High-Level Programming for Heterogeneous and Hierarchical Parallel Systems

Javier Garcia-Blas[a], Christopher Brown[b]

[a] *Universidad Carlos III de Madrid, Spain*
[b] *University of St Andrews, United Kingdom*

Abstract

High-Level Heterogeneous and Hierarchical Parallel Systems (HLPGPU) aims to bring together researchers and practitioners to present new results and ongoing work on those aspects of high-level programming relevant, or specific to GPGPUs and new architectures. The 2016 HLPGPU symposium was an event co-located with the HiPEAC conference in Prague, Czech Republic. HLPGPU is targeted at high-level parallel techniques, including programming models, libraries and languages, algorithmic skeletons, refactoring tools and techniques for parallel patterns, tools and systems to aid parallel programming, heterogeneous computing, timing analysis and statistical performance models.

Keywords: GPGPU, Heterogeneous, Domain-specific parallel patterns, performance, programming models

1. Presentation

Heterogeneous multicore/manycore systems are becoming ubiquitous. Such systems combine CPUs, GPUs etc. into coherent highly-parallel, and energy-efficient, systems. Data-intensive applications are one of the most important and commonly encountered classes of industrial application. They are often potentially highly parallel and are a clear match to emerging heterogeneous parallel systems. Near-future data-intensive applications will thus need to consider large-scale parallelism as an essential part of their design and development.

Typical state-of-the-art development methodologies treat parallelism as an after-thought, deploying, for example, inappropriate concurrency techniques that are tedious, error-prone and lacking scalability or portability to new computer architectures. Developing parallel software is still seen as a specialist activity, and strong software engineering principles are rarely applied. What is needed is a new software development approach that is simple enough to be followed by applications developers, but which is flexible and robust enough to deal with highly complex parallel hardware and systems, as required for e.g. data-intensive applications.

Email addresses: fjblas@inf.uc3m.es (Javier Garcia-Blas), cmb21@st-andrews.ac.uk (Christopher Brown)

This special issue presents innovative technical solutions showing advances on high-level parallel programming models, adaptation to heterogeneous/hierarchical platforms (i.e., GPGPU, multicore, FPGA), programming experiments and applications of heterogeneous/hierarchical platforms with a view on high-level programming methods, domain-specific parallel patterns, pattern implementations using application-specific, and refactoring tools and techniques for parallel patterns.

The invited talk was given my Daniel Garcia of Universidad Carlos III de Madrid. We had 10 paper presentations at the workshop, after which, we opened up the special issue for revised submissions. This special issue reports the accepted revised papers. The special issue includes extended versions of the selected papers of HLPGPU 2016 workshop whose topics fit in the scope of this special issue, but it has been also open to any author, through an open call.

2. Special Issue Contents

This special issue of High Performance Computing Applications Journal contains papers selected from a set of invited papers extracted from the papers presented in the HLPGPU 2016 workshop, High-Level Programming for Heterogeneous and Hierarchical Parallel Systems, held in Prague, Czech Republic, but it also covers papers coming from an open call. The special issue received 8 papers, 6 featuring extended versions of papers selected among top ranked papers of HLPGPU 2016 and two from the open call. Four of which were selected for publication after going through the High Performance Computing Applications Journal peer review process. As we show below, the accepted papers cover important aspects of the special issue topics, going from the development of parallel patterns to GPGPU-based applications.

Heterogeneous parallel platforms, comprising multiple processing units and architectures, have become a cornerstone in improving the overall performance and energy efficiency of scientific and engineering applications. Nevertheless, taking full advantage of their resources comes along with a variety of difficulties: developers require technical expertise in using different parallel programming frameworks and previous knowledge about the algorithms used underneath by the application. In "Performance Modelling and Verification of Cloud-based Auto-Scaling Policies" [3], Rio et al. alleviate this burden by proposing an adaptive offline implementation selector that allows users to better exploit resources provided by heterogeneous platforms. Specifically, this framework selects, at compile time, the tuple device-implementation that delivers the best performance on a given platform. The user interface of the framework leverages two C++ language features: attributes and concepts. To evaluate the benefits of this framework, the authors analyse the global performance and convergence of the selector using two different use cases. The experimental results demonstrate that the proposed framework allows users enhancing performance while minimizing efforts to tune applications targeted to heterogeneous platforms and also that the proposed framework delivers comparable performance figures with respect to other similar approaches.

Graphics processing units are used to accelerate reverse time migration, but these deployments suffer from limitations such as the lack of high graphics processing unit memory

capacity, frequent CPU-GPU communications that may be bottlenecked by the PCI bus transfer rate, and high power consumptions. Said et al. [5] present the work "Leveraging the accelerated processing units for seismic imaging: A performance and power efficiency comparison against CPUs and GPUs" that explores how efficiently may the APU be applicable to reverse time migration. Using OpenCL (along with MPI and OpenMP), a CPU/APU/GPU comparative study is conducted on a single node for the 3D acoustic reverse time migration, and then extended on up to 16 nodes. The authors show the relevance of overlapping the I/O and MPI communications with the computations for the APU and graphics processing unit clusters, that performance results of APUs range between those of CPUs and those of graphics processing units, and that the APU power efficiency is greater than or equal to the graphics processing unit one.

