Optimization of the coherence function estimation for multi-core central processing unit

A G Cheremnov[1], V A Faerman[2] and V S Avramchuk[3]

Tomsk Polytechnic University, Tomsk, Russian Federation

E-mail: [1]agc1@tpu.ru, [2]vaf@tpu.ru, [3]avs@tpu.ru

Abstract. The paper considers use of parallel processing on multi-core central processing unit for optimization of the coherence function evaluation arising in digital signal processing. Coherence function along with other methods of spectral analysis is commonly used for vibration diagnosis of rotating machinery and its particular nodes. An algorithm is given for the function evaluation for signals represented with digital samples. The algorithm is analyzed for its software implementation and computational problems. Optimization measures are described, including algorithmic, architecture and compiler optimization, their results are assessed for multi-core processors from different manufacturers. Thus, speeding-up of the parallel execution with respect to sequential execution was studied and results are presented for Intel Core i7-4720HQ и AMD FX-9590 processors. The results show comparatively high efficiency of the optimization measures taken. In particular, acceleration indicators and average CPU utilization have been significantly improved, showing high degree of parallelism of the constructed calculating functions. The developed software underwent state registration and will be used as a part of a software and hardware solution for rotating machinery fault diagnosis and pipeline leak location with acoustic correlation method.

1. Introduction

Solution of the number modern mechanical engineering problems requires applying of information technologies. Despite the fact that the most of these tasks are simple in terms of computations, some of them (as real-time control, prediction of technological process parameters or automated fault diagnosis) still require a huge amount of computation.

Continuous increase in volume of processed information and development of new signal processing methods requiring more computational resources lead to development of computation technology and facilitate appearance of new approaches to efficient use of computer hardware.

Primary resources of modern personal computers (PCs) are their central processing units (CPUs) whose performance is largely determined by their clock rate [1, 2]. Due to that until recently the main way to increase performance was connected to increase of the clock rate. At the same time, raised clock rate causes increases in both power consumption and heat emission inside the computer. The latter was de-emphasized by node miniaturization in the production of processors. Currently, however, further miniaturization is deemed problematic due to principal limitations, thus raising performance by increasing the number of cores becomes ever more important [2].

Introduction of parallel processing into the modern CPUs architecture creates several requirements to computation structuring: scalability, portability and adaptivity [3]. Only compliance with conditions of parallel decomposition and agglomeration leads to full use of the processing power and brings performance close to optimal values. Provision of the latter requires application of different

approaches towards optimization and some specific technologies, such as Open MP, Intel TBB, Intel Click Plus and others [4].

This paper states and solves an optimization problem in coherence function evaluation with the aim to increase the computation performance on multi-core CPUs. This function is important in digital signal processing (DSP) and is commonly used for the spectral analysis of rotating machinery vibration signals and also for pipeline leak location with acoustic correlation method.Performance characteristics obtained as a result of the optimization are shown below.

The results are of significant practicality and will find their application in creation of a software and hardware solution for the technical condition assessment of mechanical engineering equipment, fault diagnosis of vehicles engines and transimission gearboxes. Necessity for effective use of hardware is determined by both significant amount of calculations and limited potential of hardware used in the project.

2. Coherence function in signal processing

The coherence function may be seen in Digital Signal Processing as an analogue of a mutual correlation function over frequency domain, reflecting a degree of linear dependence between harmonic components of the signals being analyzed [5]. Mathematical definition of the coherence function is a ratio of mutual spectrum of the signals and square root from product of their energy spectra [5]:

$$\gamma(f_k) = \frac{\left|E(P_{AB}(f_k))\right|}{\left(E(P_{AA}(f_k) \cdot E(P_{BB}(f_k)))\right)^{0.5}}, \tag{1}$$

where $E(\cdot)$ is an averaging operator for the ensemble over non-overlapping epochs; $P_{AA}(k)$ $P_{BB}(k)$ are energy spectra represented by digital real signal samples $s_A(t)$ and $s_B(t)$; $P_{AB}(k)$ is a mutual energy spectrum of the signals. The argument of the coherence function takes the values

$$f_k = k \cdot \frac{f_d}{2n}, \tag{2}$$

where f_d is signal sampling frequency $s_A(t), s_B(t)$; $k = 0,1,\ldots, n$; $n+1$ is a number of spectral samples.

