Наука и Образование
МГТУ им. Н.Э. Баумана
Сетевое научное издание
ISSN 1994-0448

Наука и Образование. МГТУ им. Н.Э. Баумана. Электрон. журн. 2014. № 10. С. 284–291.

DOI: **10.7463/1014.0728338**

Представлена в редакцию: 30.09.2014

УДК 004.627:004.65

Методы легковесного сжатия для хранения временных рядов в базах данных и их реализация на графических процессорах

Белоус В. В.[1,*], **Пивоварова Н. В.**[1]

*walentina.belous@gmail.com

[1]МГТУ им. Н.Э. Баумана, Москва, Россия

Ключевые слова: сжатие информации, база данных, временной ряд

Основными задачами настоящей работы являются исследование и разработка легковесных методов сжатия информации, подходящих для обработки временных рядов, и их реализация на графических процессорах.

В статье рассматриваются три метода легковесного сжатия: разностное кодирование, кодирование RLE и словарное кодирование. Для методов разностного кодирования и словарного кодирования предлагаются модификации, расширяющие сферу применимости данных методов.

Предлагается адаптивный метод, основанный на каскадировании вышеуказанных алгоритмов. Он заключается в том, что после предварительной обработки данных, выполняется преобразование, представляющее собой последовательное применение одного, двух или трех из вышеприведенных методов. При этом, с некоторым периодом, производится попытка сжатия очередного пакет данных различными каскадами, из них определяется наилучший, который и используется до конца периода.

Предложенные алгоритмы были реализованы на графическом процессоре AMD FirePro S10000 с помощью интерфейса OpenCL. В статье приведены полученные показатели производительности и сравниваются коэффициенты сжатия. Для адаптивного метода производительность составила около 200 Гбит/с.

Таким образом в статье рассмотрен ряд легковесных методов сжатия временных рядов и их реализация на графических процессорах. Предложены модификации стандартных методов, а также метод адаптивного легковесного сжатия информации, ctctкоторый показал весьма перспективные результаты.

Список литературы

1. Белоус В.В. Электронные библиотеки // Инженерный вестник. МГТУ им. Н.Э. Баумана. Электрон. журн. 2012. № 11. Режим доступа: http://engbul.bmstu.ru/doc/500497.html (дата обращения 01.09.2014).
2. Белоус В.В., Домников А.С. Интеллектуальный анализ данных в электронных обучающих системах // Инженерный вестник. МГТУ им. Н.Э. Баумана. Электрон. журн. 2013. № 12. Режим доступа: http://engbul.bmstu.ru/doc/656610.html (дата обращения 01.09.2014).
3. Белоус В.В., Смирнова Е.В. Электронное обучение. Платформы и системы // Инженерный вестник. МГТУ им. Н.Э. Баумана. Электрон. журн. 2013. № 7. Режим доступа: http://engbul.bmstu.ru/doc/654234.html (дата обращения 01.09.2014).
4. Ключарев П.Г. Блочные шифры, основанные на обобщённых клеточных автоматах // Наука и образование. МГТУ им. Н.Э. Баумана. Электрон. журн. 2012. № 12. С. 361-374. DOI: 10.7463/0113.0517543
5. Ключарев П.Г. Криптографические хэш-функции, основанные на обобщённых клеточных автоматах // Наука и образование. МГТУ им. Н.Э. Баумана. Электрон. журн. 2013. № 1. С. 161-172. DOI: 10.7463/0113.0534640
6. Сэломон Д. Сжатие данных, изображений и звука: пер. с англ. М.: Техносфера, 2006. 365 с. [Salomon D. Data Compression. The Complete Reference. 3rd ed. Springer New York, 2004. DOI: 10.1007/b97635].
7. Эльшафеи М.А., Сидякин И.М. Применение метода адаптивного линейного предсказания для сжатия телеметрической информации // Наука и образование. МГТУ им. Н.Э. Баумана. Электрон. журн. 2014. № 4. С. 354-366. DOI: 10.7463/0414.0707364
8. Chang F., Dean J., Ghemawat S., Hsieh W.C., Wallach D.A., Burrows M., Chandra T., Fikes A., Gruber R.E. Bigtable: A distributed storage system for structured data // ACM Transactions on Computer Systems (TOCS). 2008. T. 26, no. 2. Art. no. 4. DOI: 10.1145/1365815.1365816
9. George L. HBase: the definitive guide. O'Reilly, 2011. 554 p.
10. Wlodarczyk T.W. Overview of Time Series Storage and Processing in a Cloud Environment // 2012 IEEE 4th International Conference on Cloud Computing Technology and Science (CloudCom). IEEE, 2012. P. 625-628. DOI: 10.1109/CloudCom.2012.6427510
11. Zukowski M., Heman S., Nes N., Boncz P. Super-Scalar RAM-CPU Cache Compression // Proceedings of the 22nd International Conference on Data Engineering (ICDE '06). IEEE, 2006. P. 59. DOI: 10.1109/ICDE.2006.150

