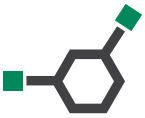


cost

Exploring OpenMP Accelerator Model in a real-life scientific application using hybrid CPU-MIC platforms

Kamil Halbiniak*, Lukasz Szustak*, Alexey Lastovetsky** and Roman Wyrzykowski*

* Czestochowa University of Technology, Poland
{khalbiniak, lszustak, roman}@icis.pcz.pl

** University College Dublin, Ireland
alexey.lastovetsky@ucd.ie

Abstract

The main goal of this paper is the suitability assessment of the OpenMP Accelerator Model (OMPAM) for porting a real-life scientific application to heterogeneous platforms containing a single Intel Xeon Phi coprocessor. This OpenMP extension is supported from version 4.0 of the standard, offering an unified directive-based programming model dedicated for massively parallel accelerators. In our study, we focus on applying the OMPAM extension together with the OpenMP tasks for a parallel application which implements the numerical model of alloy solidification. To map the application efficiently on target hybrid platforms using such constructs as omp target, omp target data and omp target update, we propose a decomposition of main tasks belonging to the computational core of the studied application. In consequence, the coprocessor is used to execute the major parallel workloads, while CPUs are responsible for executing a part of the application that do not require massively parallel resources. Effective overlapping computations with data transfers is another goal achieved in this way. The proposed approach allows us to execute the whole application 3.5 times faster than the original parallel version running on two CPUs.

Keywords Intel MIC, hybrid architecture, numerical modeling of solidification, heterogeneous programming, OpenMP Accelerator Model, task and data parallelism

I. Introduction

Heterogeneous platforms combining general-purpose processors with specialized computing accelerators (e.g., GPU or Intel Xeon Phi) offer ample opportunities for accelerating a wide range of applications [1]. However, realizing these performance potentials remains a challenging issue.

A promising way to exploit capabilities of heterogeneous platforms is the OpenMP Accelerator Model [2] offered by the OpenMP standard, starting with version 4.0. It provides an unified directive-based programming model encompassing both CPUs and accelerators. The major advantage of this extension is applying the same programming model for the whole application, that allows decreasing the code complexity and increasing its portability.

The main goal of this paper is evaluation of the OpenMP Accelerator Model for porting a real-life scientific application to platforms equipped with a single Intel Xeon Phi coprocessor. In this study, we focus on the effective utilization of new mechanisms provided by the OpenMP 4.0 standard for parallelization of the computational core of the studied application. The proposed approach allows us to execute computations 3.49x faster than the original parallel code that uses two CPUs. This application was already studied in our previous work [3], where we developed a methodology that utilized the dedicated Intel Offload interface.

This paper is organized as follows. Section 2 gives an overview of the OpenMP Accelerator Model, while Section 3 introduces the numerical model of solidification, which is based on the generalized finite difference method. The next section describes the idea of parallelizing the solidification application on hybrid platforms with OpenMP 4.0 mechanisms, while Section 5 shows performance results achieved by the proposed approach. Section 6 concludes the paper.

II. Overview of OpenMP Accelerator Model

OpenMP is the directive-based programming standard designed for programming shared-memory systems. [2]. Starting with version 4.0, OpenMP provides a mechanism called

OpenMP Accelerator Model (OMPAM in short). It aims at simplifying the issue of programming heterogeneous computing platforms with many-core accelerators such as Intel MIC or GPU. This model assumes that a computing platform is equipped with multiple target devices connected to the host device.

The execution model of OMPAM is based on a host-centric view, where the host device transfers (offloads) data and computations to target devices before execution, using **target** construct. By default, code regions offloaded to accelerators are executed using a single thread, that can spawn multiple threads after encountering an appropriate parallel construct.

Using accelerators requires usually to perform data transfers. To reduce the total amount of allocations and deallocations of device memory, OMPOA provides **target data** construct, which creates the data region for a device. This gives the possibility for sharing the same data between multiple target regions. OMPAM allows defining the data movements between the host and the device before and after the execution of the offloaded region by using **map** clause. The transfers of data are possible using the following attributes: **to**, **from** and **tofrom**. These attributes allows the implicit initialization of device buffers and determination of the direction of data copying [2]. At the same time, **map** clause with **alloc** attribute is used when the explicit allocation of device memory is required.

Another important directive of OMPAM is **target udpate**. It allows the synchronization of buffers between the host and device environments. This construct can be used only inside the device data region. The direction of update is specified using two clauses: **to** and **from**, which provide the list of synchronized buffers consistent with variables in the device data region. Another new directive, **declare target**, is used to determine regions of the source code mapped to the device, with the resulting binaries called from the **target** region.

