Case Study on Multi-domain Decomposition of k-Wave Simulation Framework

Filip Vaverka
1st year, full-time study
Supervisor: Jiri Jaros

Brno University of Technology, Faculty of Information Technology
Božetěchova 2, 612 66 Brno, Czech Republic
ivaverka@fit.vutbr.cz

***Abstract*—This article describes both the advantages and challenges of using GPU equipped clusters to implement efficient distributed algorithms. Our main focus is the decomposition of numerical simulations based on (pseudo-)spectral methods. We use a pseudo-spectral simulation of the ultrasound wave propagation in homogenous medium (k-Wave) as a showcase of the distributed algorithm. We show we are able to achieve a three fold speedup of the simulation while lowering the cost at the same time by employing a Local Fourier Basis decomposition.**

***Keywords*—Pseudo-spectral Simulation, Local Fourier Basis, GPGPU, CUDA, GPU Cluster**

I. Introduction

This article describes our plan to approach the design of distributed algorithms utilizing accelerator (such as GPU, MIC, APU or even FPGA) equipped clusters. We can see that one of main roadblocks in pursuit of higher performance of our super-computer systems is the efficiency of compute nodes and inter-node communication bandwidth. The first of those problems is usually tackled in two ways: we employ more efficient processing elements (to this date GPUs can achieve up to 8x FLOPs/W more than general purpose CPUs) and we design efficient algorithms on these processing elements.

The second problem (which is actually more painful) is the inter-node communication, which is usually several times slower than the intra-node communication. Even worse, the inter-node communication is evolving rather slowly and the discrepancy between the performance of the node and the interconnect bandwidth is growing. On the top of that, there is a plenty of tools for optimizing computation itself (compilers, profilers, etc.), yet there is significantly fewer tools to optimize communications and data locality. In distributed memory environments, we have to manually optimize the communication or employ shared memory emulation [4], which is usually unaware of the algorithm running on top of it.

The following sections briefly touch on current and near future architectures of accelerator equipped clusters (mainly GPU accelerated). After that, we show a case study of the multi-domain decomposition of the k-Wave simulation framework, and finally, roughly outline possible directions of following research.

II. Modern Accelerated Cluster Architectures

The basic building block for the current cluster architectures are multi-core CPU based nodes. Each of these nodes then has a few hundreds GBs of a coherent memory (usually NUMA architecture) and a fast network adapter. We can call such a cluster "flat" as it has mostly (ignoring CPU caches) single, flat and coherent memory space.

A simple way to build an accelerated cluster is to add an accelerator (GPU, MIC, FPGA, ...) to each node (usually connected through the PCI-E bus). As both PCI-E and the main memory are rather slow, accelerators usually have their own on-board, high bandwidth, memory with a limited size. Figure 1 shows the architecture of such a cluster (using dual socket Intel Xeon E7 CPU and R9 Fury X GPU). The on-board memory is usually not coherent with the main memory, therefore, the node memory space becomes hierarchical (with manual management). This hierarchical structure brings new challenges to the efficient use of such architectures.

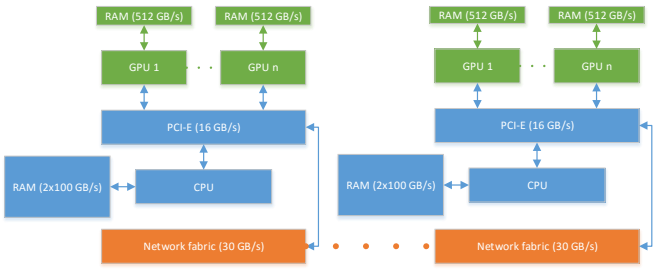


Fig. 1: GPU Accelerated Cluster

In future, we expect an increase in the bandwidth of the intra-node interconnect (NVlink and AMD GMI) and maybe even fully coherent memory, yet these improvements will probably lag behind the increase of the compute power of accelerators. There are also efforts to achieve tighter integration of accelerators with CPUs (NVlink + Power8 [9] and APUs [8]). All this may result in effortless memory coherency of the whole accelerated node, yet performance-wise we expect the hierarchical structure to remain present (in similar way as in current NUMA architectures).

III. Pseudo-spectral Simulation Approach

To show challenges associated with using such clusters we chose a GPU based pseudo-spectral ultrasound wave propagation solver as a case study of an algorithm with non-trivial decomposition.

Computationally, the most intensive part of the solver is the evaluation of the spatial derivatives which is usually calculated, by either the Finite Difference Methods (FDM [10]) or (in our case) pseudo-spectral methods [11]. The pseudo-spectral solver was chosen (a) because it's a widely spread solution and (b) its decomposition is non-trivial (compared to FDM). Big advantage of spectral methods is high accuracy (due to their ability to achieve exponential convergence).

