FRIEDRICH-ALEXANDER-UNIVERSITÄT ERLANGEN-NÜRNBERG
INSTITUT FÜR INFORMATIK (MATHEMATISCHE MASCHINEN UND DATENVERARBEITUNG)

Lehrstuhl für Informatik 10 (Systemsimulation)

High Performance Computing Education for Students in Computational Engineering

Uwe Fabricius, Christoph Freundl, Harald Köstler, and Ulrich Rüde

Technical Report 05-1

High Performance Computing Education for Students in Computational Engineering

Uwe Fabricius, Christoph Freundl, Harald Köstler, and Ulrich Rüde

Abstract

Numerical simulation using high performance computing has become a key technology for many scientific disciplines. Consequently, high performance computing courses constitute an essential component within the undergraduate and graduate programs in Computational Engineering at University of Erlangen-Nuremberg. These courses are also offered as optional courses in other degree programs, such as for majors in computer science.

1 The Erlangen Computational Engineering Program

The courses in high performance computing at University of Erlangen-Nuremberg are primarily motivated by the the *Computational Engineering* (CE) program that has been initiated by the Department of Computer Science in 1997 as a prototype two-year postgraduate program leading to a Master degree. The corresponding undergraduate program has been started in 1999. Together these two programs are accepting approximately 30 new undergraduate students and 45 graduate students, annually.

The traditional German university degree in the sciences and the engineering disciplines is the *Diplom* which corresponds approximately to the academic level of a Master degree in the US educational system. Currently the system is being reformed according to the so-called *Bologna Process*, a political agenda that is aimed at introducing a Europe-wide, standardized university degree system by 2010. This reform process will lead to an educational structure with a first degree on the Bachelor level, on top of which graduate programs leading to the Master and Doctorate can be built.

The Erlangen Computational Engineering programs are prototype implementations of this new system, since they already award Bachelor and Master degrees. Generally, the Bachelor-Master structure of academic programs is still in an experimental stage in Germany, but the transition away from the Diplom degree will accelerate during the next couple of years. All core courses of the CE Master program are taught in English and are thus open to international students without knowledge of German.

The CE program is built around a core of computer science and mathematics courses. Additionally, each student must select a *technical application* field. Currently CE in Erlangen offers application specializations in

- Mechanical Engineering
- Micro Electronics
- Information Technology
- Automatic Control
- Thermo- and Fluid Dynamics
- Material Sciences
- Sensor Technology

The curriculum requires approximately an equal number of credits in mathematics, computer science, and the application field. The university education system in Germany traditionally puts

a strong emphasis on thesis work and thus, like the Diplom degree, the Master requires a full six month thesis, and even for the Bachelor degree students must spend three months on thesis work. A more detailed description of the programs can be found in [ER-CSE]. Up-to-date information can be obtained from the Internet[1].

2 Bavarian Graduate School in Computational Engineering

Starting in fall 2004, the Bavarian Graduate School In Computational Engineering[2] (BGSCE) has been established as a network of excellence between Friedrich-Alexander-Universitt Erlangen (FAU) and Technische Universität München (TUM). The partners in this consortium consist of three existing Master Programs in the field of Computational Science and Engineering:

- Computational Mechanics (COME)[3] at TUM
- Computational Science and Engineering (CSE)[4] at TUM
- Computational Engineering (CE) at FAU

Students of the Bavarian Graduate School in Computational Engineering are recruited from the best students of each participating Master program. These students stay enrolled in their home program, but they are required to take an extra load of 30 ECTS[5] credits. In turn they are awarded a Master degree *with Honours*. The extra credits must been earned partly in classes out of the other Master programs. This is made possible by offering suitable courses in block form or in the form of summer schools.

This trans-institutional network of excellence has won special funding in a state-wide competition from the state of Bavaria in its Elite-Network[6] initiative.

3 Simulation as a core field in Computational Engineering

The undergraduate CE program is based on the traditional German four-semester *engineering mathematics* sequence, but this is complemented by two semesters of numerical mathematics in the third and fourth semester. Additionally, students are required to take a newly developed course in *Algorithms and Data Structures for Continuous Systems* to be taken in the fourth semester. This course is unique in that it presents algorithms for handling continuous data, such as required for image and video processing, computer graphics, visualization, and the simulation of technical systems. It contains material from each of these fields together with their theoretical background in (numerical) mathematics.