An example of source code written using the OpenMP Accelerator Model is shown in Listing 1.

```
#pragma omp target data map(to: n, B[0:n]) \
    map(alloc: A[0:n], C[0:n])
for(int t=0; t<num_steps; ++t) {
    #pragma omp target map(to: n, B[0:n]) \
        map(to: C[0:n]) map(from: A[0:n])
    #pragma omp parallel for
    for (int i=1; i<n-1; ++i ) {
        A[i]=C[i]*(B[i-1] + B[i] + B[i+1]);
    }
    //rest of code
}
```

Listing 1: Offloading computations in OpenMP Accelerator Model

Comparing to alternative tools that allow for programming accelerators, OMPAM provides a reasonable support for multiple heterogeneous platforms, through a growing amount of compilers. This increases the interest of developers in using OpenMP as a promising way to achieve the code portability between platforms.

III. Application: Modeling Solidification

The phase-field method is a powerful tool for solving interfacial problems in materials science. It has mainly been applied to solidification dynamics, but it has also been used for other phenomena such as viscous fingering, and fracture dynamics. The number of scientific papers related to the phase-field method grows since the 90 years of XX century, reaching for the last 7 years more than 400 positions (according to the SCOPUS database) [4].

In the numerical example studied in this paper, a binary alloy of Ni-Cu is considered as a system of the ideal metal mixture in the liquid and solid phases. The numerical model refers to the dendritic solidification process in the isothermal conditions with constant diffusivity coefficients for both phases. In the model, the growth of microstructure during the solidification is determined by solving a system of two PDEs which define the phase content ϕ (Fig. 1) and concentration c of the alloy dopant. The solutions of these PDEs are obtained on the basis of the generalized finite difference method and explicit scheme of calculations, so the resulting numerical algorithm [3] belongs to the group of forward-in-time iterative algorithms. In the model studied in the paper, values of ϕ and c are calculated for grid nodes uniformly distributed across a square domain. However, this model can be also used for irregular grids.

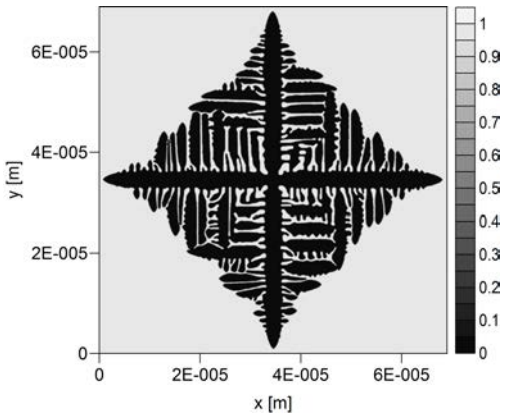


Figure 1: Phase content for the simulated time $t = 2.75 \times 10^{-3}s$

IV. Parallelization of the Application on Hybrid CPU-MIC Platforms

IV.1 Task Parallelization with OpenMP 4.0

In the studied application, computation are interleaved with writing partial results to a file. In the basic version (Fig.2a), parallel computations are executed for subsequent time steps, and writing results to the file is performed after the first time step, and then after every package of $R = 2000$ time steps.

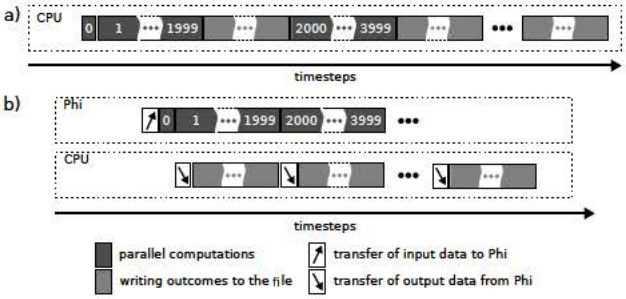


Figure 2: Adapting the application to platforms with Intel MIC [3]

In the proposed approach (Fig. 2b), the Intel Xeon Phi coprocessor is utilized to perform parallel computations, while the host processor is responsible for executing the rest of the application that not required massively parallel resources. As a result, writing outcomes to the file is assigned to the CPU, while the coprocessor is utilized for parallel computations in subsequent time steps. At the beginning, all the input data are transferred from the CPU to the coprocessor, which then starts computations for the first time step. After finishing it, all the results are transferred back to the CPU. During this transfer, the coprocessor starts computations for the next package of R time steps. At the same time, CPU begins writing results to the file, immediately after receiving outcomes from the coprocessor. Such a scheme is repeated for every package of R time steps. A critical performance challenge here is to overlap workload performed by the coprocessor with data movements. To reach this goal, data transfers between the CPU and coprocessor, writing data to the file, as well as computations have to be performed simultaneously.