Structured parallel programming models have been developed to support the design and implementation of parallel applications.These programming models provide the parallel application programmer with a set of pre-defined, ready to use parallel pattern abstractions that may be directly instantiated, alone or in composition, to model the complete parallel behaviour of the application at hand. Examples of these patterns are pipelines, farms, and data stream-based skeletons. Danelutto et al. [2] introduce a set of state access patterns suitable for managing accesses to states in parallel computations operating on streams. The state access patterns are useful for modelling typical stream parallel applications. The authors present a classification of the patterns according to the extent and way in which the state can be structured and accessed, defining precisely the state access patterns and discuss possible implementation schemas, performances and possibilities to manage adaptivity (parallelism degree) in the patterns. The paper includes experimental results relative to implementations built on top of the structured parallel programming framework FastFlow [1] that demonstrate the feasibility and efficiency of the proposed access patterns.

Nowadays the use of hardware accelerators, such as the graphics processing units, is key in solving computationally costly problems that require high performance computing. However, the development of solutions for an efficient deployment for these kind of devices is a very complex task, which relies on the manual management of memory transfers and configuration parameters. The programmer has to carry out a deep study of the particular data that needs to be computed at each moment, across different computing platforms, also considering architectural details. In the paper "Controllers: An abstraction to ease the use of hardware accelerators", Moreton et al. [4] introduce the controller concept as an abstract entity that allows the programmer to easily manage the communications and kernel launching details on hardware accelerators in a transparent way. This model also provides the possibility of defining and launching central processing unit kernels in multi-core processors with the same abstraction and methodology used for the accelerators. It internally combines different native programming models and technologies to exploit the potential of each kind of device. Additionally, the model also allows programmers to simplify the proper selection of values for several configuration parameters that can be selected when a kernel is launched. This is done through a qualitative characterization process of the kernel code to be executed.

Acknowledgements

We would like to thank all the authors, reviewers, and editors involved in the elaboration of this special issue, including also the reviewers that were involved in the HLPGPU 2016 workshop. We are especially grateful to Jack Dongarra, editor in chief of the International Journal of High Performance Computing Applications, for approving this special issue and for his help along the process of its preparation.

References

[1] M. Aldinucci, M. Danelutto, P. Kilpatrick, and M. Torquati. Fastflow: high-level and efficient streaming on multi-core. *Programming multi-core and many-core computing systems, parallel and distributed computing*, 2013.

[2] M. Danelutto, P. Kilpatrick, G. Mencagli, and M. Torquati. State access patterns in stream parallel computations. *The International Journal of High Performance Computing Applications*, 0(0):1094342017694134, 0.

[3] D. del Rio Astorga, M. F. Dolz, L. M. Sanchez, J. Fernndez, and J. D. Garca. An adaptive offline implementation selector for heterogeneous parallel platforms. *The International Journal of High Performance Computing Applications*, 0(0):1094342017698746, 0.

[4] A. Moreton-Fernandez, H. Ortega-Arranz, and A. Gonzalez-Escribano. Controllers: An abstraction to ease the use of hardware accelerators. *The International Journal of High Performance Computing Applications*, 0(0):1094342017702962, 0.

[5] I. Said, P. Fortin, J. Lamotte, and H. Calandra. Leveraging the accelerated processing units for seismic imaging: A performance and power efficiency comparison against cpus and gpus. *The International Journal of High Performance Computing Applications*, 0(0):1094342017696562, 0.

Authors' Biographies

Dr Javier Garcia-Blas is Visiting Professor of the University Carlos III of Madrid since 2015. He received the MS degree in Computer Science in 2007 at the University Carlos III of Madrid. He also received a PhD in Computer Science from University Carlos III in 2010. He has cooperated in several projects with researchers from various high performance research institutions such as HLRS (funded by HPC-Europe program), DKRZ, and Argonne National Laboratory. He is currently involved in various projects in topics such as parallel I/O, cloud computing, heterogeneous computing, and accelerators for high-performance platforms. He has been involved in three research projects funded by the European Union (such as Repara, Rephrase and IC1035 Cost action NESUS). He is currently coordinating the H2020 prject ASPIDE. He has participated in many conference organization committees, including EuroMPI 2013, ICA3PP 2016, IUCC 2016 and CCGRID 2017. He currently count with more than 80 international publications in journal and conference papers.

Dr Christopher Brown graduated with a PhD from the University of Kent in 2009, under the supervision of Professor Simon Thompson. His thesis was on building refactoring tools for functional programming languages (specifically Haskell); and, as such, he built the HaRe tool (the Haskell Refactorer) for refactoring Haskell programs. Since then, Christopher has developed an outstanding and world class publication record. Publishing over 30 papers in multi-disciplines and areas, in ACM and IEEE conferences and journals on Parallel Programming, Functional Programming, Refactoring, Machine Learning and Symbolic Computing.

Christopher is currently a senior research fellow at the University of St Andrews, Scotland, UK, where he has worked on four EU grants: SCIEnce, building parallel algorithmic skeletons for computational algebra applications in GAP and Haskell. Following SCIEnce, Christopher worked on the EU ParaPhrase project, and then on the following EU project, Rephrase, Christopher took the ideas that were fundamental to his PhD and applied them to problem of writing multi-core software. This was pioneering research and during the course of the project he demonstrated that my refactoring technology could be applied generically, across different languages and paradigms, including Haskell, Erlang and C/C++ to enable software developers to fully embrace the new multi-core era. Christopher is currently working on a new EU project, Teamplay, where he is researching into developing provably correct cost certificates for parallel C programs.