# Thoughts on Massively-Parallel Heterogeneous Computing for Solving Large Problems

Wen-mei Hwu, Mert Hidayetoğlu, Weng Cho Chew,
Carl Pearson, Simon Garcia, Sitao Huang, and Abdul Dakkak

*Coordinated Science Laboratory, University of Illinois at Urbana-Champaign, Urbana, IL 61801, USA*
w-hwu@illinois.edu

***Abstract*— In this paper, we present our view of massively-parallel heterogeneous computing for solving large scientific problems. We start by observing that computing has been the primary driver of major innovations since the beginning of the 21st century. We argue that this is the fruit of decades of progress in computing methods, technology, and systems. A high-level analysis on out-scaling and up-scaling on large supercomputers is given through a time-domain wave-scattering simulation example. The importance of heterogeneous node architectures for good up-scaling is highlighted. A case for low-complexity algorithms is made for continued scale-out towards exascale systems.**

## I. Introduction

Since the beginning of the 21st century, we have witnessed a major paradigm shift in science and industry innovations. While major innovations in the 20th century were primarily driven by physical instruments such as electromagnetic/light sources, transceivers, and satellites, the high-valued innovations in the 21st century have been primarily driven by computation methods. Examples include computational microscopy, deep-space telescopes, telepresence, and self-driving cars. Tremendous amounts of resources have been invested into innovative applications such as first-principle based models, deep learning, and cognitive computing. Many application domains are questioning the conventional "it is too expensive" thinking that formerly led to modeling inaccuracies and missed opportunities. Today, we are able to incorporate rigorous numerical methods exploiting most of the physical phenomena to improve quality of applications. As a result, the size of computational problems has increased rapidly in order to meet solution fidelity and utility requirements.

## II. Out-Scaling and Up-Scaling

Consider a time-domain wave-scattering simulation: its scaling can be investigated in two ways. Scaling out involves increasing the number of computational nodes, where each node has a given amount of memory and processing power. Spreading the problem among nodes allows accessing more memory, which enables solution of problems with larger sizes and/or finer resolutions with greater number of grid points.

Scaling up improves the single-node computational power and speeds up each time step of a simulation or each step of an iterative solver. This is important since the time steps and iterative steps are inherently sequential due to causality relationships. The saturation of single-core CPU performance in the last decade forced the computing industry to move into the direction of multi-core central processing units (CPUs), many-core processors, and graphics processing units (GPUs). The throughput oriented memory architecture of GPUs provides two to eight times speedup for many applications as compared to multi-core CPUs.

Using a Fourier relationship, the linear time-invariant simulations can be converted into a set of frequency-domain simulations, which are inherently parallel. This is often desirable because of the nice numerical properties of frequency-domain methods, e.g., stability, accuracy, etc. At first glance, it may seem like a good idea to distribute the solutions at different frequencies among the nodes and have a very efficient out-scaling; however, then we immediately run into a load-balancing problem. That is because the low-frequency solutions often require less computation than high-frequency solutions, i.e., the high-frequency problems require finer grids (with more unknowns) than the lower-frequency problems. The load-balancing can be improved with serializing the solutions at lower frequencies and parallelizing those at higher frequencies. Nevertheless, it is a difficult task to achieve high efficiency with this approach.

The scenarios given above highlight the importance of up-scaling for computational problems. Often times, some serialization of the algorithm on each node cannot be avoided, and consequently, the simulation time heavily depends on the per-node computational power.

## III. Fast Algorithms for Large Problems

Today, we have effective computational methods for many high-value applications, except some such as computer vision, natural language dialogue, stock trading, and fraud detection. The advances in computer hardware allow us to use brute-force methods which employ physically sound and mathematically rigorous formulations that do not make any fundamental approximation. This provides numerical algorithms flexibility and applicability, yielding solution of arbitrary problems whose analytical solutions are not available.

Availability of rigorous methods for practical applications is a result of not only the computer power we have today, but also fast (low-complexity) algorithms with $\mathcal{O}(N)$ or $\mathcal{O}(N \log N)$ computational complexities that have been developed in the past 20 years. These algorithms, such as multigrid methods, fast multipole methods, and hierarchical matrix methods provide solutions of many science and engineering problems

978-1-5386-1732-8/17/$31.00 

which were previously thought to be intractable. It would be a mistake to depend solely on computer power and parallelization for solving large problems. Without fast algorithms, even the world's largest supercomputers would never be able to solve even the modest-sized problems that can now be solved with low-complexity algorithms on a powerful workstation.

The mentioned fast algorithms commonly follow iterative, i.e., Krylov subspace, and multilevel approaches, often operating on a tree structure. Speaking of multilevel tree structure, once promising recursive divide-and-conquer algorithms in early 90s are obsolete today due to finite function call stack of the current computer architectures. Today we solve problems with several billion unknowns [1] with loops operating on extremely-large data structures, which requires extra programming care, and with conservative memory allocations to prevent integer overflows and accumulation of round-off errors [2].

