1) course for PhD students:

Topic: Energy-aware Execution of Fork-Join-based Task Parallelism

Authors: Thomas Rauber, Gudula Rünger

Due to environmental and monetary concerns, it is increasingly important to reduce the power consumption of applications in parallel and high-performance computing.

In this talk, we consider the popular fork-join pattern that is often used to express parallel program execution for regular applications. We first give a short overview on parallel programming in OpenMP and show how parallel programs with fork-join parallelism can be expressed in OpenMP. Then we consider the modeling and analysis of power consumption and performance with an analytical energy model at the application program level. Based on this model we address the question how to optimize the power consumption while maintaining a good performance. The analysis is performed for a task-based programming model in which tasks can be executed concurrently in a fork-join programming pattern. The modeling of the power consumption of concurrently executed tasks is performed with an analytical energy model for frequency scaling. Frequency scaling factors that lead to a minimum energy consumption are derived and are used in task-based scheduling algorithms. An experimental evaluation provides simulations for a large number of randomly generated task sets as well as energy measurements on an Intel Sandy Bridge architecture using a complex application from numerical analysis.

2) seminar

Title of the talk: Programming Support and Scheduling for Communicating Parallel Tasks

Authors: Jörg Dümmler, Thomas Rauber, Gudula Rünger

Abstract:
Task-based programming models are beneficial for the development of parallel programs for several reasons. They provide a decoupling of the specification of parallelism from the scheduling and mapping to execution resources of a specific hardware platform, thus allowing a flexible and individual mapping. For platforms with a distributed address space, the use of parallel tasks, instead of sequential tasks, adds the additional advantage of a structuring of the program into communication domains that can help to reduce the overall communication...
In this talk, we consider the parallel programming model of communicating parallel tasks (CM-tasks), which allows both task-internal communication as well as communication between concurrently executed tasks at arbitrary points of their execution. We propose a corresponding scheduling algorithm and describe how the scheduling is supported by a transformation tool. An experimental evaluation using synthetic task graphs as well as several complex application programs shows that employing the CM-task model may lead to significant performance improvements compared to other parallel execution schemes.