.

Science & Education of the Bauman MSTU
Electronic journal
ISSN 1994-0448

Science and Education of the Bauman MSTU, 2014, no. 10, pp. 284–291.

DOI: **10.7463/1014.0728338**

Received: 30.09.2014

Lightweight Compression Methods for Storing Time Series in Databases and Their GPU Implementation

V.V. Belous[1,*], N. V. Pivovarova[1]

*walentina.belous@gmail.com

[1]Bauman Moscow State Technical University, Moscow, Russia

Different lightweight methods for time series compression are discussed in the work. An approach to managing this compression is proposed, in the context of which a cascade of compression algorithms is considered; moreover these algorithms are chosen dynamically based on features of the data to be compressed. Implementation of these algorithms on graphics processing units (using OpenCL) allows a speed of operation on the order of 200 Gbit/s to be achieved. The methods proposed in the paper may find wide application in tasks of on-line storage and processing of telemetry data of different complex objects and systems.

Keywords: data compression, database, time series

1. Introduction

Now telemetry systems are applied everywhere. A data storage subsystem is the most important part of these systems. This subsystem must provide the on-line record and storage of digitized signals in the database; moreover it is often required to process millions of samples per second. Under conditions of regular work, several terabytes per day may be added to the database. A number of solutions is used for these tasks (e.g., TempoDB [10] can be noted). For the most part, they are based on the Big Table approach proposed by the Google company [8].

Telemetry data mainly are time series. While recording these data with their large size taken into account, it makes sense to use this or that algorithm of data compression. Some dedicated DBMSs, like HBase [9], use this approach; however the compression algorithms embodied in them are insufficiently optimized for time series.

Many widely used compression algorithms, such as LZW, LZ77, LZBA, arithmetic coding, etc. [6], as well as the algorithms based on the adaptive linear prediction [7], work insufficiently rapidly, which does not allow them to be used for processing of large data amounts with limited computing resources. Therefore it is expedient to apply the faster compression methods capable of real time operation but with a lesser value of compression ratio. These compression methods must utilize the time series features. By analogy with fast cryptographic algorithms (like, e.g., those proposed in [4, 5]), we call these compression algorithms the lightweight ones.

In many telemetry systems, it is necessary to provide compression of many data streams simultaneously. This process can be substantially accelerated by implementation of the algorithms at heterogeneous computing systems such as graphics processing units.

Thus, main goals of this work are investigation and development of lightweight data compression methods, suitable for processing of time series, and their implementation on graphics processing units.

2. Lightweight Compression Methods

We consider some classical methods of lightweight compression by applying to them certain modernizations as development of the approach proposed in [11].

Preprocessing. Before applying directly data compression, it makes sense to carry out a preliminary preparation of the data. In particular, in order to escape the unnecessary data storage, the data valuation with rounding should be performed. It can be often useful to store data in the format with a fixed decimal point. After data preprocessing, their compression should be produced.

Delta encoding (Differential compression). One of the important compression methods is the delta encoding. The method essence is that instead of values of each sample, differences between neighboring samples are stored in the database. This allows the samples to be encoded with fewer amounts of bits. The problem here is that if magnitudes in time series change rapidly, a difference may take sufficiently large values. Let us consider, e.g., the time series segment

123, 126, 128, 130, 133, 134, 134, 135, 208, 211, 213, 214, 216, 217, 219, 219, 220.

