Towards Efficient Decomposition and Parallelization of MPDATA on Hybrid CPU-GPU Cluster

Roman Wyrzykowski, Lukasz Szustak(✉), Krzysztof Rojek, and Adam Tomas

Czestochowa University of Technology, Dabrowskiego 73, 42-201 Czestochowa, Poland
{roman,lszustak,krojek,atomas}@icis.pcz.pl

Abstract. EULAG (Eulerian/semi-Lagrangian fluid solver) is an established computational model for simulating thermo-fluid flows across a wide range of scales and physical scenarios. The multidimensional positive definite advection transport algorithm (MPDATA) is among the most time-consuming components of EULAG.

New supercomputing architectures based on multi- and many-core processors, such as hybrid CPU-GPU platforms, offer notable advantages over traditional supercomputers. In our previous works we considered adaptation of 2-dimensional (2D) MPDATA computations to a single CPU-GPU node. The main goal of this paper is to study tenets of optimal parallel formulation of 3D MPDATA on heterogeneous CPU-GPU cluster. Such supercomputer architecture requires not only a different philosophy of memory management than traditional massively parallel supercomputers, but also a comprehensive look at load balancing in the heterogeneous co-processing computing model.

In this paper we propose an approach to implementation of 3D MPDATA algorithm on hybrid CPU-GPU cluster, using a mixture of MPI, OpenMP, and CUDA programming standards. This approach focuses on the donor-cell numerical scheme, and is based on a hierarchical decomposition including level of cluster, as well as distribution of computations between CPU and GPU components of each node, and within CPU and GPU devices. We discuss preliminary performance results for the proposed approach running on a single cluster node consisting of two AMD Opteron Interlagos CPUs and one or two NVIDIA Fermi GPUs.

1 Introduction

The Multidimensional Positive Definite Advection Transport Algorithm (MPDATA) is among the most time-consuming calculations of the EULAG model [8]. In our previous works [7,11] we proposed two decompositions of 2D MPDATA computations, which provide adaptation to CPU and GPU architectures separately. The achieved performance results showed the possibility of achieving high performance both on CPU and GPU platforms.

In this paper, we develop a hybrid CPU-GPU version of 2D MPDATA, to fully utilize all the available computing resources by spreading computations

I. Lirkov et al. (Eds.): LSSC 2013, LNCS 8353, pp. 457–464, 2014.
DOI: 10.1007/978-3-662-43880-0_52,

across the entire machine. When adapting MPDATA to modern hybrid architectures, consisting of GPU and CPU components, the main challenge is to provide high performance for each component, taking into account their properties, as well as efficient cooperation.

The proposed approach to parallelization of the 2D MPDATA algorithm is the starting point for the implementation of 3D MPDATA on hybrid CPU-GPU clusters. We propose a hierarchical decomposition including the level of cluster, as well as distribution of computations between CPU and GPU components of each node, and within CPU and GPU devices. Hybrid clusters offer a fast solution, but understanding the parallel trade-offs is crucial for providing efficiency of computations. These architectures allow for creating many thousands of threads, which has a significant influence on performance of parallel codes [3].

2 Architecture Overview

In our research we use the Cane cluster located at the Poznan Supercomputing and Networking Center, Poland [1]. This machine includes 227 nodes, connected with each other by the InfiniBand QDR network. Each node consists of two AMD Opteron 6234 CPUs (codenamed Interlagos) and one or two NVIDIA Tesla M2050 GPUs, as well as 64 GB of the main memory. The architecture of a single CPU-GPU node is shown in Fig. 1.

Each of AMD Opteron 6234 CPUs [1] includes two dies, each containing 6 cores and 8 MB of L3 cache. All dies are connected by AMD HyperTransport links. For the clock frequency of 2.4 GHz, the peak performance of these two CPUs is respectively 460.8 GFlop/s and 230.4 GFlop/s in a single and double precision.

The NVIDIA Tesla M2050 GPU [5] is based on the Fermi architecture, and includes 14 streaming multiprocessors, each consisting of 32 CUDA cores with 48 KB of shared memory and 16 KB of L1 cache. It gives a total number of 448 available CUDA cores with the clock rate of 1147 MHz. It provides the peak performance of 1.03 TFlop/s and 512 GFlop/s in a single and double precision, respectively. This graphics accelerator card includes 3 GB of global memory with the peak bandwidth of 148.4 GB/s. All the accesses to the global memory go through the L2 cache of size 512 KB.