Energy spectra and the mutual spectrum may be obtained through the following calculation:

$$P_{AA}(f_k) = F_D^*(s_A(t_i)) \cdot F_D(s_A(t_i)) = \left|F_D(s_A(t_i))\right|^2, \tag{3}$$

$$P_{BB}(f_k) = F_D^*(s_B(t_i)) \cdot F_B(s_B(t_i)) = \left|F_D(s_B(t_i))\right|^2, \tag{4}$$

$$P_{AB}(f_k) = F_D^*(s_A(t_i)) \cdot F_D(s_B(t_i)), \tag{5}$$

where F_D is direct discrete-time Short-Time Fourier Transform (STFT); F_D^* is a complex conjugate of the STFT results. The studied signals are defined over a time interval

$$t_i = i \cdot \frac{1}{f_d}. \tag{6}$$

The number of samples is determined by chosen width of a STFT chunk frame.

The coherence function is an important tool for signal processing allowing justified determination of a frequency domain where a stationary determined signal appears that carries information, [5] and then apply necessary digital bandpass and barrier filters [6].

3. Optimization of the coherence function evaluation

From the programming point of view the computational procedure may be represented as a main loop which is executed a preset number of times. At that, each iteration consists of data sampling to form the STFT chunk frames, calculation of the signal spectra and their step-by-step accumulation as per (1). Final value of the coherence function is determined at the loop termination by substituting obtained values into the equation (1).

Application of the coherence function to solution of practical tasks in signal processing, particularly that of leak location in pipelines with an acoustic emission method assumes treatment of large volumes of data. The latter imposes significant constraints on the software implementation for the project, making it more expensive, particularly if there is a necessity to provide an operating mode close to real-time. Due to that parallel processing is a preferred way to arrange the computations.

As was mentioned before, use of the coherence functions requires significant number of STFT evaluations. The Cooley-Tukey algorithm [7] for Fast Fourier Transform with time decimation over a fixed base of 2 was selected as an algorithm to calculate STFT to increase computation performance speed. Choice of the algorithm is determined by its ease of implementation, clarity and high degree of internal parallelism [8].

Additional increase in STFT performance may be achieved by preliminary treatment of data, namely by its bit-reversal (change of bit order in the number's binary representation to the opposite) [7, 8]. In this case only array elements' indexes are subjected to the transformation. As a result, an order of the input data is changed, while the numerical values are the same.

Parallelizing was performed with OpenMP and Intel Threading Building Blocks (TBB) technologies. Choice of OpenMP was determined by spacial localization of the data, while tools from the Intel TBB library were used to arrange recursive traversal of the ‘butterfly’ graph during the STFT evaluation with time decimation following the Cooley-Tukey method, because this approach suggests parallelism of tasks, rather than that of data. Built-in graphic subsystem is used as an additional co-processor to execute general calculations.

Despite significant increase in calculation performance due to the above mentioned measures the software implementation had been far from ideal. Further runtime optimization was attained with scalar optimization methods [7]. The scalar optimization includes constant folding, constant propagation and copy propagation. Constant folding is a process of preliminary calculation of constants during compilation of the application; constant propagation is substitution of previously calculated constants into equations; copy propagation is a process to substitute variables with their respective values [7].

To reduce time burden for resource distribution during parallel calculations, a manual loading of data into CPU cache from the RAM is implemented [9]. This is achieved with the data prefetching function which fetches into the into the CPU cache a 64 bytes-long line of data starting from a given address .

Practical effect manifests as reduced data access time for the soon-requested data. The reason for this is that such data are likely already in the CPU cache as a result of this manipulations.

The final optimization measure is making the mathematical core with the optimizing Intel C++ compiler.

4. Assessment of optimization and speed

In accordance with the algorithm outlined above a software solution has been developed for experimental assessment of the coherence function estimation speed. Studies were performed using two CPUs: Intel Core i7-4720HQ and AMD FX-9590. Total number of signal samples $\boldsymbol{N}$ varied from 524288 to 4194304 samples. The time to calculate the coherence function was measured one thousand times.

Because there is a multitude of factors (including pseudoparallel operation of the OS core process scheduler, interrupt processing) that can slower but not accelerate calculations [10], the shortest time was held as the execution time to increase fidelity of the study. Time measurements were carried out with a special function tbb::tick_count() in the Intel TBB library that takes into account possible changes in the processor speed during the calculations.

As an example there are results of speed assessment for calculations with Intel Core i7-4720 HQ given in Table 1.

Table 1. Execution time assessment for Intel Core i7-4720HQ CPU

Chunk frame width	Number of signal samples (N)	Sequential execution time	Parallel execution time
32768	524288	0.1801	0.0622
	1048576	0.3603	0.1241
	2097152	0.7413	0.2541
	4194304	1.4673	0.5057
65536	524288	0.1927	0.0643
	1048576	0.3875	0.1307
	2097152	0.7815	0.2586
	4194304	1.5604	0.5206
131072	524288	0.2089	0.0692
	1048576	0.4061	0.1382
	2097152	0.8245	0.2725
	4194304	1.6316	0.5195

Dependences of the acceleration parameter [8] of the sample size for different chunk size are given in Figures 1 and 2.