Building on the material taught in these courses during the first two years of study, the Department of Computer Science offers a two semester sequence in *Simulation and Scientific Computing* (SISC). These courses are designed to provide a broad view of numerical simulation and as such they put a significant emphasis on the basic elements of *High Performance Computing* (HPC). The SISC sequence is required for CE students and can be chosen as optional courses within the Computer Science (CS) curriculum. New Master degree students who do not yet have an equivalent background are also required to take the SISC sequence.

Besides the core curriculum of required courses, students can and must select additional credit hours from an exhaustive list of optional courses that can be taken either from the student's application field, computer science, or applied mathematics. Though any course of the conventional degree programs of the participating departments can be chosen, students are intensively advised and guided individually to enable them to find suitable combinations of courses.

Among the optional courses there are several that offer a further specialization in high performance computing topics. The most prominent here are *Parallel Algorithms* and *Programming*

[1] http://www10.informatik.uni-erlangen.de/CE/
[2] http://www.bgsce.de/
[3] http://www.come.tum.de/
[4] http://www.cse.tum.de/
[5] European Credit Transfer System
http://europa.eu.int/comm/education/programmes/socrates/ects_en.html
[6] http://www.elitenetzwerk-bayern.de/en/index.html

Techniques for Supercomputers. The structure outlined here has been the result of an update of the curriculum in 2003/04 and as such differs slightly from the state as described in [ER-CSE, RR].

On the graduate and advanced undergraduate level, we have also created new classes with the goal to better bridge the gap between the disciplines. These courses are interdisciplinary and are taught jointly by faculty from the different departments. They often integrate an aspect of high performance computing.

One such course is *Numerical Simulation of Fluids* which is presented jointly by Chemical Engineering and Computer Science faculty. Using [NFL] as the basic text, the course teaches students to develop an incompressible Navier-Stokes solver from scratch. This is a significant difference from how computational fluid dynamics is usually being taught in engineering. While a classical course would introduce students to existing computational fluid dynamics software and teach them how to use (and possibly extend) it, our course is deliberately designed to teach the fundamentals of flow simulation, even if this comes at the cost of being restricted to what students can accomplish in programming during one semester. The first half of the course has weekly assignments that result in a core 2D fluid simulator. The method is based on a staggered grid finite difference discretization, explicit time stepping for the velocities, and a marker-and-cell method for dealing with nontrivial geometries.

From our experience, the feeling of accomplishment results in a very high motivation for the students. When the core solver has been implemented, students are individually guided to adapt and apply their code to a more complicated application scenario. For this, they form teams of up to three students. Typical projects include the parallelization of the code either for distributed or shared memory parallel execution. In this way the course teaches high performance computing aspects in an integrated form, as driven by a typical application scenario. For students this is especially profitable, when they additionally take one of the special courses with HPC focus and as outlined in the following section.

4 High Performance Computing Courses

The department offers several courses with special emphasis on high performance computing. Besides the mandatory material included in the two-semester sequence *Simulation and Scientific Computing* (SISC), Computational Engineering students can choose courses from the following list

- Cluster Computing
- Parallel Algorithms
- Programming Techniques for Supercomputers.

While the first is being designed primarily for computer science students, giving an overview of parallel computing with clusters from a CS perspective, the latter two are primarily oriented at the needs of CE students.

Parallel Algorithms provides a general overview of parallel computing techniques and algorithms. This is complemented by *Programming Techniques for Supercomputers* which is taught out of the computing center and is specifically aimed at the performance optimization and parallelization of typical simulation algorithms.

Each of the courses is self contained so that there is some unavoidable overlap in the material presented, when a student takes all courses, but this makes it possible to choose these courses independently. Typically, a student will choose two of these courses depending on his or her special interests.

5 High Performance Computing Topics

In the following we will describe in some more detail some of the material that is currently taught as part of the course SISC and which is mandatory for all CE students.

The development over the past decade has brought enormous progress in the performance of computers. Unfortunately, the performance boost has come at the price of an ever increasing complexity of systems, an increasing internal parallelism even within the CPU, deep pipelines, and

a widening gap between memory and CPU performance. Consequently, it becomes increasingly difficult to exploit the performance potential even of single CPU systems. Additionally, many applications require parallel processing using clusters of PCs or parallel supercomputers.

Some of the program optimization techniques are similar to the problem of vectorizing algorithms, other aspects are typical for hierarchical memory systems, other are particular for specific CPU families. Outside the high performance community this problem receives relatively little attention and is therefore not well taught in standard computer science classes. However, since this knowledge has become essential for successful high performance computing, it should be addressed in the basic HPC classes of a CE program.

