The Journal of Supercomputing (2018) 74:5625–5627
https://doi.org/10.1007/s11227-018-2651-8

Reengineering for parallelism in heterogeneous parallel platforms

J. Daniel García[1] · **Kevin Hammond**[2] · **Lutz Schubert**[3]

1 Introduction

In recent years, parallel programming models have evolved dramatically. While, historically, the main focus of research has been on exploiting multi-core/many-core processors with the expectation of an increasing number of cores per chip, the emergence of increasingly heterogeneous computing has changed the landscape. We have moved from a scenario that is mostly dominated by OpenMP [1] at the node level and by MPI [2] at the cluster level towards a new situation where GPUs and other accelerators are starting to have a pervasive presence in the target parallel platforms.

The presence of GPUs in the Top 500 Supercomputer list [3] has also been increasing. In this new scenario, both new and existing applications need to be adapted to deal with different and complex hardware environments. The number of legacy applications that need to be ported to multiple heterogeneous architectures makes it necessary to improve the process of transforming existing applications to new programming models.

Parallel patterns have been in use since the 90s [4]. They have emerged as a way of expressing parallelism in existing sequential applications, providing a way of raising the abstraction level and making it possible to ensure a proper separation of concerns between the application semantics and technical implementation details. Many algorithms match a parallel pattern approach, and patterns are easily exploitable by heterogeneous parallel architectures [5]. With the emergence of heterogeneous platforms, patterns have been shown to be an excellent way to express algorithms that can then been mapped to multiple architectures, so reducing the software development effort.

✉ J. Daniel García
jdgarcia@inf.uc3m.es

1 University Carlos III of Madrid, Leganes, Madrid, Spain

2 University of St. Andrews, St Andrews, UK

3 University of Ulm, Ulm, Germany

Several recent research projects have attempted to provide a holistic approach that combines the need for transforming legacy applications with mechanisms for expressing application semantics in suitable abstractions that can then been mapped efficiently to different heterogeneous architectures. For example, the REPARA and ParaPhrase projects [6, 7] have focused on partitioning applications into components that can be deployed on different heterogeneous devices through software refactoring mechanisms. The recent RePhrase project [8] extended this to take a software engineering approach where automated identification and transformation of patterns are central.

In summary, the hardware landscape has changed dramatically over the last decade from homogeneous multi-processor and multi-core systems to a more diverse environment where different kinds of accelerators comprise a central part of the infrastructure. GPUs have become dominant amongst these accelerators, bringing new programming models as first-class citizens. Within this new research space, there is a need to express application semantics independently from specific programming models as well as to perform transformations on legacy sequential applications.

2 Special issue presentation

This special issue includes five exciting new research contributions. Two of these correspond to extended research contributions from the Repara 2016 workshop that was held in conjunction with the 16th IEEE International Conference on Scalable Computing and Communication, in Toulouse, France, during July 2016. The remaining three contributions were selected from an open call for papers.

In "Exploring the interoperability of remote GPGPU virtualization using rCUDA and directive-based programming models" [9], Castelló et al. study the integration of two directive-based programming models, OmpSs and OpenACC, to make use of remote virtualized CUDA devices through rCUDA. This work moves towards a scenario where applications do not need a full manual rewrite for moving from local GPGPU to the use of remote GPGPUs.

In "MeterPU: a generic measurement abstraction API" [10], Lu and Kessler present a low overhead abstraction layer for taking measurements on time and energy consumption from different hardware components in an heterogeneous platform. They show how this abstraction layer can be used in optimization frameworks, providing examples for autotuned skeleton back-end selection that can be used for optimizing for speed or energy.

In "Data stream processing via code annotations" [11], Danelutto et al. show how code annotations can be used to express stream parallelism. The code can be transformed to a concrete implementation (in this case the FastFlow library) through source-to-source transformation. This approach has been evaluated as a feasible way of performing quick prototyping on different parallelization strategies. The work that has been presented in this paper is an excellent example of how existing streaming applications can be refactored to target new architectures.

In "Assessing and discovering parallelism in C++ code for heterogeneous platforms" [12], Sanchez et al. introduce AKI as an automatic kernel identification and annotation tool aiming to identify potential kernels from C++ sequential applications.

Classifying kernels that can be executed in heterogeneous devices is a relevant step towards better reengineering of software in situations where heterogeneous devices are included in parallel platforms.

Finally, in "A parallel pattern for iterative stencil + reduce" [13], Aldinucci et al. present a new pattern: the iterative stencil-reduce that can be used to simplify a category of data-parallel applications. This pattern can generalize more simple data patterns, both data parallel and stream parallel. The work presented provides a valuable case study of how parallel patterns can be generalized, and also how they can help to express application semantics separated from implementation-specific details.

In summary, the papers included in this special issue are representative of the excellent progress that the research community is making towards better approaches for reengineering software in parallel heterogeneous platforms. They highlight many innovative new techniques, and suggest exciting new research directions that will enable better and easier utilization of heterogeneous parallel hardware in future parallel applications.

References

1. Dagum L, Menon R (1998) OpenMP: an industry standard API for shared-memory programming. IEEE Comput Sci Eng 5(1):46–55
2. Gropp WD, Gropp W, Lusk E, Skjellum A (1999) Using MPI: portable parallel programming with the message-passing interface, vol 1. MIT Press, Cambridge
3. TOP500 Supercomputer sites (2018) www.top500.org. Accessed 10 Sept 2018
4. Gorlatch S, Cole M (2011) Parallel skeletons. In: Padua DA (ed) Encyclopedia of parallel computing. Springer, Berlin, pp 1417–1422
5. McCool M, Reinders J, Robinson A (2012) Structured parallel programming: patterns for efficient computation. Morgan-Kaufmann, Burlington
6. REPARA Project. www.repara-project.es. Accessed 10 Sept 2018
7. ParaPhrase Project. www.paraphrase-ict.eu. Accessed 10 Sept 2018
8. RePhrase Project. https://rephrase-eu.weebly.com/. Accessed 10 Sept 2018
9. Castelló A, Peña AJ, Mayo R, Planas J, Quintana-Ortí E, Balaji P (2016) Exploring the interoperability of remote GPGPU virtualization using rCUDA and directive-based programming models. J Supercomput. https://doi.org/10.1007/s11227-016-1791-y
10. Li L, Kessler C (2016) MeterPU: a generic measurement abstraction API. J Supercomput. https://doi.org/10.1007/s11227-016-1792-x
11. Danelutto M, De Matteis T, Mencagli G, Torquati M (2016) Data stream processing via code annotations. J Supercomput. https://doi.org/10.1007/s11227-016-1793-9
12. de Rio D, Sotomayor R, Sanchez LM, Garcia-Blas J, Calderon A, Fernandez J (2016) Assessing and discovering parallelism in C++ code for heterogeneous platforms. J Supercomput. https://doi.org/10.1007/s11227-016-1794-8
13. Aldinucci M, Danelutto M, Drocco M, Kilpatrick P, Misale C, Peretti G, Torquati M (2016) A parallel pattern for iterative stencil + reduce. J Supercomput. https://doi.org/10.1007/s11227-016-1871-z