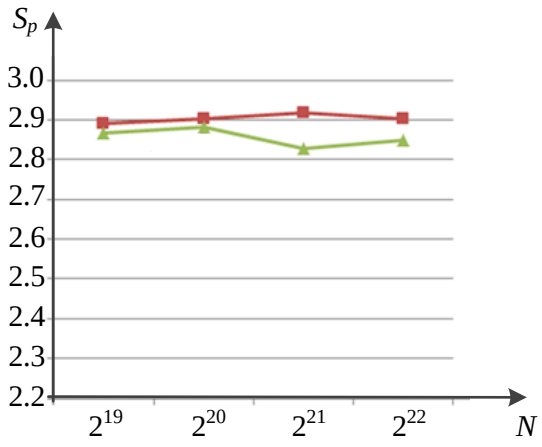

Figure 1. Dependence of acceleration on the sample size for the chunk frame size of 32768 (green – AMD FX-9590, red – Intel i7-4720 HQ).

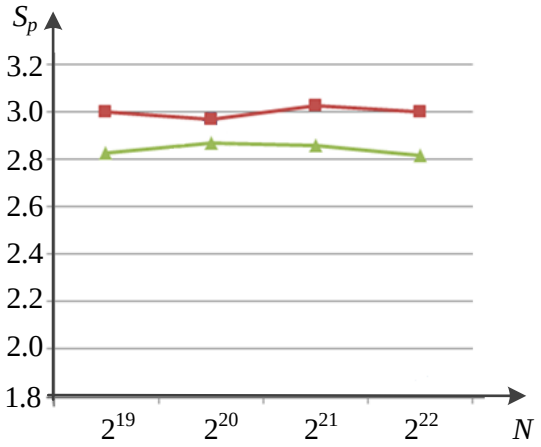

Figure 2. Dependence of acceleration on the sample size for the chunk frame size of 65536 (green – AMD FX-9590, red – Intel i7-4720 HQ).

Acceleration values are insignificantly reduced with the change of the initial sample, thus indicating good scalability of the software solution.

A study of load balancing was carried out during the calculations with the aim to determine a degree of load distribution. This study showed that the CPU utilization curve (when all cores are utilized) is almost symmetric with respect to the central value, thus indicating that there is a uniform distribution of load between the cores. However, from Figure 3 we may see, that only about 60% of the potential capability of the CPU is utilized, which we deem insufficient. Further studies with Intel

Parallel Studio debugger and IDA Pro disassembler helped to find the cause, which was sequential operation of the C++ memory allocation tools. Use of the Scalable_malloc function from the Intel TBB library allowed to increase performance by parallelizing memory allocation. At that, average CPU utilization reached 85%, which is reflected in Figure 4.

Figure 3. Memory usage and CPU utilization as functions of time, before optimization of memory allocation.

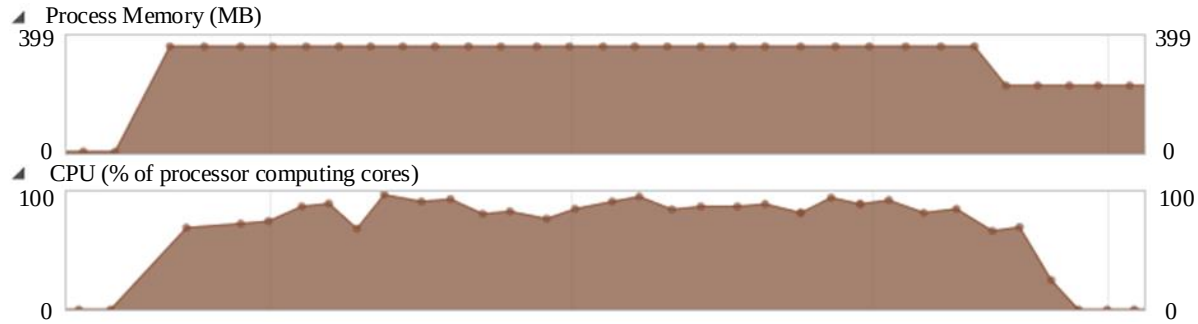

Figure 4. Memory usage and CPU utilization as functions of time, after optimization of memory allocation.

Thus, use of parallel memory allocator allowed to raise CPU utilization and by that increase performance.

For comparison indicators of both optimized and non-optimized versions of the coherence function estimation on the Intel Core i7-4720HQ CPU are given in Figures 5 and 6.

