Data Structures and High Performance Computing

Syllabus for the TEMPUS-SEE PhD Course

Gundolf Haase*
Institute for Mathematics and Scientific Computing
University of Graz

Dušan Tošić†
Faculty of Mathematics
University of Belgrade

Manfred Liebmann‡
Institute for Mathematics and Scientific Computing
University of Graz

Nov 7 – Dec 2, 2011 in Skopje, Macedonia

*gundolf.haase@uni-graz.at
†dtosic@matf.bg.ac.yu
‡manfred.liebmann@uni-graz.at

1 Goals

This course will introduce into the field of High Performance Computing (HPC) in mathematical areas. The students will acquire specialties of recent and future hardware concepts as well as on supporting software standards. The course work will be organized such that all course topics will be implemented on the appropriate hardware ranging from a single CPU via multiple CPUs to clusters of CPUs and GPUs. The students will be able to adapt research specific code such that they can take advantage of available computer resources. The three main goals of the course consist of

I) Knowledge of the students on algorithms and data structures for HPC and active use of this knowledge,

II) The students get in touch with HPC related concepts and architectures, and the students are able to adopt new developments in this area,

III) Standard compiler and software support for parallel computer architectures is known and used by the students for solving mathematical problems by means of HPC hardware.

Clearly, goals II) and III) require the adjustment of the lecture material with the new developments in these areas.

Overall lecturing time: 60 units with 45 minutes

2 Prerequisites on the student side

1. Knowledge in numerical linear algebra
2. Programming skills in C and/or C++ and/or Java
3. English language skills.

3 Overview of the course modules

The course contains the topics *Algorithms and Data Structures* in relation with *Computer Architectures and Tools* in the HPC area und it is subdivided into three modules of 20 units each:

I) Algorithms and Data Structures
Lecturer: Prof.Dr. DI Gundolf Haase[1], University of Graz, Austria.

II) HPC related Concepts and Architecture
Lecturer: Prof. Ph.D. Marjan Gusev[2], University Sts Ciryl and Methodius, Skopje, Macedonia.

III) Compiler and Software support for parallel computer architectures
Lecturer: Mag.Dr. Manfred Liebmann[3], University of Graz, Austria.

[1] http://www.uni-graz.at/~haasegu
[2] http://twins.ii.edu.mk/marjan
[3] http://math.uni-graz.at

Units	Contents
20	**Module I: Algorithms and Data Structures**
2	Introduce vector, list, stack, queue, tree wrt. parallelism.
2	Complexity of algorithms wrt. data structure, e.g., accessing, sorting.
2	Hashing functions.
2	Realization in C/C++ (STL), C/Java.
4	Object oriented programming (C++, Java).
4	Data structures and performance: complexity, memory hierarchies, and cache aware data structures.
4	Code examples on PCs taking into account cache, vector units of recent CPUs.
20	**Module II: HPC related Concepts and Architectures**
2	The von-Neumann Computer concept. Flynn's Taxiometry (SISD, SIMD, MISD, MIMD).
3	Topologies of computer/processor networks.
3	Concurrency and Correctness (data races, atomic operations, deadlock, live lock).
2	shared memory; semaphores/mutex; distributed memory; hybrid environments.
4	Partitioning; Communications; Synchronization; Data Dependencies; Granularity.
2	Limits and Coast of Parallel Programming. Speedup, weak speedup, efficiency; Amdahl's law; Gustavson's law.
4	Review of recent Multi–core processors.
20	**Module III: Compiler and Software support for parallel computer architectures**
3	Concurrent and distributed programming based on C/C++/Java.
1	Parallel processing based on Open source tools.
4	Parallel processing based on OpenMP for shared memory systems.
5	Parallel processing based on MPI for distributed memory systems.
1	Grid and Cloud computing.
6	Recent parallel programming standards as OpenCL (CUDA).

4 Educational goals of the modules

4.1 Module I)

The students are able to chose the appropriate data structures and algorithms for given problem descriptions with respect to complexity, memory requirements and hardware related issues. This module contains exercises to each item such that the students develop practical skill of the above topics in the chosen programming language. The ideas and concepts will be realized on a PC in C++ or JAVA using available container classes (STL) from these programming languages. Special emphasis will be given to performance of the structures on recent hardware.