The following sequence of differences corresponds to it:

3, 2, 2, 3, 1, 0, 1, 73, 3, 2, 1, 2, 1, 2, 0, 1.

Here each difference with a single exception can be coded with two bits. In this context, the techniques of coding with a variable amount of bits are typically used (e.g., a Huffman code, Shannon-Fano coding, arithmetic coding, etc.); however, these techniques work relatively slowly. As a lightweight approach, we offer to divide information into packets each of which consists of two parts:

1. a sequence of differences (of fixed length);
2. a sequence of large values.

In this case, the first part of the packet stores the differences which can be located in k bits (for the certain k, empirically chosen for a given time series), while instead of differences which cannot be placed in k bits, zeros are stored. In the second part of the packet, the differences with a length more than k bits are stored in the <offset><difference> format, attached with a prenex introducing their amount (for which the fixed length field $r \le \lceil \log_2 n \rceil$ is allocated).

The number k for each time series must be empirically chosen such that it would approximately minimize the packet length L, defined as

$$L = kn + \left(\lceil \log_2 n \rceil + m \right) \cdot g(k) + r,$$

where m is the maximal length of a difference, n is the amount of differences in the packet, $g(k)$ is the number of the differences d, such that $d \geq 2^k$.

Run-length encoding (RLE). Another important and well-known method is RLE, making the data to be encoded in pairs (a number of element repetitions, element itself). For example, the series [2,2,2,3,3,3,3,3,7,7,7,7,7,7,7,7] is encoded as [(3,2), (5,3), (9,7)].

This method is applicable to the series comprising a large amount of repeated values.

Dictionary encoding. If terms of a series can take on values only from a small set of values, then it is expedient, by having logged all possible values in a dictionary, to remember only indices. This method is applicable to the data that can take only a small amount of values. In the case if the data can take many values, however a probability of taking the larger part of values is small, we can offer to do the work similarly to what we did in the case of delta encoding, namely to form data packets consisting of two parts:

1. a sequence of dictionary indices;
2. an additional dictionary for the given packet.

In this case, a basic dictionary must be available, in which the most probable values are stored, and a dictionary for each packet where the values are stored that are present in the packet but not available in the basic dictionary. A size of the basic dictionary must be selected on the basis of a specific character of the individual problem.

Adaptive method. The adaptive method is based on cascading of algorithms. It implies a cascade of conversions (performed sequentially over the data after their preprocessing), which represents a consecutive application of one, two or three compression methods mentioned above (dictionary encoding, delta encoding, and RLE). In this case, an attempt to compress a next (in turn) packet of data is made with a certain period by different cascades, the best of them is determined, which is used up to the end of this period. Taking into account that it is possible to choose a single algorithm in three ways, two algorithms in six ways (an order matters), and three algorithms also in six ways (the number of permutations is 3!) and, additionally, it is permissible not to carry out compression at all (while to perform only data preprocessing), we obtain a total number of different cascades to be equal to 16. This number is sufficiently small in order that all these cascades could be tested.

3. Implementation

The offered algorithms were implemented on the AMD FirePro S10000. This graphics processing unit is intended specially for high-performance heterogeneous computations and has a peak throughput up to 5.9 TFLOPS.

The implementation of computational kernel was accomplished using the OpenCL interface. Here Python 2.7.3 was used as programming language for the host program while the PyOpenCL 2013.1 library was applied for operation with OpenCL.

Testing was conducted with various telemetry data on functioning of ten different servers providing the operation of miscellaneous electronic libraries and distance learning systems including those described in [1–3].

As a result the throughput shown in Fig. 1 was obtained. For the test data (as such were used various counters of throughput of ten different servers running under control of different versions of the Linux operating system), the comparison of data compression ratios has been performed, which is displayed in Fig. 2.

It is interesting that the adaptive method allows the larger compression ratio to be achieved, as compared with any other method, but loses little in throughput at that, which is caused by the fact that the major part of time is spent on passing data from the main memory to the memory of graphics processing unit.

