III. DSMS'S IN THE CONTEXT OF ENERGY ADAPTATION

A variety of research activities have already been directed towards different aspects of managing data streams: DSMS like Aurora [7], STREAM [8], PIPES [9], Gigascope [10], TelegraphCQ [11] and NiagaraCQ [12] represent major examples developed over the last years. Furthermore, there exists our QStream DSMS [13] where we already investigated quality-aware data streams using a hard real-time processing model. This model should be extended by an additional energy quality dimension.

The main characteristics of data streams normally do not allow the persistent storage and the subsequent time-consuming processing within a traditional data management system. Instead, the goal is a data-driven processing of the potentially unlimited data streams, where systems should behave adaptively, e.g., with regard to the current data arrival rates. That means, data must be provided as quickly as possible, in the sense of a data stream system and on the other hand, complex analyses are required, which necessitates storing large amounts of data. To materialize stream data, synopses are typically used, which will be briefly discussed in the next paragraph.

IV. ENERGY-AWARE DATA SYNOPSES

Gibbons et al. defined the term synopsis in [14] as a "random data structure" that is substantially smaller than the base data. Due to their small size, synopses come with various benefits: they can be managed in main memory, which decreases query processing time, and they can be transferred quickly within a storage hierarchy or via a network. Since synopses only represent a subset of the underlying data, query results often come with errors accordingly. However, it is possible to define exact bounds for these errors. All these features make synopses ideally suited for data analysis tasks in a DSMS. With constant synopsis size (footprint), it is thus possible to monitor data over a longer time period with little accuracy or to monitor them during short time intervals with high accuracy. In previous work [15, 16], the design of synopses was restricted to the preservation of certain memory sizes and error bounds. In our DSMS to be developed, we will consider the energy consumption as another optimization criterion when designing synopses. In this context, the energy consumption has to be considered from two perspectives: for storing and maintaining the synopses as well as for the query processing based on synopses. Furthermore, it will be decisive how many synopses have to be created and maintained for a certain workload. From the energy consumption point of view, we need to decide whether to create a few general or many specialized synopses.

V. ANYTIME ALGORITHMS

The synopses serve to buffer the stream of data and to represent it in a compact but lossy form. This allows speeding up the algorithms based on this data and to make their processing more energy-efficient. Another option to support energy efficiency is to leave out some iterations or computations within the data stream operators itself. This is possible with so-called anytime-algorithms, which trade computational resources for solution quality, i.e. the quality of the result improves with execution time. This requires a well-defined quality measure which allows to monitor the progress in problem solving and to allocate computational resources accordingly. Given this measure, it is possible to suspend the processing after a certain time period and to peek results which may not be in their final state (see Figure 1). Further it must be possible to resume the algorithm with minimal overhead.

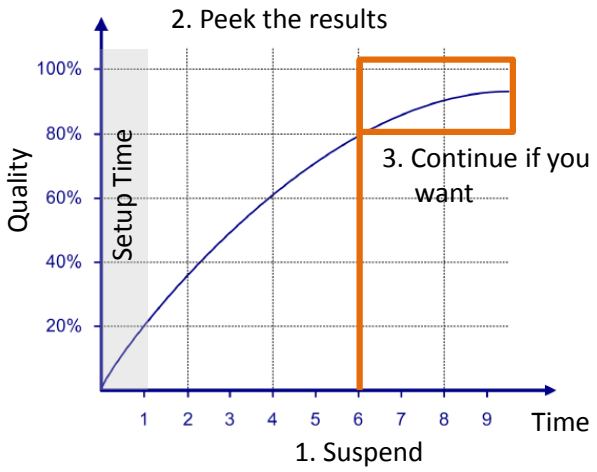

Figure 1. Anytime Processing Model

This characteristic makes them a perfect fit for an energy-efficient processing following the "need to know" principle. In particular, we will focus on anytime-data-mining algorithms, some of which have been documented in [17, 18], and on anytime-forecast algorithms. Obviously, the research work in both fields is very sparse, which implies that many algorithms need to be designed from scratch. For this purpose, we can rely on existing classifications and programming tools [19, 20] that provide an environment to support anytime-computation. This includes tools for the automatic construction of performance profiles (to quantify the performance improvement over time) as well as tools for composition, activation, and monitoring.

