High Performance Computing on Low Power Devices

Vojtěch Nikl

Computer Science and Engineering, 1st year, full-time study
Supervisor: Jiří Jaroš

Faculty of Information Technology, Brno University of Technology
Božetěchova 2, 612 66 Brno

inikl@fit.vutbr.cz

Abstract. Nowadays, the power efficiency of modern processors is becoming more and more important next to the overall performance itself. Many programming tasks and problems do not scale very well with higher number of cores due to being memory or communication bound, therefore it is often not beneficial to use faster chips to achieve better runtimes. In this case, employing slower low power processors or accelerators may be much more efficient, mainly because it is possible to get the same results using much less energy and possibly after the same amount of time, given the algorithm can scale to higher number of cores after applying some adjustments fitting the low power architecture. This paper describes the benefits of using low power chips for building an HPC cluster, the group of algorithms where this approach can be useful, results achieved so far and future plans.

Keywords: HPC, parallelism, low power, processor architecture, supercomputers, k-Wave toolbox, MPI, OpenMP, performance evaluation, numerical methods

1 Introduction

Even though computer processors have come a long way in terms of performance, there are still many tasks and problems which require large amounts of computing power to be successfully solved. For some time, hardware engineers haven’t been purely focusing on raw performance, but the energy consumption has also become a very important factor. Using low powered processors can be much more efficient for certain kinds of algorithms, mainly for memory and communication bound problems. Unfortunately, this is where algorithms’ scalability come into play.

Some algorithms scale very well. For example, *Finite difference* or *Partial differential equation* techniques, used for modeling computational electrodynamics, have constant communication complexity per core for any given size of the domain, because each unit communicates only with its closest neighbors. *Computational fluid dynamics methods* and especially its *Lattice Boltzmann* class also scale very well, although the memory demands are often rather high.

However, there is also the opposite class of algorithms, which scale very poorly. Usually, these algorithms require some sort of global or semi-global communication. Typical example are *Spectral methods* of *Partial differential equations*. These methods work with domains, represented by matrices, where each point of the matrix represents a value of some given attribute. The use of the *fast Fourier transform* is very often involved, which requires one or more so called global all-to-all communications. This results in a very intensive communication overhead especially for bigger domains and the communication dominates over the computation. Employing faster chips brings almost no benefit. This class of algorithms can be more suitable for low power processors, because the computation/communication ratio positively increases and there is more room for overlapping the communication and computation steps, while getting the results using much less energy.

2 Motivation

Today’s supercomputers are usually based on the x86 architecture, specifically the Intel Xeon one. One of many examples is the Anselm cluster[1], located in Ostrava, Czech Republic. It consists of 209 2x8 core Intel Xeon E5-

[1] https://docs.it4i.cz/anselm-cluster-documentation/hardware-overview

2665 2.6GHz nodes, each with atleast 64GB of RAM. Each node requires approximately 230W of energy under full load, while providing about 400 GFlop/s of theoretical performance. Most of that energy is dissipated into heat and therefore it requires very intensive and expensive cooling system. Some systems chose a little bit different approach towards higher power effectiveness. One of them is the Fermi cluster[2], located in the Cineca organization, Bologna, Italy. It consists of 10,240 nodes, each integrating 16 core IBM PowerA2 1.6GHz processor. While the overall performance per node is about half of the Anselm one's, the peak power consumption is only 55W. This results in almost twice as good performance per Watt ratio. *The Green 500 list*[3] provides a ranking of the most energy-efficient supercomputers in the world and Fermi is very close to the top at the 49th place, having over 2 GFlop/s per Watt. The most efficient supercomputer has about 5.3 GFlop/s per Watt.

Searching for even better efficiency, the low power processors used in tablets, smartphones and embedded systems seem to be the best bet. They have come a long way and today's smartphones have up to 8-core processors, which are capable of doing some very intensive tasks, such as recording and playing videos in 4K resolution. These chips are built with an emphasis on low power consumption to extend the battery life as much as possible. For example a "soon to be available" ARM based chip nVidia Tegra X1[4] and its GPU can provide 0.512 TFlop/s in single precision while consuming only about 10W of energy. The fastest nVidia GPU, Titan X[5], provides 6.14 TFlops with 250W of power consumption. The Tegra is more than twice as efficient while providing more than 50 GFlops per Watt!