3 Introduction to MPDATA Algorithm

The MPDATA algorithm belongs to the group of nonoscillatory forward in time algorithms [8]. The 2D MPDATA is based on the first-order-accurate advection equation:

$$\frac{\partial \Psi}{\partial t} = -\frac{\partial}{\partial x}(u\Psi) - \frac{\partial}{\partial y}(v\Psi), \tag{1}$$

where x and y are space coordinates, t is time, $u, v = const$ are flow velocities, and Ψ is a nonnegative scalar field. Equation (1) is approximated according to

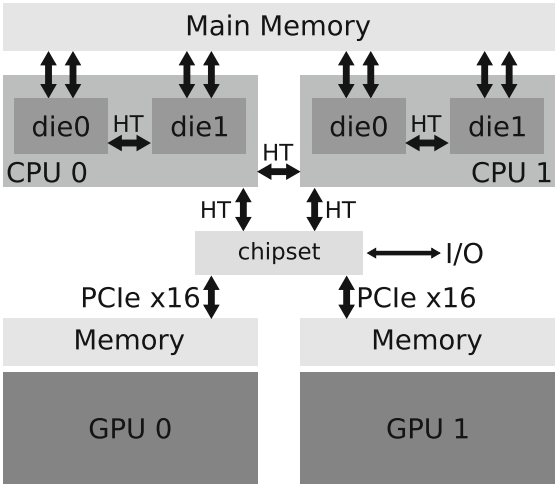

Fig. 1. Architecture of hybrid CPU-GPU node

the donor-cell scheme, which for the $(n+1)$-th time step $(n = 0, 1, 2, \ldots)$ gives the following equation:

$$\begin{aligned}\Psi^*_{i,j} = \Psi^n_{i,j} &- [F(\Psi^n_{i,j}, \Psi^n_{i+1,j}, U_{i+1/2,j}) - F(\Psi^n_{i-1,j}, \Psi^n_{i,j}, U_{i-1/2,j})] \\ &- [F(\Psi^n_{i,j}, \Psi^n_{i,j+1}, V_{i,j+1/2}) - F(\Psi^n_{i,j-1}, \Psi^n_{i,j}, V_{i,j-1/2})].\end{aligned} \tag{2}$$

Here the function F is defined in terms of the local Courant number U:

$$F(\Psi_L, \Psi_R, U) \equiv [U]^+ \Psi_L + [U]^- \Psi_R, \tag{3}$$

$$U \equiv \frac{u\delta t}{\delta x};\ [U]^+ \equiv 0,5(U + |U|);\ [U]^- \equiv 0,5(U - |U|). \tag{4}$$

The same definition is true for the local Courant number V.

The first-order-accurate advection equation can be approximated to the second-order in δx, δy and δt, defining the advection-diffusion equation:

$$\begin{aligned}\frac{\partial \Psi}{\partial t} = &-\frac{\partial}{\partial x}(u\Psi) + \frac{(\delta x)^2}{2\delta t}(|U| - U^2)\frac{\partial^2 \Psi}{\partial x^2} \\ &- \frac{\partial}{\partial y}(v\Psi) + \frac{(\delta y)^2}{2\delta t}(|V| - V^2)\frac{\partial^2 \Psi}{\partial y^2} \\ &- \frac{UV\delta x\delta y}{\delta t}\frac{\partial^2 \Psi}{\partial x \partial y}.\end{aligned} \tag{5}$$

The antidiffusive pseudo velocities $\tilde{u}$ and $\tilde{v}$ in respectively x and y directions are defined according to the following equations:

$$\tilde{u} = \frac{(\delta x)^2}{2\delta t}(|U| - U^2)\frac{1}{\Psi}\frac{\partial \Psi}{\partial x} - \frac{UV\delta x\delta y}{2\delta t}\frac{1}{\Psi}\frac{\partial \Psi}{\partial y}, \tag{6}$$

$$\tilde{v} = \frac{(\delta y)^2}{2\delta t}(|V| - V^2)\frac{1}{\Psi}\frac{\partial \Psi}{\partial y} - \frac{UV\delta x\delta y}{2\delta t}\frac{1}{\Psi}\frac{\partial \Psi}{\partial x}. \tag{7}$$

Therefore, in order to compensate the first-order error of Eq. (1), once again the donor-cell scheme is used but with the antidiffusive velocity $\tilde{u} = -u_d$ in place of u, and with the value of Ψ^* already updated in Eq. (2) in place of Ψ^n. It allows us to compute values of Ψ for the $(n+1)$-th time step.