VI. RESEARCH GOALS

The broad range of software sensor data leads to many data streams that are pre-filtered through a set of sampling steps. The synopses are found on top of this sampling layer; they buffer and pre-aggregate data for detailed analyses. When doing so, a synopsis is always fed by one or multiple

 ISBN: 978-1-61208-136-6

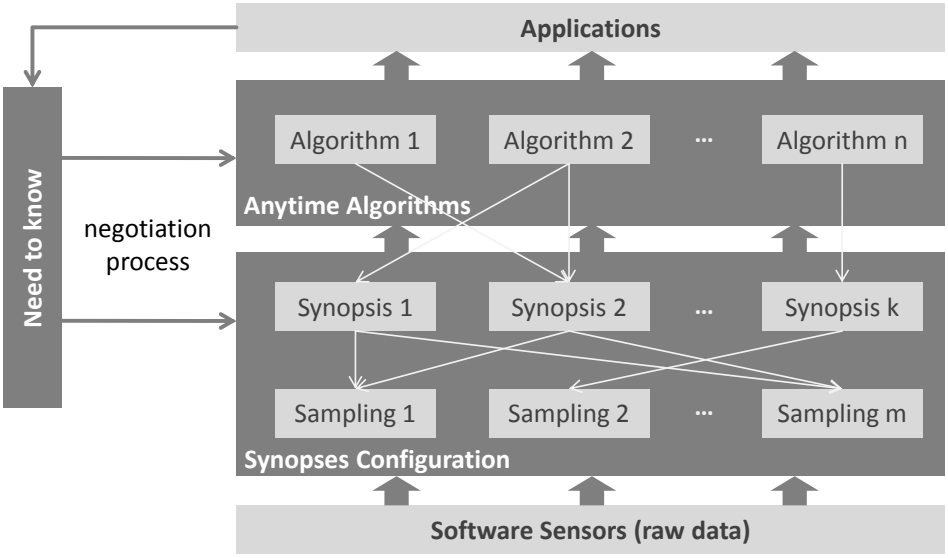

Figure 2. Data monitoring and processing following the "need-to-know"principle

sampling operators. In analogy to this, anytime-operators get their data from one or multiple synopses. All three layers are configured and controlled by the same "need-to-know" principles. In the following, we will present these ideas in detail.

A. *Formalization of the "need to know" principle as an enabler for energy-awareness*

The goal of this work package is to formalize the above mentioned "need-to-know" principle in order to specify the data quality constraints of the application layer. These constraints provide valuable help to optimize 1) the sampling preprocessing step, 2) the number, size and structure of the synopses, and the 3) algorithmic computation based on these synopses. Typical quality measures in a DSMS can be classified into time- and content-based metrics. The former indicates the DSMS's ability to adapt to the push-based data delivery and to the processing speed of the DSMS, e.g., throughput (or data rate) and the latency of the result. The latter focuses on problems caused by the handling of infinite streams or high input volumes. Content-based metrics that can be applied in this context are sampling rates, quality guarantees or error bounds. All these measures need to be reviewed and integrated into a common constraint model. Therefore, we need to check the interdependencies between these constraints and provide a declarative language that allows to express the specific requirements of the data consumers.

B. *Need-to-know" negotiation process*

In order to guarantee the "need-to-know" constraints a negotiation process is required that mediates between the application, the operation, and the synopsis layer (see Figure 2). In the first step, the capability for fulfilling the constraints need to be calculated locally on each layer and globally for the whole system. Therefore, we need to rely on data stream statistics that have to be collected during the runtime of the system. In a second step, the required resources (processing time at the operational level, space and I/O for the synopses, bandwidth for the sampling step, etc.) have to be reserved. If the statistics of the data stream do not change in any unforeseen manner, the required "need-to-know" constraints can be guaranteed. Otherwise, the negotiation process needs to be repeated.

