05101 Executive Summary
Scheduling for Parallel Architectures: Theory, Applications, Challenges
— Dagstuhl Seminar —

E. Altman[1], J. Dehnert[2], C. W. Kessler[3] and J. Knoop[4]

[1] IBM TJ Watson Research Center, US
erik@watson.ibm.com
[2] Transmeta - Santa Clara, US
dehnertj@acm.org
[3] Linköping Univ., SE
chrke@ida.liu.se
[4] TU Wien, AT
knoop@complang.tuwien.ac.at

Scheduling, the task of mapping computation units to time slots on computing resources for execution, is important for the effective use of resources in all kinds of parallel systems, ranging from the level of more coarseŋgrain tasks in multiprocessors, clusters and computational grids, to mediumŋgrain tasks at the loop level, down to instructionŋlevel parallelism (ILP).

Scheduling issues are of crucial importance in very diverse areas ranging from operating systems and realtime systems to network management to static and dynamic program optimization and code generation. Likewise, they evolve on very different levels of granularity, from coarse grain task and job scheduling over loop scheduling to fineŋgrain instruction scheduling. Though highly interrelated, these fields are tackled by usually independently working communities. However, emerging processor architectures such as chip multiprocessors will demand effective hybrid scheduling strategies that unify previously separate scopes of scheduling.

In practice, scheduling problems often do not appear in isolation but come with a domain-specific context that—explicitly or implicitly—introduces interdependencies with other optimization problems. For instance, when compiling for parallel execution platforms, decisions made in scheduling depend on and influence other aspects of the problem of generating efficient parallel code, such as resource allocation, clustering, or program transformations, such that scheduling can rarely be considered as an isolated problem. Such interdependences, even though perhaps most apparent for instructionŋlevel parallelism, appear at all levels of parallelism and are solved by various techniques, including heuristics, integer programming, dynamic programming, or genetic programming. Integrated approaches are generally more flexible but suffer from an increased problem complexity.

Interestingly, the research communities for task-level, loop-level and instruction-level scheduling appear to be quite separated from each other. Furthermore, there appears to be a gap between the theoretical foundations of scheduling, formulated in terms of abstract machine models, and the algorithms developed in both academia and industry for concrete scheduling problems in compilers and run-time systems for parallel computer architectures. This gap is exacerbated by requirements that practical schedulers deal with the complexities of irregular architectures.

The purpose of this seminar was therefore to gather leading experts from these scheduling communities, to identify common approaches, techniques, frameworks and tools, and to stimulate crossŋfertilization between the various scheduling communities. Moreover, we intended to bridge the gap between scheduling theory and methods currently applied in compilers and runŋtime systems for parallel architectures. A third goal was to encourage a constructive dialog between scheduling algorithm designers and developers of parallel architectures, specifically in the embedded systems domain.

31 researchers accepted the invitation to the seminar and met 6–11 March 2005 at Schloss Dagstuhl, Germany. The seminar participants represented a broad spectrum of research on scheduling, including instruction scheduling, job scheduling, task scheduling, loop scheduling, parallel computer architecture, and scheduling theory.

With the invitation and the opening address, we provided the following guiding questions:

- How can we bridge the gap between scheduling theory and practice?
- Can the practical ILP scheduling problems broaden the theory models?
- In particular, do recent microŋarchitectural trends such as clustered architectures add fundamentally different factors to the problem?
- Is there current theory that can lead to interesting practical algorithms?
- What are the interference effects between taskŋlevel, loopŋlevel and instructionlevel scheduling?
- Can existing scheduling approaches be transferred to other problem domains, granularities, or architecture models?
- What are the phaseŋordering effects, and the techniques, potential, and limitations of integrating scheduling with other transformations or code generation phases?
- Can we define generic scheduling approaches for flexible optimization goals (execution time, stack/register space, energy consumption)?

A central goal of this seminar was thus to bring together leading experts of the various communities to foster discussions on the usability and usefulness of approaches developed for specific areas and the impact they may have to others. By means of crossŋfertilization and synergy the seminar should contribute to both a better understanding of the key issues of scheduling and to further advancing the state-of-the-art in the various fields. The specific atmosphere of Dagstuhl Seminars, which can be characterized by openness, accessibility and cooperation, and which is supported by Schloss Dagstuhl's architecture, its services and facilities, encourages both formal and informal meetings and discussions and therefore provided a perfect environment to achieve these goals.

The seminar started with a short introductory presentation from each participant, with his/her view on the topic and expectations for the seminar. During the seminar week, there were 26 presentations by the participants, discussions in plenum and in smaller working groups, and work-in-progress sessions. The presentation abstracts are given in the remainder of this seminar report.

At the end of the seminar, the general impressions brought up in the final wrap-up discussion were thoroughly positive: The representatives of the various scheduling communities, albeit being different in scope, application domain, and even in the terminology used, really understood each other and could learn and got inspirations from each other's presentations. For instance, a constructive dialog was initiated between compiler scheduling and job/task scheduling theory researchers, providing in the one direction new problem formulations and application areas for the theory community and in the other one the wish from the practitioner's side to the theory community for better communication of the constraints and limitations of theoretical solutions, such as assumptions in the model. There was an insight into the problem of different scheduling communities and of insufficient communication of results e.g. between theory and the various application domains, leading to undesirable effects such as reinvention of the same or similar ideas in different fields and with different terminology. The need for bridging the gap between scheduling theory and practice was recognized, although a fundamental solution to this problem, of course, cannot be provided within the scope of a single seminar week. There were suggestions to create a common web page collecting resources, existing results, ongoing work, open problems etc.; to compile a common list of most important literature references, to agree on a common benchmark suite for scheduling problems that covers a broader range of problems and in particular includes malleable tasks, and even to build a common experimental compiler platform that covers all types of scheduling problems in a single framework and therefore allows e.g. to empirically study trade-off effects between instruction-level, loop-level, and thread-level scheduling, which turn out to be hard to define in a generic way because they heavily depend on the underlying hardware platform. It may be premature to speak of the birth of a new, merged scheduling community, but at least there exist concrete plans to organize a successor workshop with the same broad scope of scheduling issues for parallel architectures, prospectively in 2007.