Unfortunately, this approach has a few downsides. Since the low power processors are less powerful, it is often necessary to employ much more of them to reach the same level of overall performance. As mentioned earlier, the scalability of different algorithms may become a problem. For example if it is required to have 8x more cores in order to reach the same theoretical raw performance, the number of messages sent during one all-to-all communication phase increases by a factor of 64!

Another problem may be the amount of system memory. Fermi has 16GB of RAM per node, which can quickly become a limiting factor when calculating extensive simulations with many domains. The Tegra X1 chip supports only up to 4GB of LPDDR4 RAM, so the problem escalates even more. For example, a realistic ultrasound simulation performed by the k-Wave toolbox, introduced in the next section, typically uses a grid size of 1024^3. The amount of memory required to run this simulation is about 128GB, which means at least 2 nodes of Anselm (32 cores), but at least 8 nodes of Fermi (128 cores) or 32 Tegra chips (256 cores), excluding the memory requirements for an operating system, MPI buffers etc. A simulation with a grid size of 4096^3, which is one of the future goals of k-Wave, requires about 8192GB of RAM, so at least 2048 Tegra chips (16,384 cores) need to be employed. Next to the need of high overall performance, memory demands are very often another reason why it is important to use a higher number of cores in many HPC areas, assuming a given algorithm can even provide scalability this high.

The main goal is to explore the possibility of using such a low power architecture cluster for calculating a subset of specific tasks, which are less suitable for current clusters mentioned above. The main focus will be aimed towards spectral methods and algorithms and the exploration of new extreme scaling techniques. And here a very important question arises. Is it actually worth it, considering all pros and cons, to use these low power processors as the main computational units in a cluster and get a very efficient machine in terms of both initial and running costs, but maybe not so efficient in terms of application scalability and overall performance?

3 The k-Wave project

The k-Wave toolbox [1] is designed to simulate ultrasound wave propagation in soft-tissues and bone, modelled as fluid and elastic media, respectively. The simulation of ultrasound wave propagation through biological tissue has a wide range of practical applications including planning therapeutic ultrasound treatments of various brain disorders such as brain tumours, essential tremor, and Parkinson's disease. The major challenge is to ensure the ultrasound focus is accurately placed at the desired target within the brain because the skull can significantly distort it. Performing accurate ultrasound simulations, however, requires the simulation code to be able to exploit several thousands of processor cores and work with datasets on the order of tens of TB.

In the k-Wave toolbox, the k-space pseudospectral method is used to solve the system of governing equations described in detail by Treeby in [2]. These equations are derived from the mass conservation law, momentum

[2] http://www.hpc.cineca.it/content/ibm-fermi-user-guide
[3] http://www.green500.org/
[4] http://www.nvidia.com/object/tegra-x1-processor.html
[5] http://maxwell.nvidia.com/titan-x

conservation law, and an empirically derived acoustic pressure-density relation that accounts for acoustic nonlinearity, absorption, and heterogeneity in the material properties [2].

The k-space and pseudospectral methods gain their advantage over finite difference methods due to the global nature of the spatial gradient calculations [3]. This permits the use of a much coarser grid for the same level of accuracy. However, the global nature of the gradient calculation, in this case using the 3D fast Fourier transform (FFT), introduces additional challenges for the development of an efficient parallel code. Specifically, the FFT requires a globally synchronizing all-to-all data exchange. This global communication can become a significant bottleneck in the execution of spectral models. Fortunately, considerable effort has already been devoted to the development of distributed memory FFT libraries, such as FFTW [4] or PFFT [5], that show reasonable scalability of up to tens of thousands of processing cores.

A recently introduced hybrid MPI/OpenMP decomposition approach of calculating the FFTs [6] is able to scale almost linearly up to 16,384 cores of the Fermi cluster (see Fig. 1). This is a big asset to k-Wave, which was previously limited by its pure MPI 1D decomposition, which allowed to employ no more than about 2048 cores. This hybrid approach requires one global MPI all-to-all communication, which can be very time demanding, especially for bigger domains. Fig. 2 shows that on typical Intel Xeon Sandy Bridge based clusters, Zapat and Anselm, the communication step takes about 80% of the whole computation.