## IV. Improving First-Principle-Based Models

Some areas still use first-principle models which make crude approximations because rigorous methods are considered too expensive. Applications with approximations that cause inaccuracies and lost opportunities include medical imaging, earthquake modeling, weather modeling, astrophysics modeling, precision digital manufacturing, and combustion modeling.

Take ultrasound imaging, for example. It is one of the least expensive medical imaging modalities, whose hardware can be found in almost every doctor's office. It uses a simple acoustic phased-array transducer to scan the field of view, and uses specialized hardware to provide real-time brightness maps. Ultrasound imaging is cheap and effective for practical purposes; however, it does not use all the information in the collected field, and therefore the images are not fine enough to provide images for more demanding applications. An inverse scattering approach, however, can employ iterative nonlinear solution methods to reconstruct a quantitative profile of the imaging domain. In each iteration, it is not unusual to solve hundreds of forward scattering problems to be able to update the profile correctly, which means astronomical amount of computation. Consequently, the inverse scattering approach for imaging was considered as impractical before. Recently, there are attempts to change this perception using low-complexity fast algorithms running on large supercomputers with heterogeneous nodes [3], which has the potential to make an impact on medical imaging. This example also highlights the need of multidisciplinary work combining computational and domain-specific expertise to make impact on certain fields.

## V. Next-Generation Computing

Significant emphasis has been placed on the path to exascale computing in the high-performance computing community. This is to support the current and increasing demands for computing power to solve large real-world problems. Cutting-edge supercomputers like NCSA Blue Waters [4] and ORNL Titan [5] are composed of heterogeneous nodes with CPUs and GPUs, which typically allow for a two to eight times whole-application speedup over homogeneous computing nodes. These heterogeneous nodes allow more application tasks to be executed on the appropriate power-efficient hardware, ultimately leading to higher performance.

We expect the next wave of systems, which will be deployed in the 2019 timeframe, to have two or three times more computing throughput than of today's systems. Like the current systems, future systems will undoubtedly consist of general-purpose latency-oriented processors (like CPUs), throughput-oriented processors (like GPUs and many-core processors), and also include reconfigurable hardware (like field-programmable gate arrays), and dense non-volatile memory technologies (like flash disks). Most likely, the future nodes will be "fatter" in the sense that they will involve multiple GPUs and more amount of memory. Effectively and coherently utilizing this fabric of technologies will be a significant challenge moving forward.

The fast algorithms will play a critical role since a doubling or tripling of computing throughput would only be reflected directly to the problems size with low-complexity algorithms. Unfortunately, by their nature, the fast algorithms feature reduced data reusage, and therefore their performance will be limited by the memory bandwidth for most of the algorithms. It will be challenging to increase the data reuse of the fast algorithms, however. New computing architectures that better positions computing devices in the hierarchical memory architecture of heterogeneous nodes will help resolve the memory-bandwidth bottleneck.

## VI. Conclusions

Heterogeneous computing plays a critical role to have good out-scaling and up-scaling efficiencies for solving large scientific problems. We expect the computing throughput will be doubled or tripled in the next two years, and the fast algorithms will obtain the most benefit of this improvement. It will be crucial to use the high-memory throughput of GPUs to obtain maximum benefit.

## Acknowledgment

We acknowledge support of NVIDIA GPU Center of Excellence and NCSA Petascale Application Improvement Discovery Program grants, and NSF grant EECS-1609195.

## References

[1] B. Michiels, J. Fostier, I. Bogaert, D. de Zutter, "Full-wave simulations of electromagnetic scattering problems with billions of unknowns," *IEEE Trans. Antennas Propag.*, vol. 63, no. 2, pp. 796–799, Feb. 2015.

[2] L. Landesa *et al.*, "Successes and frustrations in the solution of large electromagnetic problems in supercomputers," *Applied Computational Electromagnetics Society Symp. (ACES 2017)*, Florence, Italy, Mar. 2017.

[3] M. Hidayetoğlu, C. Pearson, W. C. Chew, L. Gürel, and W.-M. Hwu, "Large inverse-scattering solutions with DBIM on GPU-enabled supercomputers," *Applied Computational Electromagnetics Symp. (ACES 2017)*, Florence, Italy, Mar. 2017.

[4] National Center for Supercomputing Applications, Urbana, IL, USA "Blue Waters," [online] Available: https://bluewaters.ncsa.illinois.edu, Accessed on: May 8, 2017.

[5] Oak Ridge National Laboratory, Oak Ridge, TN, USA Titan, [online] Available: https://www.olcf.ornl.gov/titan/, Accessed on: June 8, 2017.