The most often used pseudo-spectral methods are based on the Fourier transform and the derivative can be computed in Fourier basis space (i.e. $f'(x) = \mathbb{F}^{-1}\{ik\mathbb{F}\{f(x)\}\}$), where k is a matrix of wave numbers and i is the imaginary unit). The primary advantage of this approach is that we know derivatives of the basis functions exactly and therefore we are getting an advantage in precision. According to [1], our simulation needs up to an order of magnitude fewer grid points (in 1D case) to achieve accuracy on par with FDM. The secondary advantage is that the Fourier transform can be computed in $O(nlog(n))$ time by the Fast Fourier Transform.

A. Local Fourier Basis

One of a limited number of approaches to alleviate the all-to-all communication burden caused by the k-dimensional FFT computation is the use of the Local Fourier Basis (LFB) decomposition of the simulation domain (so called Multi-domain Decomposition). This method is based on subdivision of the original domain into a number of independent sub-domains. If neighboring sub-domains have a certain overlap and each of them form an LFB (i.e. local functions are smooth and periodic) then the derivatives computed locally approximate the global ones. Figure 2 shows such a decomposition of the simulation domain in 3D.

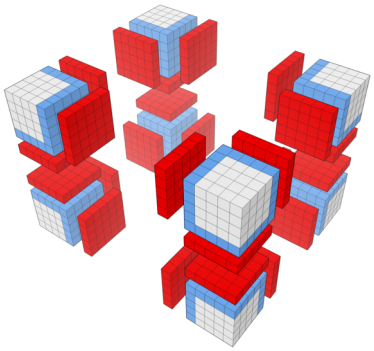


Fig. 2: Multi-domain decomposition

In order to form an LFB after splitting the original domain, we adopted a method described in [2]. Smoothing is done by multiplying a function with a so called bell function (Fig. 3) defined by equations 1 and 2.

$$\theta(x) = \sqrt{\frac{\pi}{2\epsilon}\left[1 + \mathrm{erf}\left(\frac{x\sqrt{\epsilon}}{\sqrt{2}}\right)\right]} \tag{1}$$

$$B(x) = \begin{cases} \theta(x) & \text{if } x \in \langle a_i - \epsilon, a_i) \\ 1 & \text{if } x \in \langle a_i, a_{i+1}) \\ \theta(\epsilon - x) & \text{if } x \in \langle a_{i+1}, a_{i+1} + \epsilon) \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

Where ϵ is the depth of the overlap and $\langle a_i, a_{i+1}\rangle$ is the local core interval of the function.

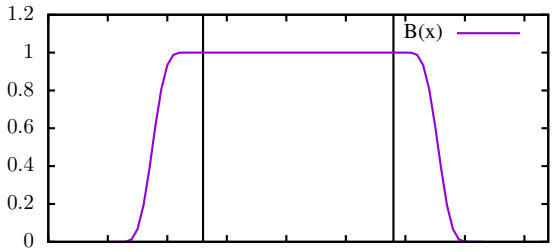
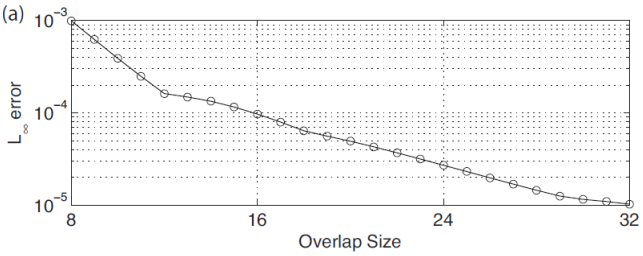
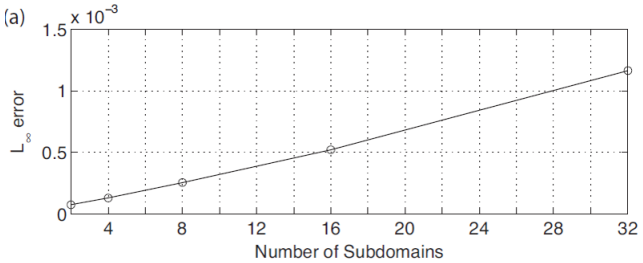


Fig. 3: Bell function

The accuracy of the approximation is given by the number of sub-domains (in a given direction of FFT) and the depth of overlaps. This gives us the opportunity to trade between the overhead (both computation and communication) and the accuracy. The impact of the number of sub-domains and the depth of overlaps on the accuracy shows Fig. 4. Its worth to note that, we can afford error up to 10^{-3}, because that's the accuracy level of the PML which is necessary to avoid periodic behavior of the spectral method.