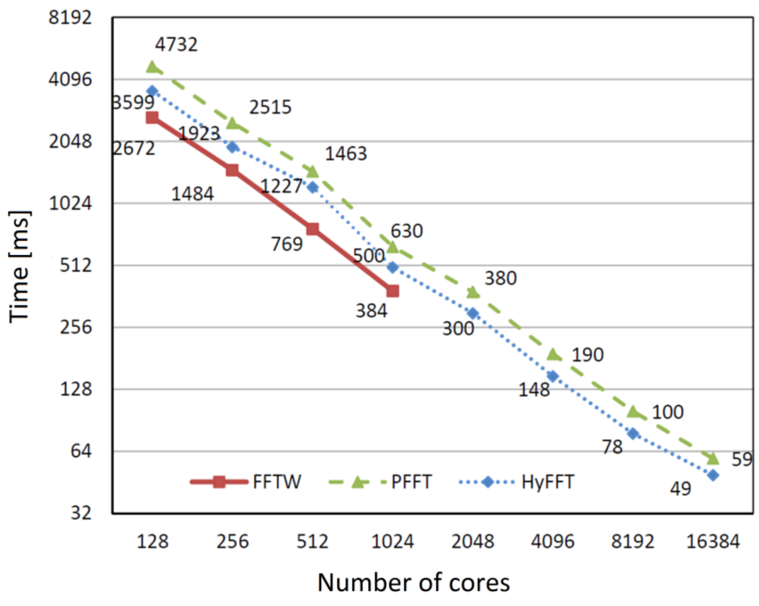

Figure 1: FFT on Fermi, 1024^3 grid points

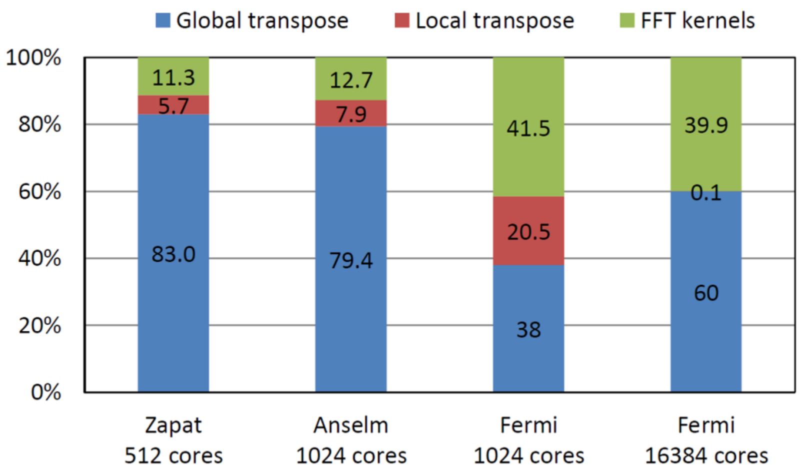

Figure 2: Hybrid FFT Time Distribution, 1024^3 grid points

This is very inefficient in terms of power consumption, because the CPUs are running during the whole time. On Fermi, however, the communication part goes down to about 60% due to slower CPUs, which may allow for better utilization of resources, such as overlapping the communication and computation of 2 or more FFTs running in parallel. This shows that for communication bound algorithms, it is not the performance performance per core that matters the most, but rather a fast interconnecting network and hiding the application latencies and communication overheads.

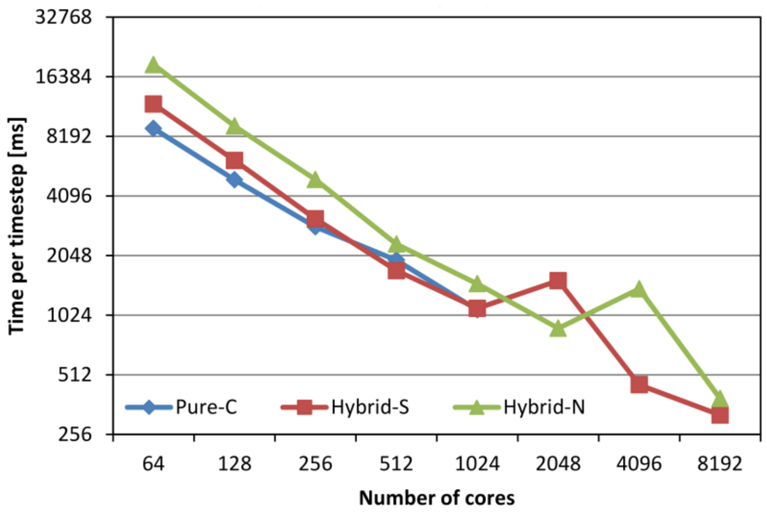

Figure 3: k-Wave on SuperMUC, 1024^3 grid points

A whole k-Wave simulation using the hybrid decomposition was benchmarked on the SuperMUC cluster[6] using up to 8192 cores [7]. Fig. 3 shows that both the Hybrid-S (1 MPI process per socket, 1 thread per core) and Hybrid-N (1 MPI process per node, 1 thread per core) setups offer almost 4 times higher performance over Pure-C (1 MPI process per core), which yields efficacy of almost 50%, which is not so bad considering the code is proven to be communication and memory bound. The peaks in execution time directly correspond to the communication share. In a typical run, the communication share is about 50%, while in those exceptional cases the communication share springs up to 75%. The profile confirmed that the distributed transposition is not done optimally and a custom routine may need to be implemented to ensure the correct behavior.