4.2 Module II)

The contents of this block are presented partially by the students due to studying the given literature and web material, especially for the last three items in this module. Exercises to data partitioning and simple tasks as reduce operations and data exchange are presented to the students and have to be performed by the students on various logical computer topologies.

4.3 Module III)

This module contains introductory lectures but mainly guided and independent work of the students on the following parallel platforms. At least two of the following platforms should be experienced by each student during the course:

- Multiple core workstation with shared memory using OpenMP.
- Distributed memory computer using MPI (OpenMPI).
- Many–core parallelization with GPUs (and Intel's Larrabee if already available) on basis of CUDA or OpenCL.

The students will be able to write their own mathematical code for the parallel platform chosen.

5 Literature

5.1 Module I)

Literature for algorithms and data structures [1] (supporting material[4]) in C++ [3] (supporting material[5]) and Java [2, 11] (supporting material[6]).
Literature on memory hierarchies [8, §6] and cache [5, §5].

5.2 Module II)

A very good general book for this module is [9].
Literature on taxiometry and network topologies [5, §8], concurrency and correctness [6, §3.1].
Literature on distributed memory [5, §6] and recent multi–core processors provided by LLNL[7].

5.3 Module III)

A good book for basics concepts of this module is [9].
The shared memory programming interface is described in OpenMP[8] and the distributed memory interface MPI is available as book [10] and as web-reference in the OpenMPI[9]–implementation.
As literature for many–core programming we use [7] and the homepages for CUDA[10] and OpenCL[11].

References

[1] Michael T. Goodrich and Roberto Tamassia. *Algorithm Design: Foundations, Analysis and Internet Examples*. Wiley, 2003.

[2] Michael T. Goodrich and Roberto Tamassia. *Data Structures and Algorithms in Java*. Wiley, 4 edition, 2006.

[3] Michael T. Goodrich, Roberto Tamassia, and David M. Mount. *Data Structures and Algorithms in C++*. Wiley, 2003.

[4] S.J. Hartley. *Concurrent Programming – the Java programming language*. Oxford University Press, 1998.

[5] John L. Hennessy and David A. Patterson. *Computer Architecture: A Quantitative Approach*. Morgan Kaufmann Publishers, 3rd edition, 2003.

[6] Maurice Herlihy and Nir Shavit. *The Art of Multiprocessor Programming*. Morgan Kaufmann, 2008.

[4] http://ww3.algorithmdesign.net/
[5] http://cpp.datastructures.net/
[6] http://ww0.java4.datastructures.net/
[7] https://computing.llnl.gov/tutorials
[8] http://openmp.org/wp
[9] http://www.open-mpi.org/
[10] http://developer.nvidia.com/object/gpucomputing.html
[11] http://developer.nvidia.com/object/opencl.html

[7] David Kirk and Wen mei Hwu. *Programming Massively Parallel Processors: A Hands–on Approach*. Elsevier, 2010.

[8] Linda Null and Julia Lobour. *The Essentials of Computer Organization and Architecture*. Jones and Bartlett, 2003.

[9] Thomas Rauber and Gudula Rünger. *Parallel Programming: for Multicore and Cluster Systems*. Springer, Berlin, 2010. ISBN 978–3642048173.

[10] M. Sair, D. Walker, and J.Dongarra. *MPI: Complete Reference*. The MIT Press, 1996.

[11] Laurence T. Yang and Minyi Guo. *High-Perfomance Computing:Paradigm and Infrastructure*. Wiley, 2005.

6 Grading

The Grading bases on the performance of the students with respect to the following items:

A) Smaller exercises for *homework* will be regularly given for a deeper understanding of the presented topics.

B) The students have to collect material on certain lecture related topic s and they have to *present* the results of their online–searches.

C) The students have to finish one or two *projects* containing HPC related programming tasks in each module.

D) A *final exam* at the end of the course, which will be comprehensive.

The final grade is obtained from items A)–D) using the following weights.

homework	10%
present	20%
projects	40%
final exam	30%

7 Prerequisites to the place of lecture

1. Beamer and black/white board for lecturer.
2. One PC with LINUX (and Windows) and a stable, fast internet connection for each student during practical parts. Preferable, WLAN for the student laptops should be available in the lecture rooms as well as in the student's dormitory.
3. Access to a local cluster of PCs or compute server.
4. Access to parallel computers and GPU–server in Graz has to be possible, i.e., internet connection for lecturers and students.