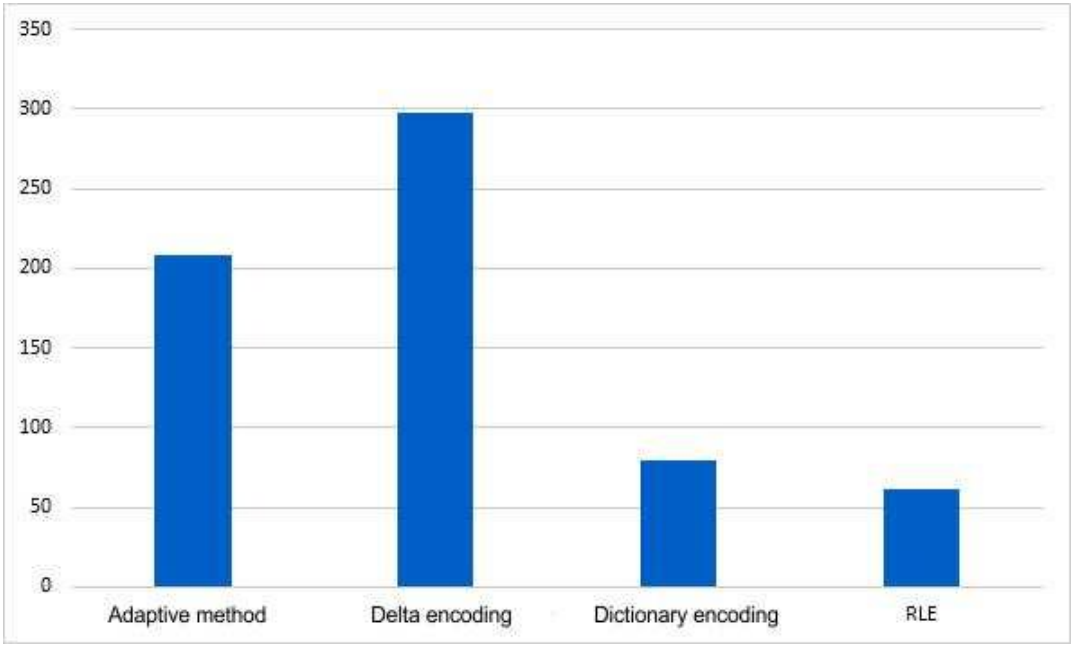

Fig. 1 – Throughput of different methods, Gbit/s

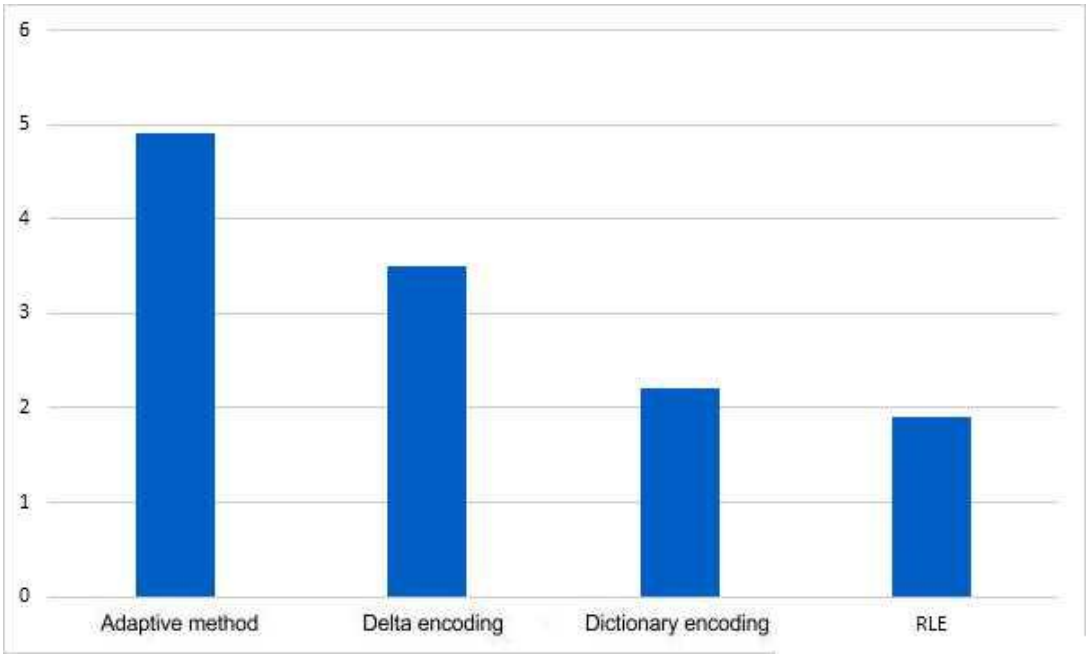

Fig. 2 – Compression ratio

4. Conclusions

Thus, a number of lightweight methods for time series compression and their implementation on graphics processing units are considered in the paper. Modifications of standard methods, as well as the method for adaptive lightweight data compression (which has demonstrated very promising results), are proposed.

Further work may be oriented on the investigation of other lightweight compression techniques including the lossy compression algorithms. Additionally it is planned to work over issues of implementation of lightweight methods for time series compression on programmable logic integrated circuits.

References

1. Belous V.V. Digital libraries. *Inzhenernyi vestnik MGTU im. N.E. Baumana = Engineering Herald of the Bauman MSTU*, 2012, no. 11. Available at: http://engbul.bmstu.ru/doc/500497.html , accessed 01.09.2014. (in Russian).
2. Belous V.V., Domnikov A.S. Data mining in e-learning systems. *Inzhenernyi vestnik MGTU im. N.E. Baumana = Engineering Herald of the Bauman MSTU*, 2013, no. 12. Available at: http://engbul.bmstu.ru/doc/656610.html , accessed 01.09.2014. (in Russian).
3. Belous V.V., Smirnova E.V. E-learning. Platforms and Systems. *Inzhenernyi vestnik MGTU im. N.E. Baumana = Engineering Herald of the Bauman MSTU*, 2013, no. 7. Available at: http://engbul.bmstu.ru/doc/654234.html , accessed 01.09.2014. (in Russian).
