Parallelization of EULAG Model on Multicore Architectures with GPU Accelerators

Krzysztof Rojek and Lukasz Szustak

Czestochowa University of Technology
Dabrowskiego 69, 42-201 Czestochowa, Poland
{krojek,lszustak}@icis.pcz.pl

Abstract. EULAG (Eulerian/semi-Lagrangian fluid solver) is an established computational model developed by the group headed by Piotr K. Smolarkiewicz for simulating thermo-fluid flows across a wide range of scales and physical scenarios.

In this paper we focus on development of the most time-consuming calculations of the EULAG model, which is multidimensional positive definite advection transport algorithm (MPDATA). Our work consists of two parts. The first part is based on the GPU parallelization using ATI Radeon HD 5870 GPU, NVIDIA Tesla C1060 GPU, and Fermi based NVIDIA Tesla M2070-Q, while the second one assumes the multicore CPU parallelization using AMD Phenom II X6 CPU, and Intel Xeon E3-1200 CPU with Sandy Bridge architecture. In our work, we use such standards for multicore and GPGPU programming as OpenCL and OpenMP.

The GPU parallelization is based on decomposition of the algorithm into several smaller tasks called kernels. They are executed in a FIFO order corresponding to the dependency tree expressing data dependencies between kernels. To optimize performance of the resulting implementation, we utilize the extensive vectorization of each kernel, as well as overlapping of data transfer with computations.

At the same time, when considering CPU parallelization we focus on multicore processing, vectorization and cache reusing. To achieve high efficiency of computations, the SIMD processing is applied using standard SSE and new AVX extensions. In this paper we provide performance analysis based on the Roofline Model, which shows inherent hardware limitations for MPDATA, as well as potential benefit and priority of optimizations. In order to alleviate memory bottleneck and improve efficient cache reusing, we propose to use the loop tiling technique.

1 Introduction

EULAG [3,5] (Eulerian/semi-Lagrangian fluid solver) is an established computational model developed by the group headed by Piotr K. Smolarkiewicz for simulating thermo-fluid flows across a wide range of scales and physical scenarios, such as numerical weather and climate prediction. EULAG is a representative of the class of anelastic hydrodynamic models.

Preliminary studies of porting anelastic numerical models to modern architectures, including GPUs, were carried out in work [10]. Selected parts of this

R. Wyrzykowski et al. (Eds.): PPAM 2011, Part II, LNCS 7204, pp. 391–400, 2012.

model were ported to ATI Radeon HD 5870 and NVIDIA Tesla C1060 cards. The achieved performance results show the potential gains in computing performance on modern computer architectures. The problem of adapting the EULAG model to modern hardware architectures was also brought up in [4]. The results achieved for porting selected parts of EULAG to NVIDIA GPUs, using an automatic approach as well, unveil potential in running scientific applications, including anelastic numerical models, on novel hardware architectures.

In this paper, we focus on parallelization of the most time-consuming algorithm of the EULAG model, which is MPDATA [10]. Our work consists of two parts. The first part is based on the GPU parallelization using NVIDIA and AMD architectures, while the second one assumes the multicore CPU implementation. In our work, we use such standards for multicore and GPGPU programming as OpenCL and OpenMP.

2 Architecture Overview

Our research is based on two kind of architectures. The first one are GPUs, while the second one are CPUs. We focus only on features of these architecture which are used in our work.

2.1 Architecture of GPUs

Our research is focused on NVIDIA Tesla C1060 and M2070-Q, as well as ATI Radeon HD 5870.

Architecture of NVIDIA Tesla. NVIDIA Tesla C1060 [6] includes 10 Thread Processing Clusters (TPC). Every TPC contains 3 compute units. Each compute unit consists of 8 processing elements, and 16KB of local memory. It gives a total number of 240 available processing elements with a clock rate of 1296 MHz. It provides a peak performance of $240 * 1.296 * 2 = 0.622$ Tflop/s in single precision. This graphics accelerator card includes 4 GB of global memory with the peak bandwidth of 102.4 GB/s.