(a) Depth of Overlaps



(b) Number of sub-domains

Fig. 4: Decomposition error

B. K-Wave Local Decomposition

The proposed k-Wave simulation decomposition was implemented using OpenMPI for inter-process communication, CUDA to access GPUs and parallel HDF5 for I/O. We used cuFFT to compute local gradients on each GPU. The whole implementation can be divided into three parts: Simulation core, communication framework and I/O subsystem.

The simulation core is mostly identical with the global single GPU implementation and consists of a few simple CUDA kernels and cuFFT calls. We have extended the core by CUDA kernels for domain overlap extraction/injection.

The communication framework is responsible for domain decomposition and overlap exchange between neighboring sub-domains. The decomposition is planed ahead and each process loads its part of the domain independently using distributed I/O (results are collected back in the similar way). Overlap exchanges are realized using asynchronous OpenMPI communications and are partially overlapped. Communication overlapping is realized in a way where we extract halos of multiple matrices and start all communications. Here, we only need to wait for all halo zones of the first matrix before computing its gradient.

C. Performance and Scaling

The implementation of the local decomposition was tested on the Emerald cluster which consists of nodes equipped with three or eight NVIDIA C2070 (GF100 - Fermi). Nodes are interconnected by a QDR Infiniband in the half-duplex mode. Both GPUs themselves and the interconnection are therefore rather outdated and we should be (and in fact we are) able to achieve much better results on clusters like Anselm (with very limited number of GPUs). Our experiments were conducted running 100 steps of the simulation (which should be enough to hide any warm-up latencies) with overlaps of 16 grid points. Figure 5 shows the scaling and overhead of our solution on a range of configurations with one to 128 GPUs and domains sizes of $256 \times 256 \times 256$ to $1024 \times 1024 \times 2048$ (2^{24} to 2^{31} points) using decomposition in all three dimensions.

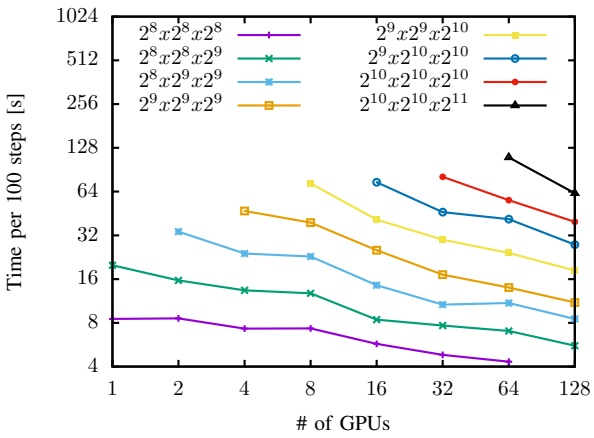


Fig. 5: Scaling on Emerald Cluster

We are able to achieve reasonable strong scaling. The overall efficiency of proposed code increases with the size of the simulation domain. This behavior is expected and can be explained as a result of the decrease in both computational and communication overhead (which grows only with surface area of the sub-domains).

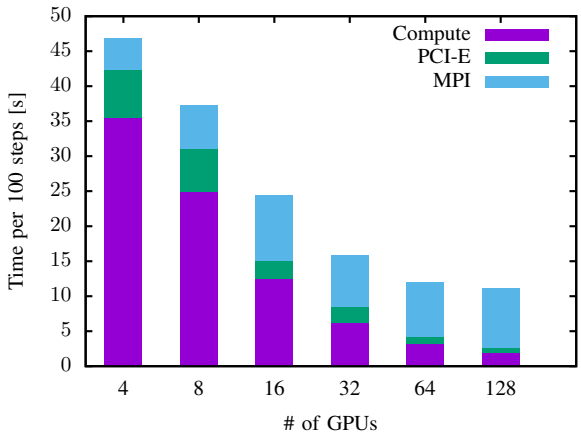


Fig. 6: Overhead on Emerald Cluster

Figure 6 shows the breakdown of the simulation run time for domain size of $512 \times 512 \times 512$ using a 3D decomposition and 16 grid points of overlap. It can be seen that the MPI communication overhead increases until the 3D decomposition is reached ($2 \times 2 \times 4$ GPUs). We can see that the pure computation time doesn't decrease exactly linearly, due to the local domain extensions overhead. The extension overhead reaches exactly 50% at 128 GPUs, which is one of the limits of algorithms strong scaling. Other more prominent limit of the strong scaling is the communication overhead which reaches over 50% of total simulation time at about 16-32 GPUs. Overall the overhead is still significantly lower that that of the global decomposition of FFT [5].

IV. Future Research

The proposed simulation algorithm is apparently a member of the large class of algorithms whose performance tend to be memory and inter-connect bandwidth (and/or latency) bound. Such behavior may not be intrinsic to the algorithm itself, but rather result of its design and implementation. It may be the case that the algorithm behaves like memory/communication bound one due to its inefficient use of the clusters memory hierarchy or poor mapping to the distributed architecture.

