Basics of Parallel Computing

Welcome!
Intro, Remarks, the Formalities,
the General Plan
Q&A

Lecturers:

Prof. Dr. Jesper Larsson Träff

Ass. Prof. Dr. Sascha Hunold

Tutors:

Andrej Kurtovic

Technical support:

Markus Spreitzer

This VU consists of

- Lectures (Thursdays, 13:05-15:00, EI 4 Reithoffer) Mandatory
- Exercises ("Übungsblätter")
- Programming Project
- Self-study

- Read, think, solve, program, experiment, learn…

No classes, no groups:
Exercise/project solutions handed in via TUWEL (groups of 2 allowed), commented/graded offline, plenary feedback

Administration

Sign up via TISS (until 21.3.2019)

Sign off via TISS if you do not want to complete the VU (until 4.4.2019)

Course homepage
www.par.tuwien.ac.at/teaching/2019s/191.114.html

Course material (slides), exercise/project hand-in via TUWEL

Check regularly!

ECTS Breakdown

- Lectures: 1.0 ECTS
- Study: 0.5 ECTS
- Exercises, Project work (implementation, test, benchmarking): 1.5 ECTS

In hours to invest…

- Lectures 11x2h = 22h
- Self-study 18h
- Written exam 8+2h = 10h
- Home exercises 2x2h = 4h
- Projects 21h

Total: 75h = 3 ECTS

Give feedback by the end of the course

Why parallel computing?

Parallel computers are everywhere, every computer scientist ("Informatiker"), every "data scientist" must know something about them:

- Why is that?
- What are they good for?
- What exactly is a parallel computer?
- How can we use them efficiently?
- How do we program them?
- What are their limitations?

… because parallel computing is core computer science

Using computational resources to solve problems efficiently (in theory AND in practice). Parallel computing is computer science with the extra dimension of "parallelism":

- Computer architecture, models
- Algorithms and data structures
- Semantics
- Programming languages, compilers
- Programming, software engineering

A great chance to revisit computer science topics in a new light

Old discipline, but with many challenging, unsolved problems, still lively and highly relevant

Parallel computing at TU Wien (≠ parallel programming)

Using parallel computers efficiently

- Aims, motivation, history ("Moore's law"), basics (time, work, and cost; speed-up; Amdahl's Law; scaling), problems and algorithms

- Shared memory parallel computing
- Concrete language: (p)threads, OpenMP

- Distributed memory parallel computing
- Concrete interface: MPI (Message-Passing Interface)

- Newer architectures, new languages (GPU, CUDA, OpenCL)

Prerequisites

Basics on

- Programming, programming languages (we will use C/C++)
- Algorithms and data structures, asymptotic worst-case analysis of algorithms $O(f(n))$, $\Omega(f(n))$, $\Theta(f(n))$
- Computer architecture
- Operating systems

Detailed plan: Lectures (Mondays)

14.3: Intro, basics (I)
21.3: Principles (II): time, work, cost, speed-up, Amdahl
28.3: Principles (III): Patterns, tasks, examples&algorithms
4.4: Shared-memory systems, (p)threads
11.4: OpenMP
2.5: OpenMP
9.5: Distributed memory systems
16.5: MPI
23.5: MPI
6.6: MPI
17.6: More efficient shared-memory programming

Attendance is mandatory

Detailed plan: Exercises ("Übungsblätter")

Thursday 28.3: First exercise ("Übungsblatt 1")
Thursday 4.4: Hand-in

Thursday 23.5: Second exercise ("Übungsblatt 2")
Monday 3.6: Hand-in

Hand-in dates are fixed, no exceptions

Detailed plan: Programming projects

Thursday 2.5: Project (OpenMP)
Thursday 6.6: Hand-in

Hand-in dates are fixed, no exceptions

Detailed plan: Special exercise

Thursday 21.3: Special exercise ("Übungsblatt 0"), get account on system, submit 4K ssh-key via TUWEL

Mandatory

Immediately after getting notification, try to log in to the system and see that everything is ok. Report problems immediately

Without a working account, the VU cannot be passed

Detailed plan: Written exam ("Klausur")

Wednesday 19.6: One hour, without aids

Date is fixed

No aids (no book, no notes, no mobile, …)

Sign-up later via TISS

Sign off before deadline if you will not take the exam. If not signed off, but absent from exam, course is failed (Grade "5")

Passing the course…

Attendance at the lectures is mandatory. This means: the material for the course is that which is covered in the lecture; if you cannot come to some particular lecture, it's your own responsibility to check up on the material (read the slides, talk to your friends…)

To pass the course, ALL exercises (0,1,2) and the project must be handed in, and the written exam must be passed

Exercises, details

Check-questions and problems based on material from the lectures (effort: a few hours)

Hand-in:
Via TUWEL, will be graded (points) by lecturers/tutors, part of final grade

Both exercises MUST be handed in. In order to pass course, 50% of the points must be reached

Exercises can be done and handed in in groups of two (recommended)

Groups can change from exercise to exercise

Hand-in's:
Via TUWEL, clearly mark hand-in with name(s), matriculation number(s)

For group hand-in's, one upload suffices.

If two solutions are uploaded, they must be identical, if not, weakest one counts. Group's responsibility to make sure at least one solution is handed-in!

Read carefully and follow instructions

Programming project, details

Own implementation of a parallel algorithm for a problem discussed in the lecture in OpenMP

Project can be done and handed in in groups of two (recommended)

Same rules as for the exercises:
Hand-in via TUWEL, project MUST be handed in, and get at least 50% of the points. Group may hand in only one solution, if two, weakest one counts. Group needs not be same as for the exercises

Read carefully and follow instructions. And start early!