4 2D MPDATA Decomposition

In this section, we shortly present adaptation of the 2D MPDATA algorithm to the hybrid CPU-GPU architecture, providing trade-off between communication and computation within its components. This approach is based on the efficient use of a single node.

The MPDATA algorithm is based on updating each point of the grid with values from neighboring grid points. Typically the neighborhood structure is fixed, in which case it is called a stencil [2,9]. Our previous research show that MPDATA is a memory-bound algorithm [7,12].

The main task here is the decomposition of MPDATA grid into CPU and GPU domains. We propose the basic strategy of grid partitioning, that assigns two stripes of grid rows to CPU and GPU. Data transfers between CPU and GPU domains are minimized by providing extra computations within both domains. Therefore, the CPU has to compute more rows, because some rows, which originally were assigned to the GPU domain only, are now duplicated in the CPU domain, and vice versa. This approach allows us to avoid communication between CPU and GPU domains within each time step of the MPTADA algorithm, since CPU and GPU components compute their domains separately. As is shown in Fig. 2, the CPU-GPU cooperation, including communication and synchronization, is required only after each time step.

When adapting MPDATA to the hybrid CPU-GPU architecture, the next task is to provide efficient performance for each component. Hence, two different adaptations of the MPDATA algorithm to CPU and GPU processors are required. Each of these adaptations takes into account constraints for the memory bandwidth.

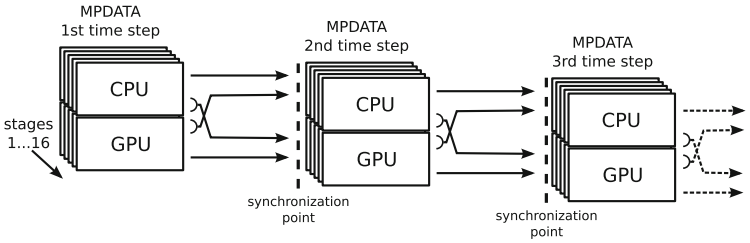

Fig. 2. Scheme of cooperation between CPU and GPU components running the MPDATA algorithm

For CPU, this goal can be achieved by taking advantage of cache memory reusing, as high as possible. This requires to apply an appropriate block decomposition strategy, when the intermediate results of computations for a single block are placed in the cache memory. Only the final results are returned to the main memory. Such an approach is commonly called the temporal blocking [4,10]. Computations within each block are distributed across available CPU cores, and the SIMD processing is applied inside each core. Each AMD Interlagos CPU contains groups of cores (or dies) connected each other by AMD HyperTransport links [1]. Dies have direct access to their own cache memory, and indirect access to caches of other dies. To eliminate inter-cache communications among dies, at the cost of extra computations, we use exactly the same grid decomposition as in the case of adaptation of MPDATA to CPU-GPU architecture. Another advantage of this approach is possibility to apply the NUMA "first-touch" policy.

For GPUs, their global memory allows us to decrease the intensity of access to the main memory, since results of GPU computations performed within a single time step can be stored in GPU only. As a result, performance restrictions due to the memory bandwidth saturation can be alleviated, and the high density of computing resources is better utilized.

The GPU parallelization of MPDATA is based on three levels of GPU parallel hierarchy: (i) overlapping data transfers between the host memory and GPU global memory with GPU computations; (ii) parallel computations across threads running on GPU cores; (iii) vectorization within a GPU thread. The first level requires to apply an appropriate decomposition of data domain into streams, in order to use the streams processing mechanism. It allows us to alleviate bandwidth constraints of PCIe connection between CPU and GPU. The second level concerns parallel processing of GPU threads, which are assembled into CUDA blocks. The last level allows for increasing the amount of computations within a single GPU thread, and reducing overheads of access to GPU global and shared memories.