To offload data and computations to the coprocessor, we use two major constructs of OMPAM: **omp target data** and **omp target**. By default, OpenMP 4.0 does not provide a mechanism for the asynchronous execution of **omp target** region. In consequence, the thread calling this pragma is stopped before completing the execution by the accelerator. Therefore to ensure overlapping computations with writing outcomes to the file, we propose to use the OpenMP task parallelism. This mechanism can be successfully applied to parallelize these operations. As a result, two tasks are distinguished in the proposed approach: (i) running parallel computations on the coprocessor, and (ii) writing results to the file. These tasks are spawned inside the parallel region by the master thread using **omp task** construct.

When applying the proposed idea of adapting the solidification application to heterogeneous platforms (Fig. 2b), the usage of task parallelism requires to provide an adequate task synchronisation, since results cannot be written to the file before completing computations for the previous package of R time steps. Therefore, the synchronization points occur after every package. To ensure the effective synchronization of tasks, we use **omp taskwait** pragma.

Overlapping data transfers with computations requires also to apply the double-buffering technique. The first buffer is utilized for performing parallel computations, while the second one is for providing data movements and writing outcomes to the file. To transfer data from the coprocessor to CPU during computations, **omp target update** construct is adopted.

IV.2 Data Parallelization

The original CPU version of the application uses the OpenMP standard to utilize cores/threads, based on the OpenMP construction `#pragma omp parallel for`. Since the Intel Xeon Phi coprocessors supports OpenMP, the application code can be rather easily ported to this platform. To ensure the best overall performance without significant modifications in the source code, we use several compiler-friendly optimizations, empirically determine the best OpenMP setup for the loop scheduling and set appropriate affinity.

The utilization of vector processing is crucial for ensuring the best performance on Intel Xeon Phi. The quickest way to achieve this goal is the compiler-based automatic vectorization. However, in the studied case the innermost loop cannot be vectorized safely, mainly because of data dependencies. To solve this problem, we propose to change slightly the code by adding temporary vectors responsible for loading data from the irregular memory region, and than providing SIMD computations (see our previous works [3, 5]).

V. Performance Results

In this work, we use a platform equipped with two Intel Xeon E5-2699 v3 CPUs (Haswell-EP), and Intel Xeon Phi 7120P coprocessor (Knight Corner). All the tests are performed for the double-precision floating-point format with 110 000 time steps, and 2D grid containing 4 000 000

Table 1: Performance results for different versions of the application

Version	Tasks		Time	Speedup
	Data writing	Parallel computing		
original	CPU	CPU	641 min 32 sec	-
offload	CPU	MIC	183 min 41 sec	3.49x
OpenMP	CPU	MIC	187 min 43 sec	3.42x

nodes (2000 nodes along each dimension), using the Intel icpc compiler (v.15.0.2) with optimization flag -O3.

Table 1 presents the comparison of the performance for: (i) original CPU parallel version of the application running on two CPUs with 18 cores each, (ii) offload-based code for hybrid CPU-MIC platforms, developed in our previous work [3], and (iii) the proposed version developed in this work using OpenMP 4.0 mechanisms. Both the second and third versions implement the proposed scheme of adapting the solidification application to platforms with a single Intel Xeon Phi coprocessor.

The total execution time of the original code is the sum of the execution times necessary for performing parallel computations and writing partial results to the file. The proposed approach allows us to hide more than 99% of data movements behind computations, for both the offload- and OpenMP-based versions, and finally accelerate the whole application of about 3.5x. Comparing the execution times for the OpenMP- and offload-based codes, we can see a very low difference of 2.2% in favour of the offload interface.

VI. Conclusion and Future Works

This paper shows that the OpenMP Accelerator Model is the promising tool for porting a real-life scientific application to heterogeneous platforms with many-core accelerators such as Intel Xeon Phi. The performance results obtained for the offload-based and OpenMP-based versions, executed on the platform with a single coprocessor, confirms that the OpenMP Accelerator Model allows achieving a quite similar performance as the Intel Offload Model dedicated directly for Intel MIC architectures. It is expected that the potential of using OpenMP for current and future architectures will be manifested for a wide range of applications.

The primary direction of our future work is to take advantage of all the computing resources (multiple CPUs and multiple MICs) of heterogeneous platforms, for executing the application. The OpenMP Accelerator Model will be compared against the offload-based [5] and hStreams-based [6, 7] solutions, taking into account both the performance and productivity. We plan to explore new features available in version 4.5 of OpenMP [8], since version 4.0 does not provides the asynchronous offload mechanisms, which are necessary for the efficient utilization of all the resources available in such multi-device heterogeneous platforms.

Acknowledgments

This research was conducted with the support of COST Action IC1305 (NESUS), as well as the National Science Centre (Poland) under grant no. UMO-2011/03/B/ST6/03500. The authors are grateful to the Czestochowa University of Technology for granting access to Intel Xeon Phi coprocessors provided by the MICLAB project no. POIG.02.03.00.24-093/13 (http://miclab.pl).