NVIDIA Tesla M2070-Q [6] is based on Fermi architecture, that supports fully coherent L2 cache. It contains 448 available processing elements with a clock rate of 1147 MHz, so the peak performance is $448 * 1.147 * 2 = 1.03$ Tflop/s in single precision. This graphics accelerator card includes 6 GB of global memory with the peak bandwidth of 148.0 GB/s.

Architecture of ATI Radeon HD 5870. ATI Radeon HD 5870 [2] includes 20 compute units. Each compute unit consists of 16 processing elements, and 32KB of local memory. Each of the processing element is built of 5 streaming processors. It gives a total number of 1600 available streaming processors with a clock rate of 850 MHz, and provides the peak performance of $1600 * 0.850 * 2 = 2.72$ Tflop/s in single precision. This accelerator card includes only 1 GB of global memory with the peak bandwidth of 153.6 GB/s.

2.2 Architecture of CPUs

Our research is based on AMD Phenom II X6 CPU, and Intel Xeon E3-1200 CPU.

Architecture of AMD Phenom II X6. AMD Phenom II X6 CPU [1] contains of six cores with clock frequency of 3.2GHz. This CPU supports Streaming SIMD Extensions (SSE), which can greatly increase performance when exactly the same operations are to be performed on multiple data objects. Therefore, the peak performance of AMD Phenom II X6 processor is $3.2 * 6 * 4 * 2 = 153.6$ Gflop/s in single precision with SSE enabled, and $3.2 * 4 * 2 = 38.4$ Gflop/s without SSE.

Architecture of Intel Xeon E3-1200. Intel Xeon E3-1200 [8] is based on Sandy Bridge architecture, which is the first implementation of Intel Advanced Vector Extensions (AVX). This processor includes four cores with clock frequency of 3.4GHz. For Intel Xeon E3-1200 processor with AVX enabled, the theoretical performance is $3.4 * 4 * 8 * 2 = 217.6$ Gflop/s in single precision, and only $3.4 * 4 * 2 = 27.2$ Gflop/s without AVX.

Intel Advanced Vector Extensions Overview. Before Sandy Bridge Intel microarchitecture, the SIMD vectorization was provided by the Intel Streaming SIMD Extensions (Intel SSE). Intel SSE instructions use eight 128-bit registers where uniform type data can be packed, and enable operating on 4 float elements per iteration instead of a single element. The Intel AVX [8] offers a significant increase in the floating-point performance over previous generations of 128-bit SIMD instruction set extensions. AVX increases the number of registers from 8 to 16 and width of the registers from 128 bits to 256 bits. The new ability to work with 256-bit vectors enables operating on 8 float or 4 double elements per iteration, instead of a single element.

3 The Scope of Our Research on the EULAG Model

One of the most time-consuming calculations [10] of the EULAG model is multidimensional positive definite advection transport algorithm (MPDATA). In this work, we take into account the linear version of MPDATA, which is based on the following equation [7]:

$$\Psi_i^{n+1} = \Psi_i^n - \frac{\delta t}{\nu_i} \sum_{j=1}^{l(i)} F_j^{\perp} S_j, \tag{1}$$

where Ψ is a nondiffusive scalar field, S_j refers both to the face itself and its surface area, ν_i is the volume of the cell containing vertex i, while $F_j^{\perp}$ is interpreted as the mean normal flux through the cell face S_j averaged over temporal increment δt.

The approximation of surface area S begins with specifying fluxes $F_j^{\perp}$:

$$F_j^{\perp} = 0.5(v_j^{\perp} + |v_j^{\perp}|)\Psi_i^n + 0.5(v_j^{\perp} - |v_j^{\perp}|)\Psi_j^n, \tag{2}$$

where the advective normal velocity $v_j^{\perp}$ is evaluated at the face S_j, and assumes the following form:

$$v_j^{\perp} = \boldsymbol{S_j} \cdot 0.5[\boldsymbol{v_i} + \boldsymbol{v_j}]. \tag{3}$$

4 GPU Parallelization

The idea of GPU parallelization is based on decomposition of the MPDATA algorithm into blocks. Each block represents a part (submatrix) of all matrices, which are computed by one GPU task. Every task is a sequence of computational kernels which compute different parts of the algorithm.

