Eur. Phys. J. Special Topics **210**, 1–4 (2012)

DOI: 10.1140/epjst/e2012-01633-0

THE EUROPEAN PHYSICAL JOURNAL SPECIAL TOPICS

Editorial

Editorial

M. Weigel[1,2,a], A. Arnold[3], and P. Virnau[2]

[1] Applied Mathematics Research Centre, Coventry University, Coventry, CV1 5FB, UK
[2] Institut für Physik, Johannes Gutenberg-Universität Mainz, Staudinger Weg 7, 55099 Mainz, Germany
[3] Institute for Computational Physics, Universität Stuttgart, Pfaffenwaldring 27, 70569 Stuttgart, Germany

Moore's law, predicting a doubling of the computational power of typical hardware about every two years, has successfully described the capabilities of single CPU systems for more than forty years. Most of this time processing speeds were improved by an increase in clock rates and instruction-level parallelism, but over the last years these increases have leveled off and, instead, the total computational capacity of CPUs merely increases through packing more and more processor cores on a single die. While this describes a general trend in the design of computing hardware, due to their originally designated purpose current graphics processing units (GPUs) have already come a longer way down the road of parallelization than current CPUs. It has been true for the last couple of generations of GPUs that their theoretical peak performance, in particular for single-precision floating point operations (FLOP/s), significantly exceeds that of the corresponding x86 based CPUs available at the same time (as of this writing up to around 100 GFLOP/s in CPUs vs. up to 5 TFLOP/s in GPUs). It is therefore natural that scientists and, increasingly, also commercial application programmers, have recently started to investigate the possible value of GPU based computing for extending the reach of numerical simulation methods.

While for many years computer simulations could not compete easily with the more traditional perturbative and field-theoretic methods of approximation commonplace in many fields of the physical sciences from lattice gauge theory to materials science, today the simulational approach has been firmly established as a third pillar of scientific research beside experiment/observation and analytical theory. In fact, a significant number of branches of research would be entirely impossible without computer simulations. The technological advances in available computing hardware, alongside with the improvement of the algorithmic toolbox, have thus attained a pivotal role for the advancement of research in the physical sciences. As a consequence, researchers employing computer simulations have taken a particular interest in the possibility of pushing the limits of computationally tractable problems with the help of emergent technologies. In specific fields, this has always included the design, construction and use of custom-built computing equipment optimized for specific computational tasks. A rather long history of such machines exists in particular in lattice field theory, most recently featuring the QPACE machine based on the Cell processor, and in astrophysical particle simulations, where a series GRAPE of special-purpose computers based on field-programmable gate arrays (FPGAs) has played a significant

[a] e-mail: martin.weigel@coventry.ac.uk

role for about a decade. While such designs typically exhibit substantial performance benefits compared to standard computing hardware, their construction and programming entails a significant financial and time effort which only pays off for a few large-scale problems, where the interests of a sufficiently large number of research groups meet in the same problem. This is where the GPU architecture convinces as an alternative with a relatively large degree of flexibility and low costs through the use of commodity hardware.

The efficiency of calculations on CPU or GPU in terms of FLOP/s per *human* time crucially depends on the availability of easily accessible programming environments for the devices at hand. While in view of a lack of such supporting schemes early attempts of general purpose GPU (GPGPU) calculations still needed to encapsulate the computational entities of interest in OpenGL primitives, the situation changed dramatically with the advent of device-specific intermediate GPGPU language extensions such as ATI Stream and NVIDIA CUDA. More recently, also device independent APIs, most dominantly OpenCL, have become available and promise to reduce the developers' workload when needing to cater for several architectures in parallel. As is clearly showcased by the breadth and sophistication of applications discussed here, these developments have firmly established GPGPU in the field of scientific high-performance computing.

The present issue tries to summarize the current state-of-the-art in the field of computer simulations on GPU and special-purpose computers with a focus on applications in physics. In particular, besides general aspects of high-performance computing, applications in numerical mathematics, soft-matter simulations, simulations of lattice spin models, in lattice field theory as well as in astrophysical gravitational simulations are being discussed.

High-performance computing. As mentioned above, the current success of GPU computing is largely due to the availability of convenient programming languages for the complex process and memory hierarchies of GPUs. Further developments in this direction are discussed in the article *"Porting and scaling OpenACC applications on massively-parallel, GPU-accelerated supercomputers"* of Hart et al. from Cray, Inc., which introduces an even simpler language that will allow to port code to accelerators just by code annotations.

The review article *"An introduction to multi-GPU programming for physicists"* by Bernaschi et al. addresses one of the big complexities that come with using accelerators, namely how to optimally overlap communication and calculation when using multiple GPUs on different computers.

GPUs are not the only hardware platform suitable for massively parallel computing beyond standard compute clusters. This is exemplified by the JANUS collaboration, which has developed a special-purpose computer for the efficient simulation of the notoriously slowly relaxing (discrete) spin-glass models. In their report *"Reconfigurable computing for Monte Carlo simulations: results and prospects of the Janus project"*, Yllanes et al. describe their design based on FPGAs which, for its specific mode of operation, significantly outperforms both GPUs and conventional CPUs.