References

[1] L. Szustak, K. Rojek, R. Wyrzykowski, and P. Gepner. Toward efficient distribution of MPDATA stencil computation on Intel MIC architecture. In *Proc. 1st Int. Workshop on High-Performance Stencil Computations (HiStencils' 14)*, pages 51–56, 2014.

[2] OpenMP Application Programming Interface. http://www.openmp.org/mp-documents/OpenMP4.0.0.pdf.

[3] L. Szustak, K. Halbiniak, A. Kulawik, J. Wrobel, and P. Gepner. Toward parallel modeling of solidification based on the generalized finite difference method using Intel Xeon Phi. *LNCS*, 9573:411–412, 2016.

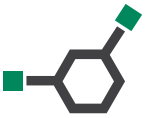
[4] T. Takaki. Phase-field Modeling and Simulations of Dendrite Growth. *ISIJ International*, 54 (2):437–444, 2014.

[5] L. Szustak et al. Porting and optimization of solidification application for CPU-MIC hybrid platforms. *Int. Journal of High Performance Comp. Applications*, (accepted to print).

[6] Chris J. Newburn et al. Heterogeneous streaming. *IPDPSW, AsHES*, 2016.

[7] L. Szustak et al. Using hStreams Programming Library for Accelerating a Real-Life Application on Intel MIC. In *ICA3PP 2016 Conference*, (accepted to print).

[8] M. Klemm. Heterogeneous Programming with OpenMP 4.5. https://www.scc.kit.edu/downloads/sca /Heterogeneous%20Programming%20with%20OpenMP%204.5.pdf.



cost

Geographical Competitiveness for Powering Datacenters with Renewable Energy

Simon Holmbacka*, Enida Sheme†, Sébastien Lafond*, Neki Frasheri†

*Factulty of Science and Engineering, Åbo Akademi University Turku, Finland
firstname.lastname@abo.fi
†Polytechnic University of Tirana Tirana, Albania
firstname.lastname@fti.edu.al

Abstract

In this paper we analyze the feasibility of using renewable energy for powering a data center located on the 60th parallel north. We analyze the workload energy consumption and the cost-energy trade-off related to available wind and solar energy sources. A wind and solar power model is built based on real weather data for three different geographical locations, and The available monthly and annual renewable energy is analyzed for different scenarios and compared with the energy consumption of a simulated data center. We show the impact different data center sizes have on the coverage percentage of renewables, and we discuss the competitiveness of constructing datacenters in different geographical location based on the results.

Keywords Green energy, datacenter, simulation, geographical locations

I. Introduction

The global energy price and tighter restrictions on energy production has led to a higher utilization of green energy, which is produced from completely carbon neutral sources. One of the latest trends in reducing the carbon footprint of data centers is powering the datacenters with renewable sources of energy. This course is being encouraged by the advances of renewable technologies and continuously decreasing renewable energy costs. Renewable energy sources have become an interesting option for large scale server farms, and initiatives such as Google Green[1] and Facebook Sustainability[2] have been taken to decrease the carbon footprint both for ecological and monetary reasons. Recently, the location of large scale server farms has shifted to the nordic countries above the 60th parallel because of a cooler climate, which in turn reduces the cooling costs for such datacenters. The energy required for computation and the infrastructure must, however, be delivered from the electric grid, preferably generated by renewable energy. This poses a challenges for northern countries because of the large variation in available solar energy throughout the year. While the summer period provides from 18 to 20 hours of sunlight, the winter period provides merely a few hours – this from a very shallow angle of reflection. The lack solar energy can be compensated with other sources such as wind energy, but the total cost of powering the data center must be sufficiently low in order to stay competitive to other geographical locations.

We present in this paper a thorough analysis of the feasibility of powering large scale datacenters in geographical locations above the 60th parallel north with renewable energy. The analysis contains simulations of different datacenters executing various workloads and the requirements in green energy production for different geographical locations. In contrast to previous work we compare different geographical locations in terms of both available renewable energy and required datacenter capacity for satisfying the end user. We also provide an a competitiveness factor between different geographical locations for the feasibility of powering a datacenter with renewable energy sources.

II. Related Work

Real implementations of green data centers. Researchers at Rutgers University [9] present Parasol and GreenSwitch, a research platform for a green data center prototype. It consists of GreenSwitch software running over a real hardware data center, Parasol. Its aim is reducing the total data center cost by properly managing workloads and available energy sources for maximum benefits. It also studies the space requirements and capital costs of self-generation with wind and solar energy. Similarly, [18] presents Blink, a physical implementation of using intermittent power to supply a cluster of 10 laptops by two micro wind turbines and two solar

[1]https://www.google.com/green/
[2]https://sustainability.fb.com/en/