4 Future plans

During late summer of 2015, a small cluster consisting of 18 Tegra X1 kits (distributed as16 compute nodes, 1 I/O node and 1 login node) will be bought and installed at our faculty. This architecture is able to run on Linux based operating system with most of its standard compilers and libraries used in HPC, such as GNU package, MPI libraries, FFTW, Matlab, CUDA etc. k-Wave has already been successfully compiled and run on the ARM processor of Tegra K1[7], the predecessor of X1, and the next step is to port it to X1 while making efficient use of the heterogeneous CPU+GPU architecture. Previous section showed that the k-space pseudospectral method of k-Wave is a good candidate for low power architectures, mainly because it is very memory and communication bound. The scalability is tested to be good enough to utilize a high number of cores, while the power consumption and heat dissipation of running the whole simulation is expected to be much lower. This means that a single surgery is expected to be much cheaper for the patient, while keeping its planning and simulation time under clinically important 24 hours.

This cluster will serve as a “proof of concept” tool for verifying the (dis)advantages of this architecture over the common architectures on suitable problems. The current task is to benchmark numerical methods for solving partial differential equations, mainly finite differential, finite element, spectral and boundary element methods,

[6] https://www.lrz.de/services/compute/supermuc/systemdescription/
[7] http://www.nvidia.com/object/tegra-k1-processor.html

and compare their performance, accuracy and memory demands on various problems. After this is done on a common cluster, later on the main goal will be to port these algorithms on the tegra cluster and compare the performance, energy demands, efficiency etc.

The goal of the Ph.D. thesis is to define a set of algorithms suitable for low power architectures, port the code, exploit the scalability as much as possible, adjust the properties of the algorithms to fit the architecture and benchmark the performance, power consumption and resource utilization. The conclusion will be: *Algorithm X can run on hardware Y more effectively (with precise specification what that means) under conditions Z.*

5 Conclusion

This paper described the motivation behind using low power architectures for solving specific tasks instead of the common architectures used today. The nVidia Tegra X1 processor was presented as one of the most power efficient architectures, while providing performance comparable to today's desktop PCs. The k-Wave project described above is one of many applications suitable for this architecture due to being memory and communication bound and being able to scale to a very high number of cores at the same time. The presented future work is focusing on the tegra cluster, which is going to verify the benefits and advantages of low power architectures for certain kinds of algorithms. The final goal of the Ph.D. thesis is to identify these sets of algorithms, adjust and benchmark them on both the common and low power architecture and define the conclusion about advantages of low power architectures.

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*, 15(2):021314, 2010.
2. B. E. Treeby, J. Jaros, A. P. Rendell, and B. T. Cox. Modeling nonlinear ultrasound propagation in heterogeneous media with power law absorption using a k-space pseudospectral method. *The Journal of the Acoustical Society of America*, 2012(131):4324–4336, 2012.
3. T. D. Mast, L. P. Souriau, D.-L. D. Liu, M. Tabei, A. I. Nachman, and R. C. Waag. A k-space method for large-scale models of wave propagation in tissue. *IEEE Trans. Ultrason. Ferroelectr. Freq. Control*, 48(2):341–354, 2001.
4. M. Frigo and S. G. Johnson. The Design and Implementation of FFTW3. *Proceedings of the IEEE*, 93(2):216–231, 2005.
5. P. Michael. PFFT-An extension of FFTW to massively parallel architectures. *Society for Industrial and Applied Mathematics*, 35(3):213–236, 2013.
6. V. Nikl and J. Jaros. Parallelisation of the 3D Fast Fourier Transform Using the Hybrid OpenMP/MPI Decomposition. In *Mathematical and Engineering Methods in Computer Science, LNCS 8934*, pages 100–112. Springer International Publishing, 2014.
7. J. Jaros, V. Nikl and B. E. Treeby. Large-scale Ultrasound Simulations Using the Hybrid OpenMP/MPI Decomposition, EASC 2015, Edinburgh (accepted paper).