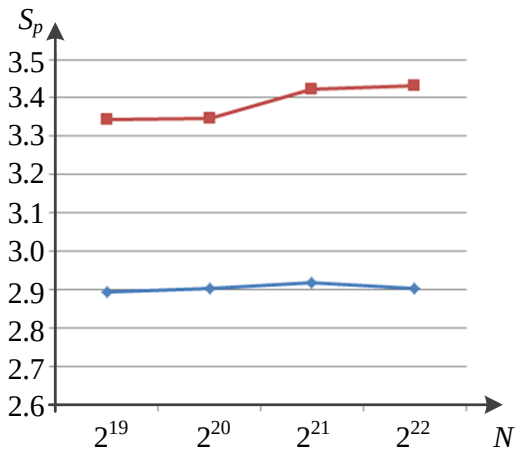

Figure 5. Acceleration as a function of sample size for the chunk frame size of 32768 (blue –

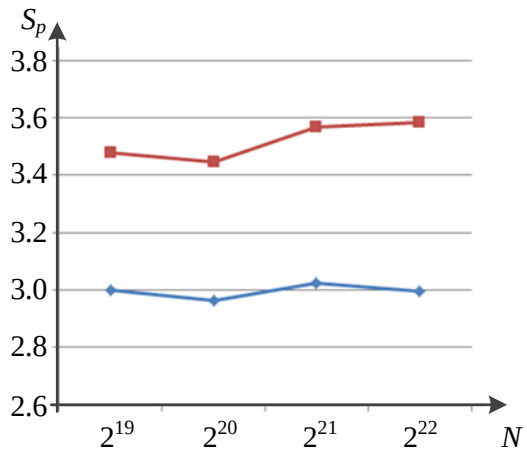

Figure 6. Acceleration as a function of sample size for the chunk frame size of 65536 (blue –

AMD FX-9590, red – Intel i7-4720 HQ). AMD FX-9590, red – Intel i7-4720 HQ).

5. Conclusion

This paper outlines a solution of a calculation optimization task for calculation of the coherence function on multi-core CPUs. This task is of great applied significance because the coherence function is widely used in solving different tasks of digital signal processing, for example in pipeline leak location with correlation acoustic method.

Introduction of parallelism is deemed the most approachable way to raise hardware resources utilization without significant additional costs. In this case a PC with a modern multi-core CPU serves as a computational platform.

Compiler, architecture and algorithmic optimization methods were employed to increase parallelizing performance during the coherence function estimation. The obtained data allowed for development of specialized software that provides automated selection of efficient division of the task into subtasks for parallel data processing as well as for selection between sequential and parallel methods of data treatment depending on the sample size.

The performed analysis of the coherence function parallelizing attests to efficiency of its software implementation. In particular, manual data prefetching into the cache line of the CPU significantly increases performance because there is no need for information exchange with the RAM during the calculations.

Further increase of performance may be attained by delegating the most resource-consuming part of the task, i.e., fast Fourier transform to graphics processors using such technologies as CUDA or OpenCL [11].

The software that has been developed during this study underwent state registration with the authorities of the Russian Federation and will be included into a hardware and software solution for rotating machinery fault diagnosis and pipeline leak location with acoustic correlation method.

Acknowledgements

The reported study was funded by Russian Foundation for Basic Research according to the research project No. 16-37-00049 mol_a.

Reference

[1] Blake G, Dreslinski R G and Mudge T 2009 *IEEE Signal Proc. Mag.* **26(6)** 26-37
[2] Herlihy M 2007 *Lect. Notes Comput. Sc.* **4855** 1-8
[3] Shukla S K, Murthy C N S and Chande P K 2015 *Adv. in Intelligent Syst. and Comp.* **1089** 537–545
[4] Kim H and Bond R 2009 *IEEE Signal Proc. Mag.* **26(6)** 1-8
[5] Carter C G 1987 *Proc. of IEEE* **75(2)** 236-255
[6] Ionel R, Ionel S and Ignea A 2010 *Proc. of Int. Joint Conf. on Computational Cybernetics and Technical Informatics (Timisoara)* (IEEE: USA) 5491234
[7] Avramchuk V S, Luneva E E and Cheremnov A G 2014 *Proc. of Int. Conf. on Mechanical Engineering, Automation and Control Systems* (IEEE: USA) 6986858
[8] Gupta A and Kumar V 1993 *IEEE Parall. Distr.* **4(8)** 922-932
[9] Avramchuka V V, Luneva E E and Cheremnov A G 2014 *Adv. Mat. Res.* **1040** 969-974
[10] Manikandan N and Subha S 2016 *Int. J. of Pharmacy and Techn.* **8** 3916-27
[11] Skirnevskiy I and Korovin A 2015 *Key Eng. Mat.* 685857-862