In our approach, we distinguish the following levels of GPU parallelization:

- MPDATA task decomposition into kernels;
- overlapping of data transfer with computations;
- computations on GPU excecuted by GPU threads (work-items in OpenCL terminology).

Each MPDATA task is decomposed into 15 kernels, based on synchronization points and data dependencies. Each kernel computes a different part of MPDATA, and is configured in individual way considering the following OpenCL parameters:

- number of global work-items;
- number of local work-items;
- number of dimensions of work-group;
- vector size.

These kernels are executed in a FIFO order corresponding to the dependency tree expressing data dependencies between kernels. Fig. 1 shows the data dependency tree of MPDATA.

One of the most important feature of modern GPU architecture is possibility of overlapping data transfers with computations. It can be achieved by the stream processing. In our approach, each stream consists of a sequence of following instructions:

- sending data blocks from host memory to GPU global memory;
- computations performed by kernels;
- receiving data blocks from GPU global memory to host memory.

An example of stream processing on GPU that support overlapping of data transfers with computations is shown in Fig. 2. In the ideal case (with no time

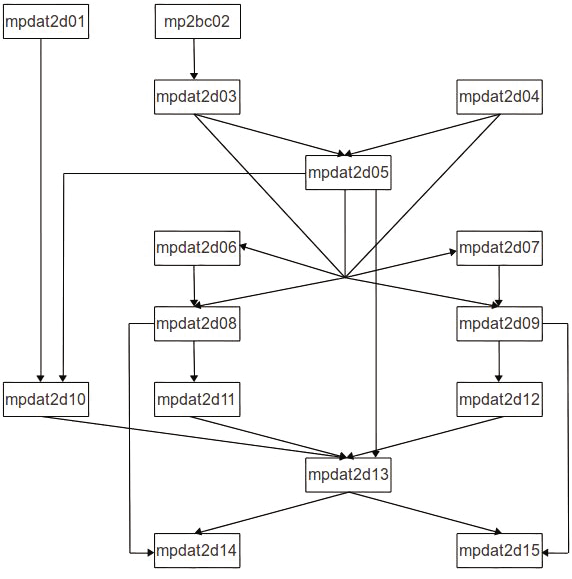

Fig. 1. Data dependency tree of MPDATA for GPU parallelization

delay of communication), we can observe that the more number of streams the more performance gaining can be achieved. Taking into account the time delay of communication, the maximum performance of our parallelization is achieved using four streams. This value was evaluated empirically.

Another level of parallelization are GPU threads called work-items. MPDATA is executed by work-items that are grouped in work-groups. In our approach, we use 1- or 2-dimensional work-groups. One of the biggest challenge here is providing the independence between work-groups because there is no synchronization mechanisms between them.

5 Performance Analysis for GPU Parallelization

The algorithm was tested on the NVIDIA Tesla C1060 card, ATI Radeon HD 5870 and NVIDIA Tesla M2070Q. Table 1 shows the performance results for the linear version of MPDATA with mesh of size 1024x1024, for 100 timesteps. As we can see, 70.3% of data transfer is overlapped with 38.7% of computations on C1060, while only 17.3% of data transfer is overlapped with computations on ATI. The overall time of MPDATA execution is shorter by 25% on ATI than on C1060, and by 29% on M2070Q than on ATI. The kernels time is shorter by 55% on ATI than on C1060, and by 14% on M2070Q than on ATI.

