J Supercomput (2018) 74:1419–1421
https://doi.org/10.1007/s11227-018-2278-9

Special section on parallel programming

Ami Marowka[1] · **Przemysław Stpiczyński**[2]

This special section of *The Journal of Supercomputing* contains revised and extended versions of selected papers presented at the 7th Workshop on Language-Based Parallel Programming Models, WLPP 2017, which was a part of the 12th International Conference on Parallel Processing and Applied Mathematics, PPAM 2017, held on September 10–13, 2017, in Lublin, Poland.

WLPP 2017 was a two-day workshop focusing on high-level programming for large-scale parallel systems and multicore processors, with special emphasis on component architectures and models. Its goal was to bring together researchers working in the areas of applications, computational models, language design, compilers, system architecture, and programming tools to discuss new developments in programming Clouds and parallel systems.

Papers in this section cover the most important topics presented during the workshop. The first two deal with parallel programming models. Thoman et al. [4] provide an initial task-focused taxonomy for HPC technologies, which covers both programming interfaces and runtime mechanisms and discuss its usefulness by classifying state-of-the-art task-based environments that are used today. Posner and Fohry [7] propose a hybrid work stealing scheme, which combines the lifeline-based variant of distributed task pools with the node-internal load balancing implemented as an extension of the APGAS library for Java.

✉ Przemysław Stpiczyński
przem@hektor.umcs.lublin.pl

Ami Marowka
amimar2@yahoo.com

1 Parallel Research Lab, 52900 Jerusalem, Israel

2 Maria Curie–Skłodowska University, Pl. Marii Curie-Skłodowskiej 1, 20-031 Lublin, Poland

The next set of papers concern language-based parallelization techniques. Marowka [6] investigates the efforts devoted by the Python community to improve the performance of this language in scientific applications. Mostly, he focuses on Numba, a much-promised solution that preserves the performance improvement in code ported to different target architectures. Stpiczyński [8] evaluates intrinsics, OpenMP, TBB and Cilk Plus as basic language-based tools for simple and efficient optimization of computational problems that need both task and data parallelization techniques. Blöcker and Hoffmann [2] propose a fully deterministic process calculus for parallel and distributed programming implemented in Haskell as a domain-specific language. Batko and Kuta [1] compare their new functional language Anemone with actor systems of Scala and Erlang.

Finally, four papers present the use of parallel programming for solving real-world computational problems. Bylina [3] demonstrates the methodology for parallelization of finding stochastic bounds for Markov chains on multicore and manycore platforms using OpenMP and Cilk Plus. Tokura et al. [10] present an almost optimal column-wise prefix-sum algorithm for GPUs. Halver et al. [5] discuss the use of OpenCL as a language for implementation of classical molecular dynamics simulation useful for benchmarking different hardware platforms. Szustak [9] presents an innovative strategy for the data flow synchronization in stencil parallel computations on shared-memory systems.

Acknowledgements The guest editors of this special issue wish to thank the authors for their interesting contributions and the reviewers for useful suggestions. Finally, we would like to express our gratitude to Professor Hamid Arabnia (Editor-in-Chief of *The Journal of Supercomputing*) for the opportunity to edit this special issue and his great guidance.

References

1. Batko P, Kuta M (2018) Actor model of Anemone functional language. J Supercomput. https://doi.org/10.1007/s11227-017-2233-1
2. Blöcker C, Hoffmann U (2018) Pardis: A process calculus for parallel and distributed programming in Haskell. J Supercomput. https://doi.org/10.1007/s11227-018-2289-6
3. Bylina J (2018) Parallelization of stochastic bounds for Markov chains on multicore and manycore platforms. J Supercomput. https://doi.org/10.1007/s11227-018-2235-7
4. Dichev PT, Heller T, Iakymchuk R, Aguilar X, Hasanov K, Gschwandtner P, Lemarinier P, Markidis S, Jordan H, Fahringer T, Katrinis K, Laure E, Nikolopoulos DS (2018) A taxonomy of task-based parallel programming technologies for high-performance computing. J Supercomput. https://doi.org/10.1007/s11227-018-2238-4
5. Halver R, Homberg W, Sutmann G (2018) Function portability of molecular dynamics on heterogeneous parallel architectures with OpenCL. J Supercomput. https://doi.org/10.1007/s11227-017-2232-2
6. Marowka A (2018) Python accelerators for high-performance computing. J Supercomput. https://doi.org/10.1007/s11227-017-2213-5
7. Posner J, Fohry C (2018) Hybrid work stealing of locality-flexible and cancelable tasks for the APGAS library. J Supercomput. https://doi.org/10.1007/s11227-018-2234-8

8. Stpiczyński P (2018) Language-based vectorization and parallelization using intrinsics, OpenMP, TBB and Cilk Plus. J Supercomput. https://doi.org/10.1007/s11227-017-2231-3
9. Szustak L (2018) Strategy for data-flow synchronizations in stencil parallel computations on multi-/manycore systems. J Supercomput. https://doi.org/10.1007/s11227-018-2239-3
10. Tokura H, Fujita T, Nakano K, Ito Y, Bordim JL (2018) Almost optimal column-wise prefix-sum computation on the GPU. J Supercomput. https://doi.org/10.1007/s11227-018-2242-8