5 2D MPDATA Performance Results

Table 1 presents execution times of the 2D MPDATA algorithm for 500 time steps and different sizes of grid, using a single node of the target cluster. The achieved performance results correspond to different configuration including the basic serial version running on a single CPU core without using block decomposition and SIMD vectorization, and parallel versions using configurations with 1CPU, 2CPUs, 1GPU, and 2GPUs, as well as hybrid configuration with 2CPUs and 2GPUs. In all the parallel versions, the block decomposition and SIMD vectorization techniques are applied to speedup MPDATA computations over the basic serial version, which does not use these techniques. The speedup of parallel versions over the basic serial version is shown in Fig. 3. For all the grid sizes, the hybrid version allows us to achieve the highest performance. In particular, for the grid of size 4096×4096, it gives speedup of 93.46 over the serial version.

Table 1. Execution times of 2D MPDATA for 500 time steps

size	serial	1CPU	2CPUs	1GPU	2GPUs	2CPUs+2GPUs
1024×1024	99.24	5.21	2.55	2.81	1.47	1.38
2048×2048	384.68	19.14	9.66	10.83	5.48	4.74
3072×3072	869.91	40.91	20.78	26.12	13.07	9.45
4096×4096	1568.22	74.50	37.81	53.99	22.43	16.78

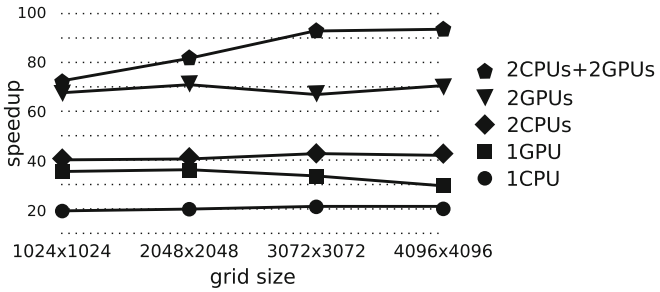

Fig. 3. Speedup of parallel versions over basic serial version

For the largest grid size, the hybrid version is about 2.25 times faster in comparison with using 2CPUs, and about 1.33 times faster than the 2GPUs version.

6 3D MPDATA Decomposition

The achieved performance results show a high perspectives of using the hybrid architecture to the MPDATA algorithm in the 3D case, as well. Following these results, in this section we propose an approach to adaptation of 3D MPDATA to the CPU-GPU cluster, employing both CPU and GPU computing resources. Our approach is based on a hierarchical decomposition including level of cluster, as well as distribution of computations between CPU and GPU components of each node, and within CPU and GPU devices. To take advantage of CPU-GPU cluster, the MPI standard is used across nodes, while OpenMP and CUDA are applied within each node.

This adaptation consists of two basic steps. The first step (Fig. 4a) takes into account the decomposition on the cluster level, and provides data distributions across a 2D mesh of nodes. Each node consists of a group of components, which include CPU and GPU resources. The second step takes into account the data decomposition within a single CPU-GPU node (Fig. 4b). This step is based on the approach previously developed for 2D MPDATA.

The 3D MPDATA algorithm performs simulations determined by size $n \times m \times l$ of the grid. In case of simulations in the EULAG numerical weather prediction [6], the size of grid is usually specified by the following constraints: $n = 2 * m$,

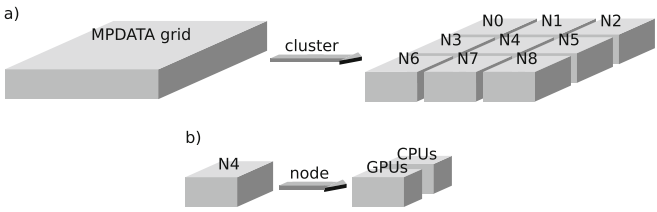

Fig. 4. Grid decomposition of 3D MPDATA onto CPU-GPU cluster

$l \leq 128$, and $n, m >> l$. Such a 3D grid is mapped on a 2D mesh of CPU-GPU nodes of size $r \times c$. As a result, the MPDATA grid is partitioned into subdomains of size $n_p \times m_p \times l$, where each node is responsible for computing within a single subdomain, and:

$$n_p = \frac{n}{r}; \quad m_p = \frac{m}{c}. \tag{8}$$

Every subdomain is further partitioned into two parts, assigned to CPU and GPU resources separately. This partitioning is given by the following equations:

$$S_{GPU} = (G * n_p) \times m_p \times l, \tag{9}$$

$$S_{CPU} = (C * n_p) \times m_p \times l, \tag{10}$$

where parameters G and C characterize GPU and CPU parts, respectively, satisfying the following constraints:

$$G + C = 1; \; G, C \in [0; 1]. \tag{11}$$

Currently, to provide the load balancing between CPU and GPU components, values of G and C parameters are evaluated in an empirical way. However, a dynamic load balancing model will be developed in future work, allowing us to increase the portability of MPDATA code across a variety of hybrid clusters.