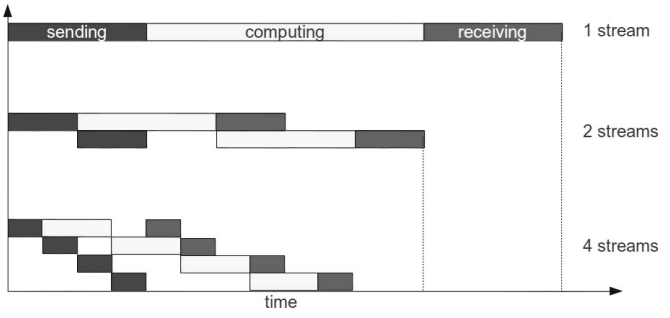

Fig. 2. Overlapping of data transfer with computations

Table 1. Performance results for GPU parallelization of linear version of MPDATA

	NVIDIA Tesla C1060			ATI Radeon HD 5870			NVIDIA Tesla M2070Q		
Streams count	1	2	4	1	2	4	1	2	4
Exec. time [s]	0.505	0.454	0.445	0.354	0.345	0.336	0.292	0.269	0.239
Kernels time [%]	63.6	63.2	64.5	40.8	44.8	69.8	43	43.4	46.5
Comm. time [%]	36.4	38.8	35.5	59.2	55.1	30.2	57	56.6	53.5
Overlapping kern. and comm. [%]	0	13.0	24.9	0	2.6	5.4	0	8.9	19.4
Kernel overlapped [%]	0	21.2	38.7	0	**4.7**	7.7	**0**	**20.3**	**41.6**
Communication overlapped [%]	**0**	**33.5**	**70.3**	**0**	4.2	**17.8**	0	15.6	36.2

6 CPU Parallelization

In this paper, when considering the CPU parallelization we focus on multi-core processing, vectorization and cache reusing. It is necessary to provide the load balancing of computations between available cores. For this aim, the whole problem is divided into six and four equal chunks for AMD and Intel CPUs, respectively. To achieve high efficiency of computations, it is required to apply the SIMD processing and provide a suitable data allocation in the main memory. The SIMD processing is applied manually, using the standard SSE and new AVX extensions. Aligning data to vector lengths is always recommended. When using SSE instructions, data should be aligned to 16 bytes. Similiarly, to achieve best results using Intel AVX instructions on 32-byte vectors, data should be aligned to 32 bytes. Therefore, each row of matrices is aligned to 16 and 32 bytes for SSE and AVX, respectively.

Fig. 3 shows the data dependency tree of MPDATA for the CPU implementation. The linear version of MPDATA corresponds to the first three stages marked

in Fig. 3 with f1, f2, and x', which conventionally are computed as the following sequence of steps:

```
f1: loading data; serial calculations; saving results;
f2: loading data; serial calculations; saving results;
x': loading data; serial calculations; saving results.
```

In order to exploit parallel features of CPU architecture, another approach is considered, which assumes the following steps:

```
f1: data partitioning; loading data; SIMD calc.; saving results;
f2: data partitioning; loading data; SIMD calc.; saving results;
x': data partitioning; loading data; SIMD calc.; saving results.
```

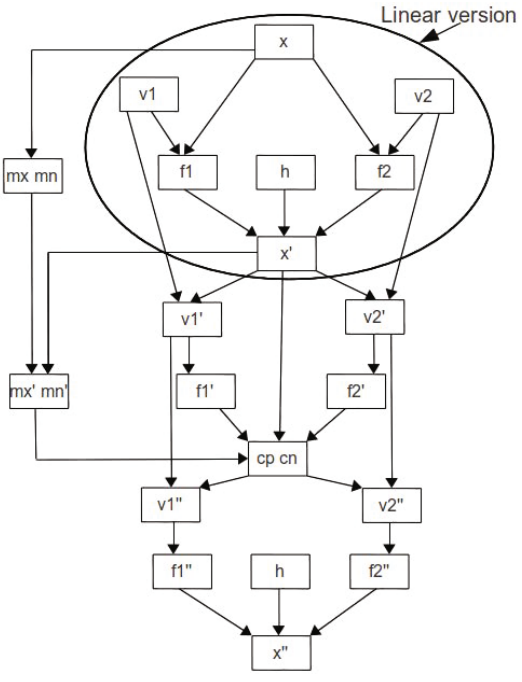

Fig. 3. Data dependency tree of MPDATA for CPU parallelizatoin

7 Performance Analysis for CPU Parallelization Using the Roofline Model

The algorithm was implemented using the OpenMP programming standard, as well as SSE and AVX extensions on Intel Xeon E3-1200 and AMD Phenom II X6. Table 2 shows the performance results of the linear version of MPDATA with mesh of size 5120x5120. As we can see, the speedup is only about 2, even