C. *Data Stream Sampling*

The continuous data flow which need to be monitored calls for data stream sampling schemes as a pre-processing step (see Figure 2) to reduce or bound resource consumption. Although many techniques for maintaining database samples can also be used for data streams, there are problems that are unique to data stream sampling. The main difference in data stream sampling is that samples are often biased towards more recent items because recent items are considered more important by applications. Such a notion of recency does not exist in traditional database sampling and calls for novel sampling schemes. Additionally, since a data stream is changing continuously, efficient sample maintenance strategies must be developed as well.

D. *Synopsis Design and Configuration*

Synopses are necessary to materialize data stream content, which allows to apply a wide range of analytical operations such as outlier and trend detection, clustering, forecasts, and so on. Given a specific memory bound (footprint) and a set of "need-to-know" constraints, we want to derive the optimal synopsis configuration regarding energy-consumption. Since there is a many-to-many relationship between operations and synopses, we have to decide between general-purpose synopses, which may serve

 ISBN: 978-1-61208-136-6

a wide range of applications, and synopses designed for a very specific purpose. The design of the synopses must be coordinated with the previous sampling step, i.e. it has to be decided which data items will be pruned at the first level and which data items are considered to derive groups and aggregates at the second level. Another problem arises from the synopsis computation effort, which increases exponentially with the input parameters such as the number of relations or the number of grouping attributes. In order to keep the system load (and thus the energy consumption) for the computation and maintenance of synopses as low as possible, we will thus have to develop appropriate heuristics and greedy approaches.

E. Anytime-Algorithms

Whereas synopses restricted the storage consumption of the data, we also need new algorithmic approaches that make it possible to abort computations as early as possible. For this purpose, we want to look at the discipline of anytime-algorithms and transfer our findings to the "need-to-know" concept. That means that the underlying algorithms are not executed until they converge, but they are stopped as soon as the intermediate result fulfills the "need-to-know" constraints. These kind of algorithms, which continuously generate intermediate results, need to be developed for a broad range of application areas. Therefore, we have to perform research on algorithms whose runtime is not determined in advance but can be interrupted at any time during execution and return a result. Further, we have to work on performance profiles, which returns result quality (defined by the "need-to-know" negotiation process) as a function of time. As a starting point, we focus on clustering operations and, more precisely, on the k-means algorithm, which is widely used and well understood in the research community. In particular, there already exists a wavelet-based anytime-algorithm, which should be reviewed for its applicability [17]. This will give us first insights on how the "need-to-know" principle may interplay with the idea of anytime-algorithms.

Furthermore, we have to find the optimal configuration of the sampling steps (load shedding), the synopses (pre-aggregation), and the variety of analysis algorithms and operators.

VII. CONCLUSION

In this paper, we presented the concept of an energy-aware data stream management system. Therefore, we introduced the overall “need-to-know” principle which should be applied to each layer of the DSMS architecture. At the data layer we will use synopses to minimize the quantity and granularity of data. At the process level we will apply anytime operators which will allow us to prematurely abort the execution of operations as soon as a defined quality has been reached. Both concepts give us additional flexibility to build an adaptive and most of all an energy-aware DSMS.

REFERENCES

[1] Dimitris Tsirogiannis, Stavros Harizopoulos, and Mehul A. Shah. “Analyzing the energy efficiency of a database server,” In SIGMOD’10, pages 231–242, New York, NY, USA, 2010. ACM.

[2] Justin Meza, Mehul A. Shah, Parthasarathy Ranganathan, Mike Fitzner, and Judson Veazey. “Tracking the power in an enterprise decision support system,” In ISLPED’09, pages 261–266, New York, NY, USA, 2009. ACM.

[3] Willis Lang and Jignesh M. Patel. “Towards eco-friendly database management systems,” In CIDR, 2009.

