Compared to sequential programs, the execution behavior of parallel and distributed programs generally is much harder to predict during program development and can sometimes be surprising even for skilled programmers. Runtime tools, which monitor a program's execution and allow the programmer to systematically analyze its run-time behavior have become mature and useful instruments to address this problem. However, there is a high demand for still more advanced tools that — amongst other things — address user-friendliness by including an automatic analysis process or even an automatic feed-back, or support a more detailed analysis of an application's behavior, e.g. with respect to the memory hierarchy.

This special session includes four research papers that present new ideas and advanced techniques, which are not yet found in established programming tools. Three of these contributions deal with performance analysis or prediction, indicating that on the one hand there is a high demand for sophisticated performance tools, on the other hand there are still many interesting research issues in this area.

The paper “Supporting the Memory System Evaluation with a Monitor Simulator” by J. Tao introduces a simulation-based environment that provides a detailed analysis of the memory behavior of applications running on PC clusters with a shared memory system. The environment is based on a monitoring component which can be combined with any target system simulator. The monitoring component can be connected to several levels of the memory hierarchy (L1 cache, L2 cache, local memory, remote memory), which allows to collect accurate and comprehensive performance data.

In their contribution “Performance Modeling, Analysis and Simulation of Scientific Applications: A Case Study”, T. Fahringer et al. also use simulation techniques for performance analysis. In contrast to the previous paper, however, the simulation is on a higher level. It uses a performance model of an application and a simulator for distributed heterogeneous systems to predict the performance of the application on a specific target system. Since the simulator takes into account the detailed architecture of both the network and the compute hosts, the authors could achieve rather accurate predictions for a real-world scientific application.

F. Wolf and B. Mohr address the issue of a user-friendly, automatic analysis of performance problems in their contribution “Automatic Performance Analysis of Hybrid MPI/OpenMP Applications”. Their EXPERT tool analyses a trace file produced during a program's execution. It searches for patterns that represent common performance problems resulting from an inefficient use of the underlying programming models. Besides the class of performance behavior, the analysis also determines the position in the call-tree and the thread where the behavior occurs. These results are presented in a three-dimensional hierarchy which can then be browsed by the programmer.

Besides performance analysis, developers of parallel programs are faced with another at least equally important problem: debugging. Particularly, reproducing parallel program executions in a deterministic way is a notoriously hard and time consuming task. Making this task easier is the goal of the session's fourth contribution.

In their paper “ERoS: An Efficient Method in Minimizing The Replay Time Based on The Replay Dependence Relation”, N. Thoai et al. present an enhanced method that allows a programmer to quickly and deterministically replay a program execution up to a specified point, where the program state may then be examined in detail. This is achieved by periodically checkpointing the individual processes and by additionally logging a small number of messages exchanged between these processes. By properly choosing the messages to be logged, the ERoS method can guarantee that the time needed for the replay is less than a given multiple of the checkpointing interval.