Modern techniques of single CPU program optimization are therefore included in a sequence of seven 90 minute lecture units within the SISC sequence. The material is partly based on the monograph by Goedecker and Hoisie [GH] which is used as the textbook for this part of the course. The material is roughly organized into the units

- review of computer architecture
- examples of high performance systems
- basic efficiency guidelines
- code profiling and tuning
- optimization of floating point operations
- optimizing of memory access
- cache blocking

Students are grouped in teams of three and have to work on three assignments which have the character of a little projects. The results have to be presented by the team in a short 10 minute talks. Each team is required to prepare a set of approximately 10 slides. The presentations (also of the German students) are given in English, thus providing students with a valuable experience in giving presentations to an international audience. This scheme has evolved over several years and has proved to be very motivating for all students. In particular it often leads a very fruitful combination of cooperation and competition. Generally, students often put much more than average effort into the assignments and often go much beyond the required work. The topics of the three core assignments may change from year to year. A typical setup is as follows:

- Matrix-matrix multiply: Here students are given the (seemingly) simple task to code a matrix-matrix multiplication and to compare the performance of different loop orders. Students are then given access to a highly optimized code that performs about ten times faster. This comparison code is taken from the ATLAS web site[7], see also [ATLAS]. Students are required to use profiling tools (as discussed in the lectures) and present their analysis why the differences in performance occur. Though this is not required in the assignment, the best students will typically explore blocking techniques or discuss using the Strassen multiplication algorithm for faster performance.

- Gauss-Seidel-Iteration for Poisson's equation in 2-D on a square grid in red-black order, as used as a smoother within a multigrid algorithm: Students are required to use blocking techniques, experiment with different CPU architectures, analyze and explain their findings.

- Gauss-Seidel for a (stationary) variable coefficient Poisson-like partial differential equation (PDE), implemented both on a structured grid and alternatively using a compressed row sparse matrix data structure. Students are required to try various optimization techniques and to analyze and present their performance measurements, in particular in comparison with the previous assignment.

The remainder of the lecture covers special algorithms. Typical topics are the Lattice Boltzmann method for simulating fluid flow and as an example of using cellular automata in CE. Other topics include the conjugate gradient and multigrid algorithms as the dominant algorithms in PDE solvers.

[7] http://math-atlas.sourceforge.net/

Typical further assignments will require students to implement these algorithms, and thus students will have ample opportunity to exercise the HPC programming techniques.

Parallel programming is another topic that will be included in elementary form within SISC, once the above mentioned curriculum change has propagated accordingly. More information on the current contents of the course can be downloaded from its web site[8].

The audience in SISC is mixed, consisting of both CE students and CS students. CE students are usually primarily motivated by the applications they want to study, e.g. in fluid dynamics or electrical engineering. For these students, HPC techniques are therefore a tool necessary to successfully solve application problems. CE students who come out of our own undergraduate program have a quite substantial programming expertise and background knowledge in computer architecture. Since they have a good grasp of their application field and the basic algorithms, they tend to be well prepared for the class.

The situation is different for CE Master students whose basic education is in an engineering field, and who often do not have a systematic CS education. For these students, much of the material related to computer and systems architecture is new and requires substantial effort to master. For these students, the assignments are especially valuable, since this may be their first in-depth programming experience.

CS students tend to have the opposite problem. Naturally they are more intrigued by the aspects of HPC to computer architecture, compilers, and programming methodology, but they often do not have a good background in the applications and algorithms. The algorithms taught in the typical CS curriculum do not emphasize numerical simulation, and furthermore, the mathematics in the CS curriculum does not go deep enough in calculus and numerical analysis, as would be required for understanding the more complex mathematical algorithms. From another perspective: this is exactly, why a separate CE program is necessary besides the standard CS curriculum.

The heterogeneous mix of the audience makes teaching the SISC sequence a challenge, but often it is exactly this diversity of student backgrounds and the large variety of interests that leads to an especially lively discussion among the students. The student presentations have proved to be an effective scheme in promoting this kind of interdisciplinary exchange. Furthermore, we believe that this is an essential aspect of Computational Science and Engineering itself, and that exposing students to the need of collaborating and discussing scientific problems with students from a different background is an important part of their education.

