Cache-aware algorithm for multidimensional correlations

E A Altman, T V Vaseeva, A V Aleksandrov
Omsk State Transport University, Omsk, Russia

E-mail: altmanea@gmail.com, tvvaseeva@gmail.com, aleksandrovav@omgups.ru

Abstract. In the paper discusses a speed of a convolution or correlation functions in case of a big size of input data. When a data volume exceeds a cache size, a significant part of an execution time is spent on a loading data into the cache. A performance of correlation is explored an example of receiving a long radio signal by the use of comparison with a big number of patterns. To decrease this spending a method of reordering calculation and memory operations and is proposed. The method uses cache aware approach to construct cache friendly algorithms. A decreasing of execution time (in some case) achieves up to 44%

Keywords: convolution, correlation, cache memory, algorithm, pattern.

1. Introduction
A convolution is an important mathematical formula, which base correlation, filter and other operations of digital signal processing. An efficient computation of the convolution is a popular subject of a science research and is interested a software and hardware developers.

A fast implementation of the convolution is required an applying of a mathematical algorithm with a low quantity of an arithmetic operations and an optimization of a program code with an accounting of a hardware. In the paper, we pay attention to the second requirement.

Two main directions of the hardware optimization are an effective using of a CPU executive units and a utilization of a memory bandwidth. The first direction assumes an increasing of a load of the CPU units and using commands that execute several arithmetic operations (primarily, SIMD instructions). The second direction helps to avoid an idle of CPU by providing a data on time. Its main idea consists in a usage of a calculation order that minimizes a data transfer between the memory and the CPU and between the CPU and its cache.

In the paper, we present a method of a decreasing of a correlation calculation time by an improving of a pattern of a CPU cache usage. This method can be useful for multidimensional correlations and convolutions that have an input data size exceeding the CPU cache size. An idea of the method consists in to find a calculation scheme that minimizes a transfer data between the cache and memory. The method is so called "cache-aware", because it needs information about the cache size. A "cache-oblivious" variant of the method is possible, but it requires some additional memory operations and less effective.

2. Related work
The patterns of a CPU cache usage are special cases for patterns of an access to a hierarchical memory. For an analysis of the access to a hierarchical memory is usually used a Hierarchical Memory Model (HMM), which was introduced in a publication [1]. In that publication determined...
levels of a computer system memory hierarchy: fast memory (registers of the CPU), several levels of cache, main memory, extended and external storages. This determination is completely actual nowadays. In [1] reconsidered matrix multiplication, FFT and sorting algorithms for HMM and gave an estimation of theirs time complexity bounds.

Early researches were concentrated on a problem of main memory small size and external storage latency [2]. A strong motivation of those researches is a high degradation of calculation performance when a data volume exceeds the main memory size. A review of external memory algorithms can be found in [3] and later works of the same author.

A significant increasing of CPU clock frequency relative the main memory revealed problems of main memory latency and a cache size limitation. A mainstream solution of those problems is cache friendly block algorithms. The block algorithms are most effective for case of matrix operations [4]. They change a calculation order from processing a whole row or columns to manipulating blocks of matrixes that fit entirely in cache. Methods for a finding of the cache friendly calculation order for other algorithm types are a little more complicated and a less universal. For example, cache friendly sorting was researched in a publication [5].

The next step in the researches of the cache friendly algorithms is an invention of cache-oblivious algorithms [6] (latest review – [7]). A first cache-oblivious algorithm was proposed for fast Fourier transform. An idea concludes in a recursive union the small size blocks of the processing data in bigger size ones. At some level of recursion the block size fits the cache size and one cache load is required for its calculation. This idea has been successfully applied to various algorithms and data structures [8-10].

A common modern review of the algorithms of the convolution can be found in a publication [11]. Despite of the long studies there are opportunities for improving its algorithmic complexity [12]. In the practical terms hardware optimization, including through the use the cache friendly algorithm, is more important than the algorithmic complexity of the convolution [13]. Current research of this issue primarily focuses on multiple processor architectures [14, 15].

3. Practical goal

There are enough software libraries, allowing to calculate the various type correlation and convolution, including free and open source ones. In most cases they work very well. However, exploring a performance of big multidimensional correlation, we revealed an extra degradation of an execution time when convolution size exceeds some level. Situation analysis showed that, the extra degradation arises from insufficient the cache size.

We were using the big multidimensional correlation to receive a radio signal that had a power much less a noise power. This task is often applied in secret and secure transmission systems [16]. To detect the signal, its duration was about a few second and the sampled signal had thousands of samples. To receive the signal, a computer had to compare it with hundreds of patterns those differ a frequency of a signal spectrum, a start time and a transmitted symbol.

A measured a correlation execution time is presented in figure 1. The measurements were made on Intel i7 processor, with a frequency 2 GHz and 6 GB RAM. The processor has L1 cache 2*32 KB for data and instruction, L2 cache 2*256 KB (for two cores) and 4MB L3 cache. The correlation was calculated by NumPy library of Python language that was provided by an Anaconda package manager.

In figure 1 on x axis is a number of the samples in the patterns, on y axis is an average execution time in milliseconds. Different lines in the figure correspond to different lengths of the received signal. The time values are the average time of correlation calculation for 100 experiments.
Figure 1. A measured a convolution execution time.

All lines in the figure have surges at a point about 2000 samples. An analysis of the situation suggests that this point matches a pattern length at which the data size of all patterns no more than the L3 cache size.

Presume that the surges of the execution time can be deleted by changing the calculation order.