4. Klyucharev P.G. Construction of pseudo-random functions based on generalized cellular automata. *Nauka i obrazovanie MGTU im. N.E. Baumana = Science and Education of the Bauman MSTU*, 2012, no. 12, pp. 361-374. DOI: 10.7463/0113.0517543 (in Russian).
5. Klyucharev P.G. Cryptographic hash functions based on generalized cellular automata. *Nauka i obrazovanie MGTU im. N.E. Baumana = Science and Education of the Bauman MSTU*, 2013, no. 1, pp. 161-172. DOI: 10.7463/0113.0534640 (in Russian).
6. Salomon D. *Data Compression. The Complete Reference*. 3rd ed. Springer New York, 2004. DOI: 10.1007/b97635 (Russ. ed.: Salomon D. *Szhatie dannykh, izobrazheniy i zvuka*. Moscow, Tekhnosfera Publ., 2006. 365 p.).
7. El'shafei M.A., Sidyakin I.M. Lossless Compression of Telemetry Information using Adaptive Linear Prediction. *Nauka i obrazovanie MGTU im. N.E. Baumana = Science and Education of the Bauman MSTU*, 2014, no. 4, pp. 354-366. DOI: 10.7463/0414.0707364

8. Chang F., Dean J., Ghemawat S., Hsieh W.C., Wallach D.A., Burrows M., Chandra T., Fikes A., Gruber R.E. Bigtable: A distributed storage system for structured data. *ACM Transactions on Computer Systems (TOCS)*, 2008, vol. 26, no. 2, art. no. 4. DOI: 10.1145/1365815.1365816
9. George L. *HBase: the Definitive Guide*. O'Reilly, 2011. 554 p.
10. Wlodarczyk T.W. Overview of Time Series Storage and Processing in a Cloud Environment. *2012 IEEE 4th International Conference on Cloud Computing Technology and Science (CloudCom)*. IEEE, 2012, pp. 625-628. DOI: 10.1109/CloudCom.2012.6427510
11. Zukowski M., Heman S., Nes N., Boncz P. Super-Scalar RAM-CPU Cache Compression. *Proceedings of the 22nd International Conference on Data Engineering (ICDE '06).* IEEE, 2006, p. 59. DOI: 10.1109/ICDE.2006.150