While SISC is designed to cover the basic elements of HPC programming, the elective courses and in particular *Programming Techniques for Supercomputers* extend the material to in-depth parallel computing using MPI and OpenMP. Though this course can also be taken by itself, it is a natural follow-up for those CE or CS students who want to put a special focus on high performance computing as part of their education.

6 Parallel High Performance Computers

Exposing students to current HPC systems is essential for a competitive CE education.

For the SISC course described above, students are given access to various workstations and PCs. Currently this is typically an up-to-date Pentium 4 based system and some Athlon-based systems, all running Linux. Previously the class was using several Alpha-based systems (under True-64 Unix) that are still available but which are by now somewhat outdated. An Opteron-based system with dual and quad nodes with a total of 60 processors has been newly acquired in Nov 2004 and is being made available to the students in the course[9]. For students, this system is usually only available with some restrictions, however, it is fully available for student projects, Bachelor -, Master -, and PhD thesis research.

All machines are accessible from the course laboratory, but most students prefer to work remotely either from home or from other labs on campus. Though this may be personally convenient, some students deliberately choose to work in the lab, since this makes it easier to work as a team, exchange ideas with other students, and provides the opportunity to receive individual help and advise from the tutors.

[8] http://www10.informatik.uni-erlangen.de/de/Teaching/Courses/SiwiR/
[9] http://www10.informatik.uni-erlangen.de/Cluster/hpc.shtml

For higher performance requirements (in particular for the course *Programming Techniques for Supercomputers*) the larger machines within the Erlangen Computing Center are available. This currently includes an 300 processor Intel IA-32 based cluster, an SGI Origin 3400 with 28 MIPS R14000 processors and 56 GByte memory, plus an SGI Altix 3700 super-cluster with 28 Itanium2 CPUs and 112 GByte memory.

Additionally, the University of Erlangen is part of a consortium in High Performance Computing (KONWIHR)[10] operated by the state of Bavaria and has access to the supercomputers at the Leibniz computing center[11] of the Bavarian Academy of Sciences. The largest machine there is currently a Two-Teraflop Hitachi SR-8000-F1 super computer with 1300 CPUs. This machine was originally installed in year 2000, and is now scheduled for replacement with a 60 TFlop supercomputer in 2006.

Machines of this class are usually not available freely to students in the above courses, but will be made available for thesis research on the Bachelor , Master , or PhD level, or to students working as research assistants in projects using these machines. The declared goal of the above courses is to train students to become competent users of such HPC computers and thus enable them to work at the leading edge in CE research.

Additionally, the CE program is involved in several international collaborations within Europe and the USA, through which we can gain access (usually for benchmarking comparisons) to an even wider class of machines. Currently the primary machine class not directly available to us are classical vector supercomputers, since our own Fujitsu based vector computer was outdated and has been taken offline, recently. Access to NEC vector supercomputers is e.g. possible through the Stuttgart supercomputer center.

7 Conclusions

At University of Erlangen we have established a systematic set of HPC classes which are primarily oriented at the requirements of our new Computational Engineering program, but which are also open to students in other degree programs. For the courses and thesis research, a comprehensive selection of up-to-date HPC-systems is available.

References

[NFL] Michael Griebel, Thomas Dornseifer, and Tilman Neunhoeffer: *Numerical Simulation in Fluid Dynamics: A Practical Introduction*, SIAM, 1997.

[GH] Stefan Goedecker and Adolfy Hoisie: *Performance Optimization of Numerically Intensive Codes*, SIAM, 2001.

[ER-CSE] U. Ruede: *Computational Engineering Programs at the University of Erlangen-Nuremberg*, in Computational Science - ICCS 2002: International Conference, Amsterdam, The Netherlands, April 21-24, 2002. Proceedings, Part III, P.M.A. Sloot, C.J. Kenneth Tan, J.J. Dongarra, A.G. Hoekstra (Eds.), Lecture Notes in Computer Science 2331, pp. 852–860, Springer, 2002.

[RR] Rosemary A. Renaut and Ulrich Ruede: Editorial, Future Gener. Comput. Syst. 19, vol. 8, p.1265, Elsevier, 2003.

[ATLAS] R. Clint Whaley and Antoine Petitet and Jack J. Dongarra: *Automated Empirical Optimization of Software and the ATLAS Project*, Parallel Computing 27, 1–2, pp 3–35, 2001.

[10] http://konwihr.in.tum.de/index_e.html

[11] http://www.lrz-muenchen.de/wir/intro/en/#super