RESEARCH OF IMPROVING THE PERFORMANCE OF EXPLICIT NUMERICAL METHODS ON THE X86 AND ARM CPU

V. Furgailo[a], E. Elchinov, N. Khohlov

Russia, Moscow Institute of Physics and Technology (MIPT)

E-mail: [a] furgailo@phystech.edu

This paper is a continuation of the research of improving the computing performance of explicit numerical methods on the CPU. We considered the computing possibility of such common computing architectures as x86 and arm, for using optimizations on the data layer as vectorization and tiling. Other aspects of high-performance optimizations of explicit numerical methods have also been explored - metaprogramming, code generation, and OpenMP technology. However, the novelty of this research is the optimization of the arm architecture for the task of computing by the FDTD method and the assessment of the effectiveness of using the arm architecture for solving such a range of scientific problems. This paper considers a number of optimization algorithms, provides a description of the algorithms, test calculations for various architectures.The results of the re search and further directions of work on this topic are also presented.

Keywords: expicit numerical methods, FDTD, tiling, x86, arm

Vladislav Furgailo, Egor Elchinov, Nikolay Khohlov

1. Introduction

Explicit numerical methods are used to solve and simulate a wide range of mathematical problems whose origins can be mathematical models of physical conditions. However, simulations with large model spaces can require a tremendous amount of floating point calculations and run times of several months or more are possible even on large HPC systems.

The vast majority of HPC systems in the field today are powered by x86 and ARM CPUs [1]. Our aim is to investigate methods of increasing computational speed for simulation on CPUs and also to compare the performance and energy efficiency on x86 and ARM CPUs. High-order finite difference time domain (FDTD) method to solve the 3D acoustic equation was used in our work.

For HPC, in conjunction with parallel computing, we used CPU capabilities like SIMD-computing (AVX on x86 and NEON on ARM) [2] and hierarchical structure of the memory of the CPU caches to optimize data locality. For data locality was used the method of changing order of traversal on the iteration space – loop tiling [3]. Our work considers a number of optimization tiling algorithms and test calculations for x86 and ARM architectures. In particular, we considered recursive and non-recursive cube-tiling [4] and ZCube data locality optimization.

2. Mathematical Problem

In this paper, an explicit numerical method is the solution of the acoustic equation in three dimensional space with free-boundary condition by the central finite difference time domain (FDTD) method with a fourth-order accuracy. Thus, the stencils can be represented in Fig.1

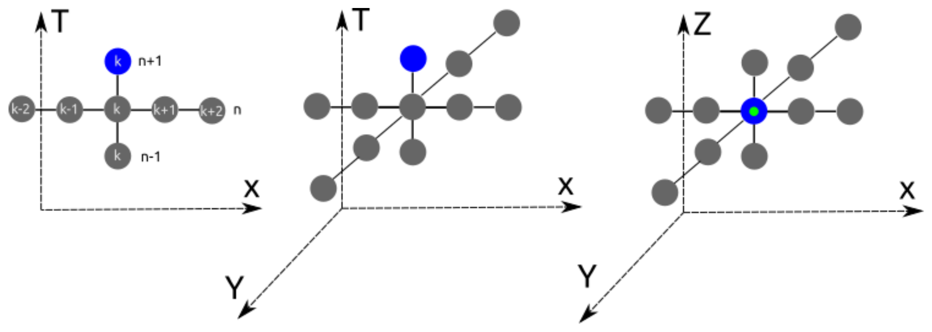

Figure 1. Stencils for 1D, 2D, 3D wave-equation

3. Optimization algorithms

3.1 Tiling

To increase the performance of the computation, the method of partitioning the iteration space into cube-tiles was used with vectorization. The algorithm shows the highest performance on tiles that are the same size as the first level (L1) of the CPU cache, as shown in fig.2. and fig 3.

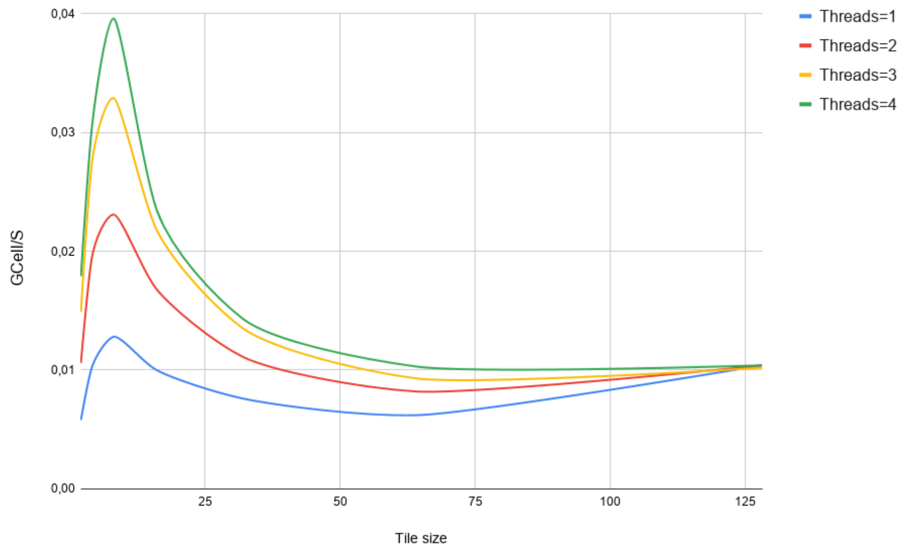

Figure 2. Graph of the number of points computed per second from a fixed tile size on various threads. Grid size varies. Computed on ARM - CortexA53.