7 Conclusions and Further Work

New strategies for memory and computing resources management allow us to ease memory bounds, and better exploit the theoretical floating point efficiency of hybrid architectures. The hybrid computing is a promising approach for increasing performance of numerical simulations of geophysical flows using the EULAG model.

We propose the basic strategy of partitioning the MPDATA grid, that assigns two stripes of grid rows to CPU and GPU components. Thus, data transfers between CPU and GPU domains are minimized by providing extra computations within both domains. Moreover, two separate adaptations of MPDATA algorithm to CPU-GPU hybrid architecture are required, to better utilize features of hybrid architectures.

For 2D grids, the hybrid version gives the best results for all the grid sizes, providing speedup of 93.46 over the serial version of the MPDATA algorithm. The achieved performance of 2D MPDATA gives a high perspectives of using the hybrid programming model to the 3D MPDATA case, as well.

Our parallelization of the EULAG model is still under development. The future work will focus on investigation of MPDATA parallelization based on the proposed 3D grid decomposition. Apart from GPU architectures, the particular attention will be paid to other accelerators such as Intel Xeon Phi.

Acknowledgments. This work was supported by the Polish National Science Centre under grant no. UMO-2011/03/B/ST6/03500.

References

1. AMD and GPGPU cluster, https://hpc.man.poznan.pl/modules/resourcesection/item.php?itemid=61
2. Datta, K., Kamil, S., Williams, S., Oliker, L., Shalf, J., Yelick, K.: Optimization and performance modeling of stencil computations on modern microprocessors. SIAM Rev. **51**(1), 129–159 (2009)
3. Kurzak, J., Bader, D., Dongarra, J.: Scientific Computing with Multicore and Accelerators. Chapman & Hall/CRC , Boca Raton (2010). (Chapman & Hall/CRC Computer and Information Science Series)
4. Nguyen, A., Satish, N., Chhugani, J., Changkyu, K., Dubey, P.: 3.5-D blocking optimization for stencil computations on modern CPUs and GPUs. In: Proceedings of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis, pp. 1–13 (2010)
5. NVIDIA Best Practices Guide, http://developer.nvidia.com/nvidia-gpu-computing-documentation
6. Piotrowski, Z., Wyszogrodzki, A., Smolarkiewicz, P.: Towards petascale simulation of atmospheric circulations with soundproof equations. Acta Geophys. **59**, 1294–1311 (2011)
7. Rojek, K., Szustak, L.: Parallelization of EULAG model on multicore architectures with GPU accelerators. In: Wyrzykowski, R., Dongarra, J., Karczewski, K., Waśniewski, J. (eds.) PPAM 2011, Part II. LNCS, vol. 7204, pp. 391–400. Springer, Heidelberg (2012)
8. Smolarkiewicz, P.: Multidimensional positive definite advection transport algorithm: an overview. Int. J. Numer. Meth. Fluids **50**, 1123–1144 (2006)
9. Venkatasubramanian, S., Vuduc, R.: Tuned and wildly asynchronous stencil kernels for hybrid CPU/GPU systems. In: ICS, pp. 244–255 (2009)
10. Wittmann, M., Hager, G., Treibig, J., Wellein, G.: Leveraging shared caches for parallel temporal blocking of stencil codes on multicore processors and clusters. Parallel Process. Lett. **20**(4), 359–376 (2010)
11. Wyrzykowski, R., Rojek, K., Szustak, L.: Model-driven adaptation of double-precision matrix multiplication to the cell processor architecture. Parallel Comput. **38**, 260–276 (2012)
12. Wyrzykowski, R., Rojek, K., Szustak, Ł.: Using blue gene/P and GPUs to accelerate computations in the EULAG model. In: Lirkov, I., Margenov, S., Waśniewski, J. (eds.) LSSC 2011. LNCS, vol. 7116, pp. 670–677. Springer, Heidelberg (2012)