when using multicore and SIMD processing. In theory, the features of these architectures should allow for achieving maximum speedups of 32 and 24 for Intel and AMD CPUs, respectively. Therefore, we decided to use the Roofline Model [9] to identify bottlenecks of implementing MPDATA on these architectures.

Table 2. Performance results for standard approach of MPDATA

Mesh size 5120x5120	Intel Xeon E3-1200 (4 cores + AVX)		AMD Phenom II X6 (6 cores + SSE)	
	time [s]	speedup	time [s]	speedup
1 core without SIMD	0.16	-	0.21	-
multicore without SIMD	0.077	2.07	0.11	1.9
1 core with SIMD	0.079	2.02	0.16	1.3
multicore with SIMD	0.074	2.16	0.10	2.1

The Roofline Model shows inherent hardware limitations for a given kernel, as well as potential benefit and priority of optimizations. It relates processor performance to memory traffic. The model is based on the operational intensity parameter Q meaning the amount of operations per byte of DRAM traffic (flop/byte). The attainable performance R_a (flop/s) is then upper bounded by both the peak performance R_{max} (flop/s), and the product of the peak memory bandwidth B_{max} (byte/s), and the operational intensity Q:

$$R_a = min\{R_{max},\, B_{max} * Q\}\ [flop/s]\,. \tag{4}$$

Fig. 4 presents performance analysis for Intel Xeon E3 1270 CPU using the Roofline Model. For stages f1, f2 and x', the operation intensity can be expressed as:

$$Q = \frac{n \cdot m \cdot 25}{n \cdot l \cdot 4 \cdot 11} = 0.56\ [\frac{op}{byte}], \tag{5}$$

where computing a problem of size $n \times m$ requires $n \cdot m \cdot 25$ operations, and transfer of 11 matrices of size $n \cdot l \cdot 4$ bytes. Consequently, the attainable performance is only $R_a = 0.56\ [\frac{op}{byte}] \cdot 21\ [GB/s] = 11.7\ [Gop/s]$ as compared to 108.8 $[Gop/s]$ of peak performance.

The Roofline Model shows that the memory traffic is bottleneck when implementing MPDATA on multicore CPUs. To alleviate this limitation, we propose to use the loop tiling technique as a way of providing the efficient cache reusing. Thanks to that, partial results will be stored in cache, which reduces the memory bottleneck. This idea is presented in Fig. 5, where the size $nBlockSize \times mBlockSize$ of blocks has to be adjusted to the cache size.

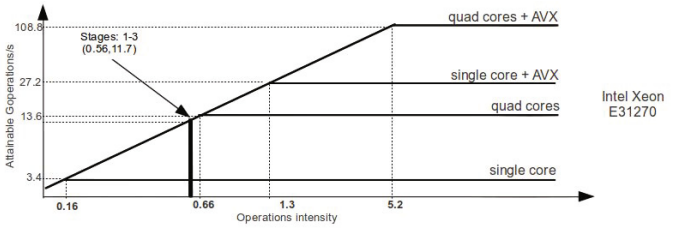

Fig. 4. The Roofline Model for Intel Xeon E3 1270

$$
\begin{aligned}
&for \tfrac{n}{nBlockSize} tiles \quad //i\ dimension \\
&\quad for \tfrac{l}{mBlockSize} tiles \quad //j\ dimension \\
&\qquad MPDATA\_block(...)\ \{ \\
&\qquad\quad loading\ data\ from\ main\ memory\ to\ cache; \\
&\qquad\quad stage\ 1:\ parallel\ computations; \\
&\qquad\quad saving\ partial\ results\ in\ cache; \\
&\qquad\quad stage\ 2:\ parallel\ computations; \\
&\qquad\quad saving\ partial\ results\ in\ cache; \\
&\qquad\quad (...) \\
&\qquad\quad saving\ final\ results\ in\ main\ memory\ for\ each\ block; \\
&\qquad \}
\end{aligned}
$$

