Topic 9
Parallel Programming: Models, Methods and Languages

José C. Cunha, Sergei Gorlatch, Daniel Quinlan, and Peter H. Welch

Topic Chairs

This topic provides a forum for the presentation of research results and practical experience in the development of parallel programs. Advances in algorithmic and programming models, design methods, languages, and interfaces are needed to produce correct, portable parallel software with predictable performance on different parallel and distributed architectures.

The topic emphasizes results that improve the process of developing high-performance programs, including high-integrity programs that are scalable with both problem size and complexity. Of particular interest are novel techniques by which parallel software can be assembled from reusable parallel components without compromising efficiency. Related to this is the need for parallel software to adapt, both to available resources and to the problem being solved.

This year, 13 papers were submitted to this topic. Each paper was reviewed by four reviewers and, finally, we were able to select 7 papers. Globally, the accepted papers discuss methods and programming language constructs to promote the development of correct and efficient parallel programs.

The approaches based on higher-order skeletons are discussed in two papers, for computations on two-dimensional arrays, and for dynamic task farming. Data parallel programming is discussed in another paper concerning the automatic parallelisation of "for-each" loops for grid and tree algorithms. Shared memory parallel programming is discussed in two papers that propose extensions to OpenMP, for handling irregular parallel algorithms, and for improved control and synchronisation of multiple threads. Improved support for multithreading models is also discussed in another paper that proposes a methodology towards more efficient memory management for threading libraries that are based on non-preemptive models. Distributed termination detection is discussed in a paper that proposes the concept of "partial quiescence" as a construct of a distributed programming language.

We would like to thank all the authors who submitted papers to this topic, and the external referees, for their contribution to the success of this conference.