DOI 10.12694/scpe.v17i1.1145
Scalable Computing: Practice and Experience ISSN 1895-1767
Volume 17, Number 1, pp. iii–iv. http://www.scpe.org © 2016 SCPE

INTRODUCTION TO THE SPECIAL ISSUE ON HIGH PERFORMANCE COMPUTING SOLUTIONS FOR COMPLEX PROBLEMS

In the last decades, the complexity of the current and upcoming scientific/engineering problems has increased considerably. Computations involved in numerical simulations, molecular dynamics, computational fluid dynamics, bio-informatics, image processing, linear algebra, deep-learning, information retrieval, or big-data computing are just a few examples of such problems.

At the same time, improvements in high performance computing (HPC) systems are mainly associated with an important increasing in the complexity of computer architectures, making difficult the code optimization. This results in a greater gap between the general scientific / engineering user community (in need of easy access to efficient high performance computations) and the HPC programmers community (who design codes for narrow sub-classes of problems). The development of user-friendly codes for non-HPC-trained user community becomes a big challenge. As consequence, effective use of HPC centers requires specialized / individual training for each user group.

Today, programmers of HPC or/and scientific applications have to deal with numerous details regarding computer architectures at low level to take advantage of the last features of the current and upcoming computing systems. It makes difficult the efficient and optimized software development. Thus, strategies and tools that can help us to adapt our codes over different computing architectures is of vital importance. However, previously, we must know and identify what are most efficient programming strategies and architectonic features. This is a difficult task as both, strategies and features, depend on the particular problem to be dealt. Portability is other important issue today. Multiples kind of processors have arisen in the last years. Most of these current platforms use their own compilers, languages, etc., being a very complex task to implement portable codes.

This special issue provides several studies, which involve different applications and strategies to improve performance and to achieve better usage of current computing systems. It is composed by three works.

The work "Performance Optimizations for an Automatic Target Generation Process in Hyperspectral Analysis" by Fernando Sierra-Pajuelo, Abel Paz-Gallardo, and Antonio Plaza presents several optimizations for hyperspectral image processing algorithms intended to detect targets in hyperspectral images. The algorithm used is the automated target generation process (ATGP) and the optimizations comprise parallel versions of the algorithm developed using open multi-processing (OpenMP, including Intel Xeon Phi) and message passing interface (MPI).

The study "Using Computational Geometry to Improve Process Rescheduling on Round-Based Parallel Applications" by R. da Rosa-Righi, V. Magalhaes-Guerreiro, G. Rostirolla, V. Facco-Rodrigues, C. Andre da Costa, and L. Dagnino-Chiwiacowsky, proposes two novel heuristics applied to process rescheduling, named MigCube and MigHull, to choose the candidate processes for migration and their destination. Both heuristics consider the use of computational geometry for plotting computation, communication and migration costs metrics in a 3D graph without any user intervention.

Finally, the work "Many-Task Computing on Many-Core Architectures" by Pedro Valero-Lara, Poorima Nookala, Fernando L. Pelayo, Johan Jansoon, Serapheim Dimitropoulos, and Ioan Raicu, studies what are the Many-Task Computing (MTC) programming mechanisms to take advantages of the massively parallel features of current hardware accelerators for the particular target of MTC. Also, the hardware features of the two dominant many-core platforms (NVIDIAs GPUs and Intel Xeon Phi) are also analyzed for our specific framework. This study consisted of comparing the time consumed for computing in parallel several tasks one by one (the whole computational resources are used just to compute one task at a time) with the time consumed for computing in parallel the same set of tasks simultaneously (the whole computational resources are used for computing the set of tasks at very same time). Finally, both software-hardware scenarios were compared to identify the most relevant computer features in each of our many-core architectures.

We would like to thank the editorial board of SCPE and reviewers for their effort and time, which is very appreciated.

Dr. Pedro Valero Lara, The University of Manchester, UK.
Prof. Dr. Fernando L. Pelayo, University of Castilla-La Mancha, Spain.
Prof. Dr. Johan Jansson, Basque Center for Applied Mathematics (BCAM), Bilbao, Spain and KTH, Royal Institute of Technology, Stockholm, Sweden.