In the future we therefore want to focus our effort on the design of efficient decomposition methods (such as proposed local decomposition), communication, memory access patterns and data locality optimizations. We plan to tackle these areas in multiple phases, first we are going to analyze some other simulation algorithms and try to find their common patterns. Secondly we will try to find common solutions of these problems and finally propose ways to automatize these solutions.

At this point we are leaning towards functional representation of the algorithms, which should give us some advantages for their automatic analysis. There is being done some progress at mapping functional paradigm onto massively parallel hardware [6] and we would like to extend such (or similar) approach to distributed architectures with multiple memory address spaces.

There is also a new development in the area of memory access pattern analysis (for non-uniform memory architectures), which significantly reduces the effort necessary to use multiple GPUs or port GPU kernels to new architectures. In the recent work [7] these qualities are achieved by introducing new representation of the GPU kernels. It should be possible to take similar approach and create such representation of the algorithm from its original functional form.

V. Conclusion

This article outlines some of the current problems in the high performance computing on accelerated clusters with non-uniform memory and multiple memory address spaces. We discussed our work in this area on the pseudo-spectral simulation where we achieved significant improvements in the performance. We also briefly outlined possible directions of our future research in the design of efficient distributed algorithms.

Acknowledgment

This work was supported by the FIT-14-2297 Architectures of Parallel and Embedded Computer Systems project.

References

[1] B. E. Treeby and B. T. Cox, "k-Wave: MATLAB toolbox for the simulation and reconstruction of photoacoustic wave fields," *Journal of biomedical optics*, vol. 15, no. 2, pp. 021 314–021 314, 2010.

[2] M. Israeli, L. Vozovoi, and A. Averbuch, "Spectral multidomain technique with local Fourier basis," *Journal of Scientific Computing*, vol. 8, no. 2, pp. 135–149, 1993.

[3] J. Jaros, A. P. Rendell, and B. E. Treeby, "Full-wave nonlinear ultrasound simulation on distributed clusters with applications in high-intensity focused ultrasound," *International Journal of High Performance Computing Applications*, p. 1094342015581024, 2015.

[4] J. Nieplocha, R. J. Harrison, and R. J. Littlefield, "Global arrays: A nonuniform memory access programming model for high-performance computers," *The Journal of Supercomputing*, vol. 10, no. 2, pp. 169–189, 1996. [Online]. Available: http://link.springer.com/article/10.1007/BF00130708;http://www.emsl.pnl.gov/docs/global/papers/tjs.pdf

[5] A. Gholami, J. Hill, D. Malhotra, and G. Biros, "AccFFT: A library for distributed-memory FFT on CPU and GPU architectures." *CoRR*, vol. abs/1506.07933, 2015. [Online]. Available: http://dblp.uni-trier.de/db/journals/corr/corr1506.html#GholamiHMB15

[6] M. M. Chakravarty, G. Keller, S. Lee, T. L. McDonell, and V. Grover, "Accelerating Haskell array codes with multicore GPUs," in *Proceedings of the sixth workshop on Declarative aspects of multicore programming*. ACM, 2011, pp. 3–14. [Online]. Available: http://dl.acm.org/citation.cfm?id=1926358;http://129.94.242.51/~keller/Papers/acc-cuda.pdf

[7] T. Ben-Nun, E. Levy, A. Barak, and E. Rubin, "Memory access patterns: the missing piece of the multi-GPU puzzle." in *SC*, J. Kern and J. S. Vetter, Eds. ACM, 2015, pp. 19:1–19:12. [Online]. Available: http://dblp.uni-trier.de/db/conf/sc/sc2015.html#Ben-NunLBR15

[8] "The AMD Fast Forward Project," Feb. 2014. [Online]. Available: https://asc.llnl.gov/fastforward/AMD-FF.pdf

[9] "Summit and Sierra Supercomputers: An Inside Look at the U.S. Department of Energy's New Pre-Exascale Systems," Nov. 2014. [Online]. Available: http://www.teratec.eu/actu/calcul/Nvidia_Coral_White_Paper_Final_3_1.pdf

[10] K. Okita, K. Ono, S. Takagi, and Y. Matsumoto, "Development of high intensity focused ultrasound simulator for large-scale computing," *International Journal for Numerical Methods in Fluids*, vol. 65, no. 1-3, pp. 43–66, 2011. [Online]. Available: http://dx.doi.org/10.1002/fld.2470

[11] Jan S. Hesthaven, Sigal Gottlieb, David Gottlieb, *Spectral methods for time-dependent problems*. Cambridge : Cambridge University Press, 2007.