Fig. 5. The idea of block version of MPDATA

8 Conclusions and Future Work

The heterogeneous GPGPU computing is a promising approach for increasing performance of numerical simulations of geophysical flows using the EULAG model. Our implementation supports multiple streams processing, which allows for overlapping data transfer with computations, as well as provides a significant reduction in the use of GPU global memory space. The MPDATA task decomposition allows for avoiding dependencies between work-groups. To achieve high efficiency of computations, it is required to apply the SIMD processing using dynamic size of vector.

The vector processing using the AVX extension allows for significant increase of CPU performance. The standard approach does not give a high performance. The obtained speedup is only about 2, even when using multicore and SIMD processing. Therefore, the performance analysis is provided. The Roofline Model shows that the memory traffic is bottleneck for the standard approach to MPDATA implemented on CPU. The loop tiling allows for efficient cache reusing to alleviate memory bottleneck. The block version of MPDATA requires

additional calculations for each block, however, this overhead can be reduced by storing partial results in cache.

Our parallelization of MPDATA is still under development. One of leading approaches is using the autotuning technique which allows for algorithm self-adapting to properties of a system architecture. The final result of our work will be adaptation of MPDATA to hybrid architectures with CPUs and GPUs. In this case, the first challenge is to provide high performance for each system's hybrid component, taking into account their properties. The second challenge concerns data partitioning and load balancing across heterogeneous resources.

Acknowledgments. This work was supported in part by the Polish Ministry of Science and Higher Education under Grant Nos. 648/N-COST/2010/0 COST IC0805 and BS/PB-1-112-3030/2011/S.

We gratefully acknowledge the help and support provided by Jamie Wilcox from Intel EMEA Technical Marketing HPC Lab. The authors are grateful to Krzysztof Luka from AMD for granting access to ATI Radeon HD 5870 GPU.

References

1. AMD Corporation: AMD Phenom II X6 Feature Summary, `http://www.amd.com/`
2. AMD Corporation: ATI Radeon HD 5870 Feature Summary, `http://www.amd.com/`
3. Smolarkiewicz, P.K.: Multidimensional positive definite advection transport algorithm: an overview. Int. J. Numer. Meth. Fluids 50, 1123–1144 (2006)
4. Kurowski, K., Kulczewski, M., Dobski, M.: Parallel and GPU based strategies for selected CFD and climate modeling models. Information Technologies in Environmental Engineering 3, 735–747 (2011)
5. Eulag Research Model for Geophysical Flows, `http://www.eulag.com/`
6. Lindholm, E., Nickolls, J., Oberman, S., Montrym, J.: NVIDIA Tesla: A Unified Graphics and Computing Architecture. IEEE Micro 28, 39–55 (2008)
7. Smolarkiewicz, P., Szmelter, J.: MPDATA: An edge-based unstructured-grid formulation. ELSEVIER Journal of Computational Physics 206, 624–649 (2005)
8. Gepner, P., Gamayunov, V., Fraser, D.L.: Early performance evaluation of AVX for HPC. ELSEVIER Procedia Computer Science 4, 452–460 (2011)
9. Williams, S., et al.: Roofline: an insightful visual performance model for multicore architectures. Communications of the ACM 52, 65–76 (2009)
10. Wyrzykowski, R., Rojek, K., Szustak, Ł.: Using Blue Gene/P and GPUs to Accelerate Computations in the EULAG Model. In: Lirkov, I. (ed.) LSSC 2011. LNCS, vol. 7116, pp. 670–677. Springer, Heidelberg (2012)