Numerical mathematics and general aspects. One of the key ingredients in simulational codes are reliable sources of (pseudo) random numbers. For the massively parallel architecture provided by GPUs, potentially millions of independent sources are required which, due to the limited amount of fast cache memory, should have very small states. How a number of classical and newly designed random-number generators fare with respect to these requirements is discussed in the article by Manssen et al., "*Random number generators for massively parallel simulations on GPU*".

Brandes *et al.* discuss in their contribution "*CPU vs. GPU - Performance Comparison for the Gram-Schmidt Algorithm*" in depth how the Gram-Schmidt method

can be implemented optimally both on GPUs *and* CPUs. Efficient implementations on CPU involve complicated blocking structures, which make the CPU code comparable in complexity to the GPU code. On the other hand, such optimized code almost reaches the performance of a GPU even a classical CPU.

Soft-matter simulations. The contributions of Röhm and Arnold, *"Lattice Boltzmann simulations on GPUs with ESPResSo"*, and Kopp and Höfling, *"GPU-accelerated simulation of colloidal suspensions with direct hydrodynamic interactions"*, both deal with using GPUs to accelerate the calculation of hydrodynamic interactions in molecular dynamics, however using very different approaches. The lattice Boltzmann algorithm of Röhm and Arnold uses a lattice fluid to mediate hydrodynamic interactions, while Kopp's and Höfling's approach is based on computing the mobility matrix. The work by Röhm and Arnold is part of the open source ESPResSo package. A conventional molecular dynamics code running on multiple CPUs via MPI is coupled with a hydrodynamics solver on GPU.

The article of Block et al., *"Accelerated GPU Simulation of Compressible Flow by the Discontinuous Evolution Galerkin Method"* again treats flow phenomena, however on the macroscopic scale. The contribution shows how a significant speedup can already be achieved by porting only the evolution operator to GPUs, even if adaptive meshes are used.

Simulations of lattice models. A number of articles discuss in some detail the use of GPUs for the simulation of lattice spin models. Block and Preis in "*Computer Simulations of The Ising Model on Graphics Processing Units*" review in an introductory fashion a number of implementations of Ising model simulations of increasing sophistication. The locality in these models leads to particularly remarkable speed-ups. While these implementations are based on CUDA, Block in "*Platform independent, efficient implementation of the Ising model on parallel acceleration devices*" introduces a device-independent code base for simulations of the Ising model that can be compiled to result in CUDA code as well as in OpenCL (or even OpenMP or MPI) code able to run on a variety of devices.

For the case of spin glasses with continuous spins, where JANUS is not applicable, the contribution *"Optimized GPU simulation of continuous-spin glass models"* by Yavors'kii and Weigel discusses in detail how GPUs can be employed for such simulations, here using a complex multi-replica simulation technique. Also, they discuss the porting of the order parameter calculations, which is a must for a real application, and again shows that the times are over, when GPU codes were often just proofs of concept.

Lattice models for *non-equilibrium* phenomena are discussed by Kelling et al. in "*Comparison of Different Parallel Implementations of the 2+1-Dimensional KPZ Model and the 3-Dimensional KMC Model*". These systems are models for interface growth and thermally activated diffusion, respectively. Using sophisticated schemes of domain decomposition very significant speed-ups can be achieved, bringing simulations at experimental length and time scales into reach. A number of different architectures and APIs, including CUDA and OpenCL are considered.

Applications in QCD. GPU computing has also found rather widespread use in lattice quantum chromodynamics (QCD). In their contribution "*Lattice QCD with overlap fermions on GPUs*", Walk, Wittig and Schömer discuss the use of GPUs in a specific lattice implementation of the Dirac operator known as the Neuberger-Dirac operator which allows them to study the spontaneous breaking of chiral symmetry. The bulk of the necessary computations are problems in linear algebra; using a

careful distribution of the work-load between CPU and GPU, significant performance increases as compared to a purely CPU implementation can be achieved.

Astrophysical simulations. The benefit of using GPUs in large-scale simulations in astrophysics are discussed by Bédorf and Portegies Zwart in their review "*A pilgrimage to gravity on GPUs*". While for a number of years special-purpose computers based on FPGAs were used as state-of-the-art, in recent years GPUs have become the tool of choice for accelerating direct N-body simulation codes as well as the more sophisticated hierarchical tree-based techniques. The latter, in particular, are rather tricky to parallelize due to the complexities of parallel traversals of tree structures.

The present issue on "Computer simulations on GPU" originated in an international symposium of the same name held May 30–June 1, 2011 in Schloss Waldthausen near Mainz, Germany. This event proved to be particularly fruitful due to scientists from a rather wide range of application fields coming together. Discussing the current challenges in a specific field, it was repeatedly found that practitioners in a different field were facing very similar problems and, in some cases, had already solved them, thus leading to unexpected help for solving the problem at hand. With the present issue, we hope to stimulate the exchange of ideas between researchers involved with GPU computing in different areas of the simulational sciences. In particular, we hope that young researchers and those new to the field can benefit from the mixture of reviews and original research articles presented here.

The authors gratefully acknowledge financial support by Volkswagen foundation under contract No. 85 785 for the symposium SimGPU 2011 which laid the foundations for the present collection of articles.