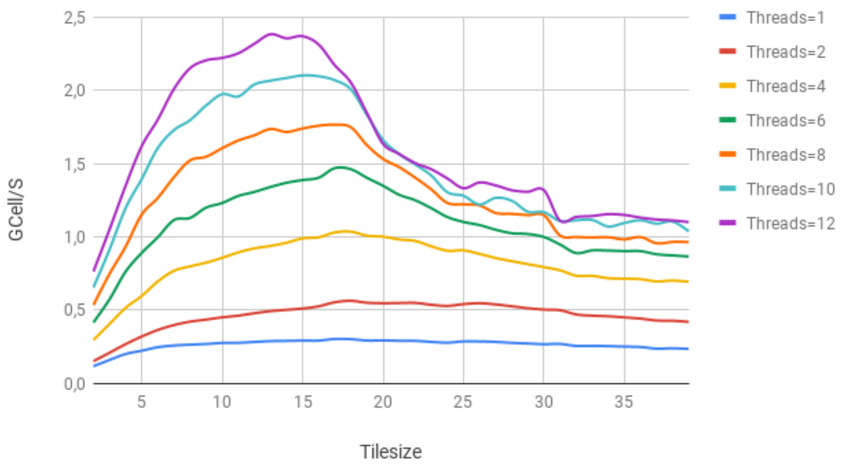

Figure 3. Graph of the number of points computed per second from a fixed tile size on various threads. Grid size varies. Computed on x86 - Intel(R) Xeon(R) CPU E5-2620 v2.

3.2 ZCube

To increase the spatial locality a new data storage principle was used, such that the grid cells are grouped into small cubes forming a new type of cell, each storing original cells in Z-order, as represented on fig 4.

Combine ZCube with recursive tiling or nested tiles, we got the following results - as shown on fig.5 the problem of low performance of Neon-computing occurs due to an overflow of Cortex-A53 data cache[5]. Consequently, ZCube tiling improve the performance of utilizing Neon data-registers and instructions. Also on x86, as shown on fig.6, combination AVX-vector and ZCube recursive tiling provides up to 2-times speedup compared to the naive implementation.

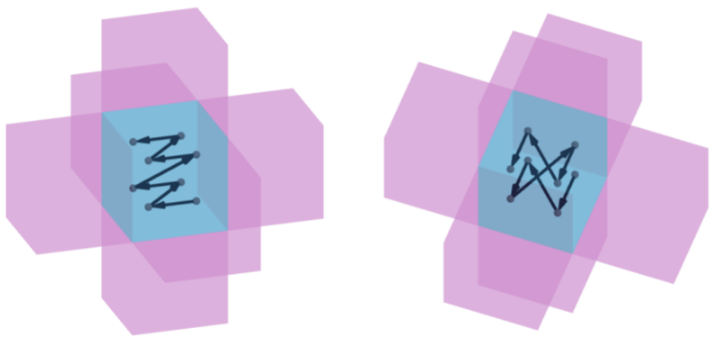

Figure 4. Z-cube 2×2×2, with stencil representation

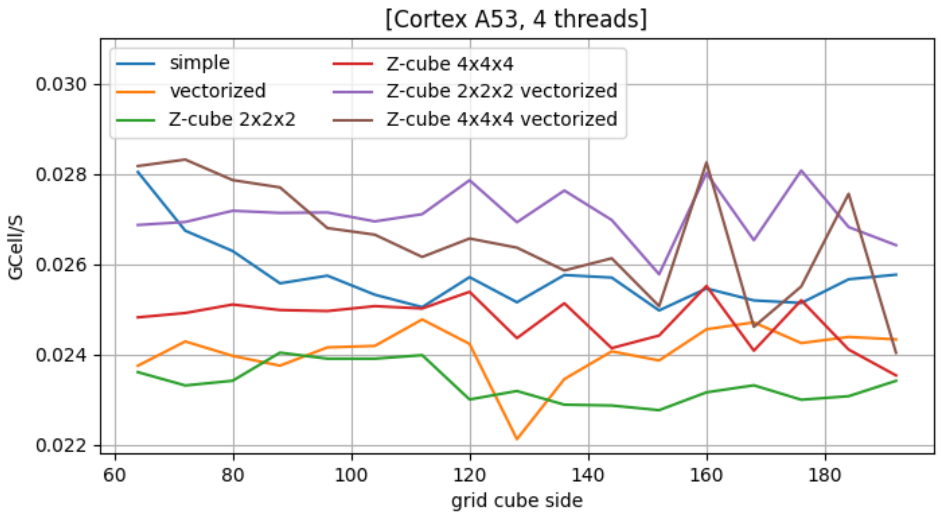

Figure 5. ZCube efficiency for the variable grid size on ARM.