4. Results

A key idea of cache friendly correlation is to split the data into several parts which data sizes fit cache size. The development of such an algorithm for multidimensional correlation requires answers to two questions: what dimension to use for splitting and how many parts to divide it.

Splitting data into large parts increases a quantity of intermediate data, but makes the algorithm friendly to a small size cache. A selecting the dimensionality to split also affects the amount of the intermediate data, the amount of which proportional to the size of non-splitting dimensionalities.

In our case, the cached data can be pictured as it presented on figure 2, where “symbol” is number of transmitted symbol, “harmonic” is number of using frequency and “sample” is number of samples in pattern. In cache friendly calculation scheme the sample dimension is splitted into $n$ parts, where $n$ estimated by:

$$ n = \frac{N_{\text{Sym}}N_{\text{Sam}}N_{\text{harm}}V_1}{V_c} $$

where $N_{\text{Sym}}$ is the number of symbols, $N_{\text{Sam}}$ is number of samples, $N_{\text{harm}}$ is number of harmonics, $V_1$ is size of one variable of an using data type and $V_c$ is volume of cache.
A studies of a various ways of splitting correlation reveal that the best performance is obtained by splitting a samples dimensionality on parts, that size no more than 2000 samples. The final version of the correlation function that provides the best performance by changing the calculation order is present in figure 3.

**Figure 2.** A cached data.
A measured cache aware correlation execution time is presented in figure 4.
Figure 4. A measured a cache aware correlation execution time.

As seen in figure 3, the cache aware correlation does not have the surges of the execution time. The little larger execution time at the beginning of the lines compared to the direct correlation is due mainly to a non-optimized program code in the high-level language. The increasing of speed at the pattern size 2000 samples in comparison with the direct correlation was up to 44%.

Similar results were obtained when testing these functions on other computers. Involved processors have the various volumes of L3 cache, consequently there were surges of the direct correlation performance at various pattern sizes.

5. Discuss

In the paper was proposed and analyzed the method to the speed improving of the correlation function on the desktop processor. This method simplifies the modern cache-obvious methods and can be applied to wide range algorithms, which work with big volumes of data.

The implementation of this method successfully solves a problem that described in "practical goal" section. It can be applied to find the big size convolution and correlation. The performance improvement up to 44% is achieved using a high-level language and can be increased through a software optimization.

L2 cache size of modern widely used processors lies in narrow range from 2 to 8 MBytes. A cache obvious approach in this case is not needed because we can use 2MBytes as maximum size of subcorrelation for all modern processors without a meaningful performance lost.

A promising area of further research is a bringing together the cache aware method and well-known method "divide and conquer" for decreasing of arithmetic complexity.

References
[1] Aggarwal A, Alpern B, Chandra A, and Snir M 1987 A model for hierarchical memory *19th annual ACM symposium on Theory of computing* (New York: ACM) pp 305-314
[2] Goodrich M T, Tsay J J, Vengroff D E and Vitter J S 1993 External-memory computational geometry. *Foundations of Computer Science, 34th Annual Symposium* (Palo Alto: IEEE) pp 714-723
[3] Vitter J S 1998 External memory algorithms The European Symposium on Algorithms (Berlin, Heidelberg: Springer) pp 1-25
[4] Lam M D, Rothberg E E and Wolf M E 1991 The cache performance and optimizations of blocked algorithms J. ACM SIGARCH Computer Architecture News 19 pp 63-74
[5] LaMarca A and Ladner R E 1999 The influence of caches on the performance of sorting Journal of Algorithms 31 pp 66-104
[6] Frigo M, Leiserson C E, Prokop H and Ramachandran S 1999 Cache-oblivious algorithms. Foundations of Computer Science 40th Annual Symposium (New York) p 8
[7] Frigo M, Leiserson C E, Prokop H and Ramachandran S 2012 Cache-oblivious algorithms. ACM Transactions on Algorithms (ACM) p 4
[8] Demaine E D 2002 Cache-oblivious algorithms and data structures EEF Summer School on Massive Data Sets p 1-249
[9] Kumar P 2003 Cache oblivious algorithms. Algorithms for Memory Hierarchies (Springer, Berlin, Heidelberg) pp 193-212
[10] Günther F, Mehl M, Pögl Mand Zenger C 2006 A cache-aware algorithm for PDEs on hierarchical data structures based on space-filling curves SIAM Journal on Scientific Computing 28 pp 1634-50
[11] Karas P and Svoboda D 2013 Algorithms for efficient computation of convolution Design and Architectures for Digital Signal Processing (Electronic Materials) pp 179-208
[12] Zakharenko E I and Altman E A 2018 A new algorithm of fast SSD calculation for motion estimation J. Phys.: Conf. Ser. 1050 012002
[13] Battenberg E and Avizienis R. 2011 Implementing real-time partitioned convolution algorithms on conventional operating systems 14th International Conference on Digital Audio Effects (Paris) pp 313-320
[14] Jillings N, Reiss J D and Stables R 2017 Zero-Delay large signal convolution using multiple processor architectures Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA, IEEE) pp 339-343
[15] Xu M, Zhu M, Liu Y, Lin F X and Liu X 2018 DeepCache: Principled Cache for Mobile Deep Vision 24th Annual International Conference on Mobile Computing and Networking (ACM) pp 129-144
[16] Altman EA, Malutin AG, Romanov Yu V, Fomin VV, Chizhma SN2012 System of secure data transfer based on quasiorthogonal signals J. Achievements of Modern Radio electronics 11 pp 26-31