[4] Zichen Xu. “Building a power-aware database management system,” In IDAR’10, pages 1–6, New York, NY, USA, 2010. ACM.

[5] Luiz André Barroso and Urs Hölzle. “The case for energy-proportional computing. Computer,” 40(12):33–37, 2007.

[6] Theo Härder, Karsten Schmidt, Yi Ou, and Sebastian Bächle. ”Towards Flash disk use in databases - keeping performance while saving energy?,” In BTW’09, pages 167–186, 2009.

[7] Nesime Tatbul, Ugur Çetintemel, Stan Zdonik, Mitch Cherniack, and Michael Stonebraker. “Load shedding in a data stream manager,” In VLDB’03, pages 309–320. VLDB Endowment, 2003.

[8] Brian Babcock, Mayur Datar, and Rajeev Motwani. “Load shedding for aggregation queries over data streams,” In ICDE’04, page 350, Washington, DC, USA, 2004. IEEE Computer Society.

[9] Jürgen Krämer and Bernhard Seeger. ”Pipes: a public infrastructure for processing and exploring streams,” In SIGMOD’04, pages 925–926, New York, NY, USA, 2004. ACM.

[10] Chuck Cranor, Theodore Johnson, Oliver Spataschek, and Vladislav Shkapenyuk. “Gigascope: a stream database for network applications,” In SIGMOD’03, pages 647–651, New York, NY, USA, 2003. ACM.

[11] Sirish Chandrasekaran, Owen Cooper, Amol Deshpande, Michael J. Franklin, Joseph M. Hellerstein, Wei Hong, Sailesh Krishnamurthy, Samuel R. Madden, Fred Reiss, and Mehul A. Shah. “Telegraphcq: continuous dataflow processing,” In SIGMOD’03, pages 668–668, New York, NY, USA, 2003. ACM.

[12] Jianjun Chen, David J. DeWitt, Feng Tian, and Yuan Wang. “Niagaracq: a scalable continuous query system for internet databases,” In SIGMOD’00, pages 379–390, New York, NY, USA, 2000. ACM.

[13] Sven Schmidt, Thomas Legler, Sebastian Schär, and Wolfgang Lehner. ”Robust real-time query processing with QStream,” In VLDB ’05, pages 1299–1301. VLDB Endowment, 2005.

[14] Phillip B. Gibbons and Yossi Matias. “Synopsis data structures for massive data sets,” In SODA ’99, pages 909–910, Philadelphia, PA, USA, 1999. Society for Industrial and Applied Mathematics.

[15] Rainer Gemulla, Wolfgang Lehner, and Peter J. Haas. ”A Dip in the Reservoir: Maintaining Sample Synopses of Evolving Datasets,” In VLDB’06, pages 595–606, 2006.

[16] Rainer Gemulla and Wolfgang Lehner. ”Sampling time-based sliding windows in bounded space,” In SIGMOD ’08, pages 379–392, New York, NY, USA, 2008. ACM.

[17] Michail Vlachos, Jessica Lin, Eamonn Keogh, and Dimitrios Gunopulos. “A wavelet-based anytime algorithm for k-means clustering of time series,” In In Proc. Workshop on Clustering High Dimensionality Data and Its Applications, pages 23–30, 2003.

[18] Philipp Kranen, Ira Assent, Corinna Baldauf, and Thomas Seidl. “Self-adaptive anytime stream clustering,” In Wei Wang, Hillol Kargupta, Sanjay Ranka, Philip S. Yu, and Xindong Wu, editors, ICDM’09, pages 249–258. IEEE Computer Society, 2009.

[19] Bei Hui, Ying Yang, and GeoUrey I. Webb. “Anytime classification for a pool of instances,” Mach. Learn., 77(1):61–102, 2009.

[20] Joshua Grass and Shlomo Zilberstein. “Anytime algorithm development tools,” SIGART Bull., 7(2):20–27, 1996.

 ISBN: 978-1-61208-136-6