Preface

Michael Bader and
Hans-Joachim Bungartz

Institut für Informatik,
TU München, Germany
E-mail: bader@in.tum.de
E-mail: bungartz@in.tum.de

Ulrich Rüde

Technische Fakultät der
Universität Erlangen-Nürnberg, Germany
E-mail: ruede@informatik.uni-erlangen.de

Biographical notes: Michael Bader is senior research assistant at the chair of Scientific Computing in Computer Science (SCCS) at TU München (TUM). He received a PhD in Informatics at TUM in 2001. His research interests include hardware-aware algorithms in scientific computing based on space-filling curves.

Hans-Joachim Bungartz is Professor of informatics and mathematics and Head of the Scientific Computing chair in TUM's Department of Informatics. His main fields of interest comprise CSE, scientific computing (with a focus on efficient discretisations for PDE and fast solvers), and high performance computing.

Ulrich Rüde is Professor in the Department of Computer Science at University Erlangen-Nuremberg and Head of the Chair for System Simulation. His fields of interest include high performance computing, computational science and engineering, and adaptive multilevel algorithms.

Many simulation tasks in science and engineering are computationally very intensive. More complex models entailing more degrees of freedom not only require more available space in memory, but also demand faster and faster machines to compute solutions to these models within an acceptable time frame.

The recent trends in hardware development have added additional challenges to this scenario, because today's codes no longer guarantee to exploit the performance of next-generation hardware to a satisfying degree:

- The so-called *memory wall*, i.e., the increasing performance gap between memory access and processor speed, forces scientific computing software to deal with the efficient use of hierarchies of cache memory.
- Multicore, hyperthreading, and similar keywords reflect the current trend towards having more than one processor core on a single CPU. However, only simulation codes that allow for such fine-level parallelism will be able to exploit the theoretical performance gain.
- HPC systems are often no longer vector computers only, or multiprocessor computers only, but more likely of a hybrid architecture, which again poses additional demands to the programmer.

Hence, it is no longer sufficient for a simulation code to show an optimal efficiency in the classical $\mathcal{O}(n)$- or $\mathcal{O}(h)$-type efficiency considerations. We also need to make sure that algorithms can be efficiently mapped to the underlying hardware; we need to make sure that the given memory, CPU performance, and parallelism are exploited in an optimal way; and finally, we have to keep in mind that a scientist should be able to produce such an implementation within an acceptable amount of time.

During the 19th Symposium on Simulation Technique organised by ASIM – the German committee for simulation and German branch of EUROSIM, the Federation of European Simulation Societies, – we organised a minisymposium on 'Implementation aspects in scientific computing'. The contributions covered a wide range of problems – from new algorithmic approaches (hardware-aware or cache oblivious) via concepts for generic parallelisation up to optimised implementations for specific hardware (both classical and new, such as the Cell processor) – and formed the starting point for the present special issue of *IJCSE*.

We present six contributions that are dedicated to such algorithmic and implementation issues in scientific and high performance computing. While they are certainly not able to cover all questions of hardware-aware simulation, they do, however, represent a good number of active areas

of research in this field, and offer interesting approaches for a diverse collection of scientific problems.

Donath et al. present a performance comparison of different, inherently cache efficient (so-called *cache oblivious*) parallel implementations of the lattice Boltzmann method. Starting from a classical, cache oblivious approach for stencil-based computations, they extend this method for multi-core and multi-socket systems.

Bader et al. also present a cache oblivious approach, however, they concentrate on the question of generating, adaptively refining, and implementing iterative solvers on recursively structured, adaptive triangular grids, where low memory consumption and exploitation of locality properties of space-filling curves are the features of their approach.

Weidendorfer and Trinitis, in their work, concentrate on the question how to improve the cache performance of existing algorithms and implementations in high performance computing. They introduce an infrastructure or application-controlled prefetching of data, also anticipating multiprocessor or multicore platforms, where lightweight processors could be used for such optimisations of cache access.

Stürmer et al. demonstrate how multigrid solvers can be efficiently implemented in a hardware-oriented way on modern microprocessors. While highly tuned linear algebra libraries are already commonly available (see BLAS, LAPACK, etc.), there is clearly a lack of such approaches for multilevel methods.

Göddeke et al. also tackle the efficient implementation of multigrid solvers. However, their approach is based on exploiting the computational power of special-purpose hardware, in particular that of Graphical Processing Units (GPUs). They present how respective well-defined solver modules can be integrated into an established Finite Element package.

Blatt and Bastian, finally, address an approach for the generic parallelisation of iterative solvers within typical Finite Element computations. They propose a clear-cut separation of parallelisation aspects from data structures, which allows to stay flexible with respect to parallel programming paradigms and, hence, underlying parallel hardware.

Research on implementation aspects for simulation software is far from being a closed chapter. Hence, with this special issue of *IJCSE*, we not only want to provide an overview of running activities, but also hope to give stimulations for future work in this thriving sub-field of scientific computing.