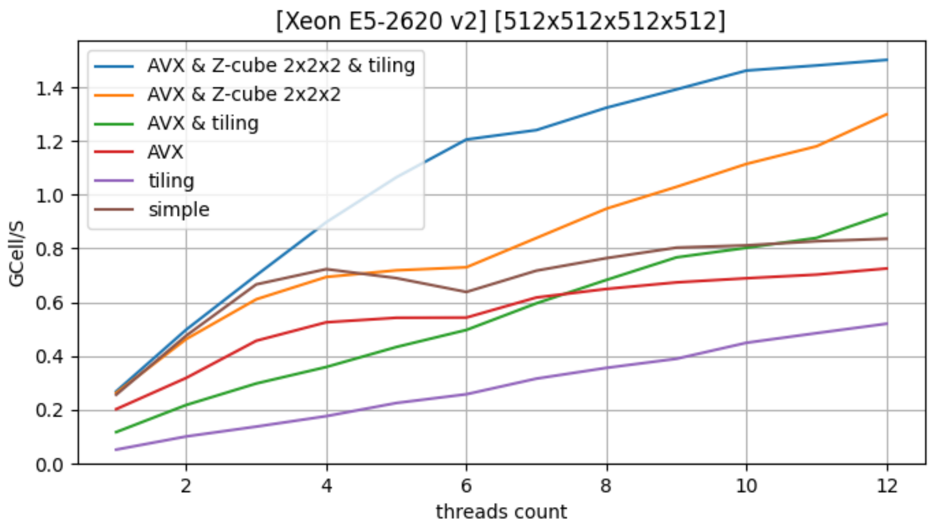

Figure 6. ZCube efficiency for the variable grid size on x86

4. Conclusion

In this paper, we implemented a new data locality algorithm that increases the performance of multi-threaded explicit stencil computation. Vectorization over the outer space of iterations and Z-Cube recursive tiling were applied to achieve data locality and to speed up multithreaded computing. However, non-recursive tiling remains a more effective data localization algorithm to FDTD problem.

Also, as shown in Fig. 7, the computation of explicit numerical equations on the ARM architecture by non-recursive tiling is 12 times more energy efficient in peak power consumption. In this respect, extending our experiments on ARM-cluster computing with increasing performance of non-recursive and recursive tiling would be of interest.

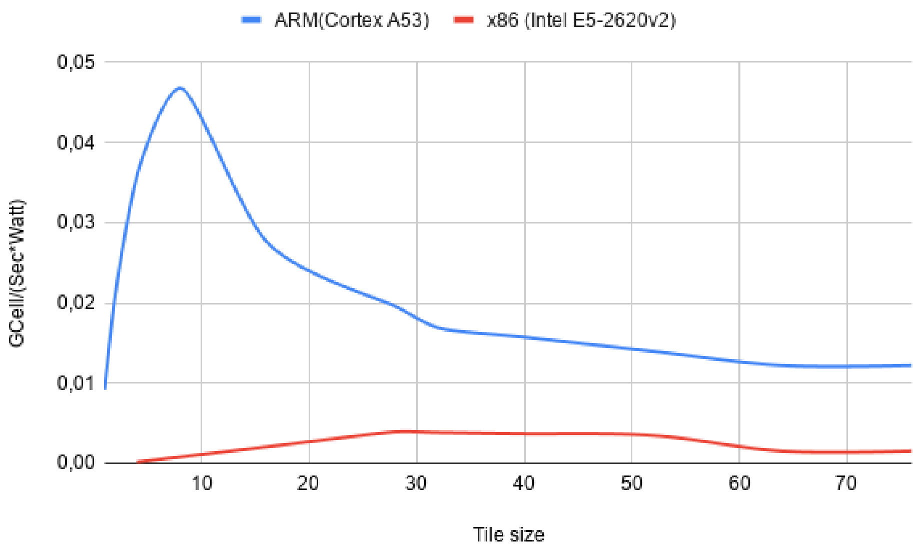

Figure 7. Graph of non-recursive vectorized tiling performance/power effectiveness ofx86 and ARM.

5. Acknowledgement

This work was performed with the financial support of the Russian Science Foundation (project No. 21-11-00139).

References

[1] http://www.top500.org/
[2] S. M. et. al., “Vector instructions to enable efficient synchronization and parallel reduction operations,” U.S. Patent WO2009120981A2, Oct. 2009.
[3] J. Xue, “On tiling as a loop transformation,”Parallel Processing Letters, vol. 07,no. 04, pp. 409–424, 1997.
[4] V. Furgailo, A. Ivanov, and N. Khokhlov, “Research of techniques to improve the performance of explicit numerical methods on the cpu,” pp. 79–85, 09 2019.
[5] J. Bakos,Embedded Systems: ARM Programming and Optimization. Elsevier Sci-ence, 2015.