# Preface

The first International Parallel Tools Workshop (IPTW) was held on July 9–10, 2007 at HLRS in Stuttgart. The idea was to bring industrial and academic High-Performance Computing (HPC) user communities together with developers of tools to discuss state-of-the-art parallel programming tools and supporting technologies. The mission of the event was twofold: on the one hand increasing users' awareness and understanding of parallel programming tools, and on the other hand providing tool developers with feedback on users' needs and input from other tool developers. The vision behind all of it: tools for parallel programming in High-Performance Computing is the enabler for an important step forward towards application correctness, performance, and efficiency of use of resources.

This book comprises the continuation of a successful series of publications that started with the second International Parallel Tools Workshop in 2007. It contains contributed papers presented at the International Parallel Tools Workshop 2016,[1] held October 4–5, 2016, in Stuttgart, Germany. The workshop was jointly organised by the High-Performance Computing Center Stuttgart (HLRS)[2] and the Center for Information Services and High Performance Computing of the Technical University of Dresden (ZIH-TUD).[3]

With the IPTW 2016 held in Stuttgart, we celebrate the tenth anniversary of this workshop series. The motto of this year's event was the transition of initially academic prototype-like helpers to stable production tools and further on to commercial products. Indeed, in the last decade the HPC tools landscape has changed dramatically: simple command-line scripts have developed into fully-flagged automatic or automated analysis suites, have been provided with rich graphical user interfaces where appropriate, and enriched with a broad set of documentation and training material. Close collaboration within the tools community has led to wide-spread acceptance of terminology, and standardisation of techniques and

[1] http://toolsworkshop.hlrs.de/2016/.

[2] http://www.hlrs.de.

[3] http://tu-dresden.de/die_tu_dresden/zentrale_einrichtungen/zih/.

data-formats. This allows for a tight integration and interoperability of commercial and open-source tools alike, which increases the user's productivity by shedding light on the issue from different perspectives.

Although there are many open source tools from the researcher community today, there are surprisingly only a few commercial products. Here, the keynote held by Allen D. Malony with the title "The Value Proposition for Parallel Tools", posed the question what the value of these tools is, how they are funded and whether monetizing tools is the right measure of success—and even whether parallel tools have, in and of themselves, any value at all.

As has always been the case, HPC users are faced with ever increasing complexity of hardware and software aspects. One of these aspects is the deep memory hierarchy in today's computing systems ranging from the introduction of a forth cache level in the POWER8 processors to the introduction of new technologies such as NVRAM at the opposite end of the hierarchy. Thus, all but the simplest "fit to cache models" are difficult to handle by the programmer at the moment of writing code. So tools are required to assist at this point to reach the best performance and energy efficiency. Such issues are addressed by the tool Kernkraft as presented in the first chapter of this book. Kernkraft combines memory models and simulators with instruction analysis to transform loops automatically for best performance on a given target architecture.

Another aspect introducing complexities is system size. While MPI programs running on tens or hundredth of single core CPUs could be understood relatively easy, today's applications running on multi- or many-cores need to use hybrid MPI+OpenMP models. Thus communication does not only include more processes but also shows more complex patterns as more sophisticated algorithms use, e.g. overlapping techniques, and may even require to take data-locality into account. Detection and reasoning on communication pattern becomes increasingly important to understand the application behaviour. The topic of the second chapter is the detection of these communication patterns independently of the number of resources used and relative to their process placement.

The shared-memory parallel programming model OpenMP has recently been extended significantly to support data-dependencies between computation tasks and off-loading of tasks onto heterogenous accelerators of various types. OpenMP programmes thus become much more complex than the traditional fork-join model. This has led to an effort of tool developers to define a standard OpenMP Tools interface (OMPT), which is scheduled to be included in the next major version of OpenMP. In the third chapter, the tool Extrae show-cases the potential for tracing and sampling on heterogenous hardware of this new interface.

At the same time common functionalities in tools are standardised in a way, that collaboratively maintained tools APIs are created. These APIs allow to focus on new tool features without the need of re-inventing the infrastructure below. One of them is the Score-P measurement infrastructure, which provides an extendable plugin interface. In the fourth chapter, the potential of this interface is presented by a variety of new ideas to support the development process of parallel applications.

Another important trend is one-sided communication to reduce communication overheads. This is also reflected in the fact that the Message Passing Interface (MPI) has undergone various updates in this area. Two chapters are related to the fundamental issue of synchronisation in this programming model, where many errors can be made by the user at the moment. One is related to the detection of synchronisation errors and the other on lock contention.

Finally, the last chapter of this book gives a bit of an outlook on the path to future of parallel programming: automatic program transformation. While a specialist may provide a few simple transformation rules to increase the efficiency of a code on a given hardware, the combination of a large set of such transformation rules leads to such a number of combinations that only a tool is capable of evaluating them in an effective way. Here an approach based on machine learning techniques is presented.

The topics covered in this book, clearly show the potential of giving parallel programming tools a better, first-hand view on the internals of a parallel programming model, as for instance by providing standard tool interfaces as OMPT, thus allowing them to present to the user a fuller and semantically richer picture of the application state. Also, the trend to re-use common tool infrastructure, e.g. by providing standard APIs, data-formats, or plugin facilities, leads not only to faster development of tools on a wider range of systems, but also to the creation of new tools beyond the original scope of infrastructure. Finally, tools are semi-automatically assisting developers with complex tasks such as analysis of applications structure regarding communication or cache-access patterns, or code transformations for various underlying hardware.

Stuttgart, Germany — Christoph Niethammer
Stuttgart, Germany — José Gracia
Dresden, Germany — Tobias Hilbrich
Dresden, Germany — Andreas Knüpfer
Stuttgart, Germany — Michael M. Resch
Dresden, Germany — Wolfgang E. Nagel
January 2017

http://www.springer.com/978-3-319-56701-3

Tools for High Performance Computing 2016
Proceedings of the 10th International Workshop on Parallel Tools for High Performance Computing, October 2016, Stuttgart, Germany
Niethammer, C.; Gracia, J.; Hilbrich, T.; Knüpfer, A.; Resch, M.M.; Nagel, W.E. (Eds.)
2017, IX, 140 p. 60 illus., 48 illus. in color., Hardcover
ISBN: 978-3-319-56701-3