Own implementation of a parallel algorithm for a problem discussed in the lecture in. Compile, test, run, experiment, improve…

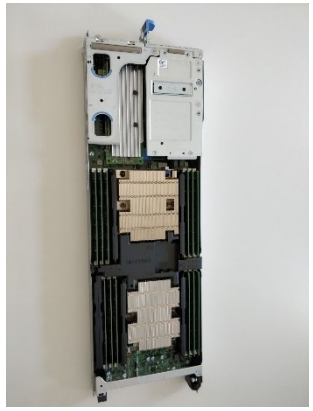
Goal: Correct and efficient implementation (according to expectations)

Hand-in's:
Will be graded by lecturers/tutors. Correctness is important, all programs will be compiled and run against "secret" input, and correctness is a major criteria for the points

System

Programs can be developed at home on own computer (prerequisites: C compiler, OpenMP, MPI), but final test and benchmarking must be done on our (TU Wien) system

- "Hydra": 36-node x 32-core Intel Skylake-OmniPath cluster



Serverroom, Favoritenstr. 9-11

System

Programs can be developed at home on own computer (prerequisites: C compiler, OpenMP, MPI), but final test and benchmarking must be done on our (TU Wien) system

- "Hydra": 36-node x 32-core Intel Skylake-OmniPath cluster

Thursday 21.3: Special exercise ("Übungsblatt 0")
Thursday 28.3: Last chance for submitting key

Mandatory

Project, system

The compute cluster is a rare, limited resource (1 cluster, shared with bachelor lecture, 300+ users). We run it like a compute center…

Use your resources with care, start early with the projects

Final grading ("Note"), how to pass the course

- To pass the course, ALL exercises (0,1,2) and the project must be handed in
- Exercise and project solutions must get at least 50% of the assigned points
- Valid account on systems (special exercise)

Exercises/Project will contribute 60% towards final grade, exam 40% towards final grade

Grade scheme:

1 ("sehr gut"):	90-100%
2 ("gut"):	75-90%
3 ("befriedigend"):	60-75%
4 ("genügend"):	50-60%

Final grading ("Note"), how to pass the course

- To pass the course, ALL exercises (0,1,2) and the project must be handed in
- Exercise and project solutions must get at least 50% of the assigned points
- Valid account on systems (special exercise)

Exercises/Project will contribute 60% towards final grade, exam 40% towards final grade

Do not cheat, do not plagiarize

Solutions and programs can likely be found somewhere, no point in handing such things in. Allowed to exchange ideas with friends and colleagues.

Cheating results in failing grade

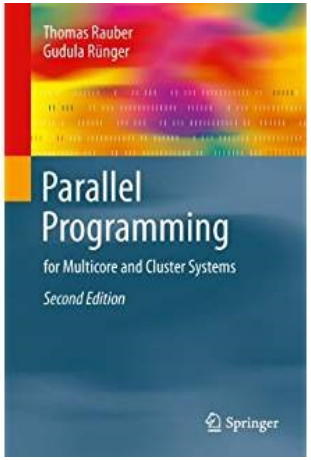
"Wir haben alle Ausreden analysiert.
Schwarzfahren kostet so oder so…"

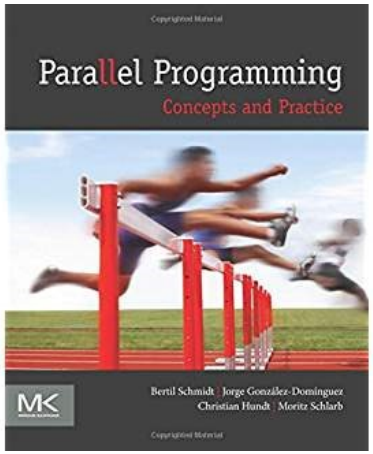
Constructive criticism is always welcome. You can evaluate the course by the end of the semester (TISS)

Literature

- Slides, course material

- T. Rauber, G. Rünger: Parallel Programming for Multicore and Cluster Systems. 2nd Ed., Springer, 2013 (auch auf Deutsch)
- B. Schmidt et al.: Parallel Programming. Concepts and Practice. Morgan-Kaufmann, 2018.

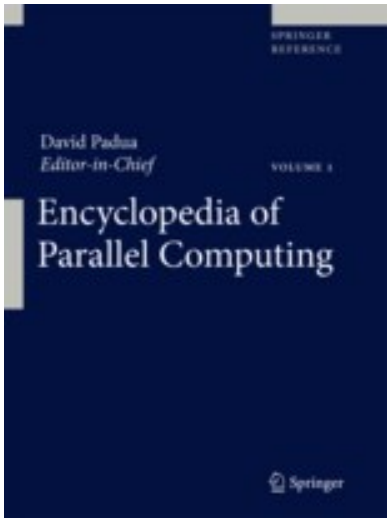


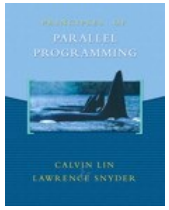


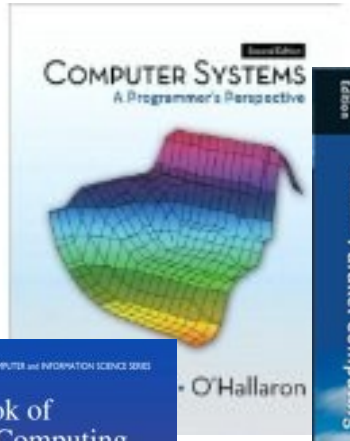
Additional Literature

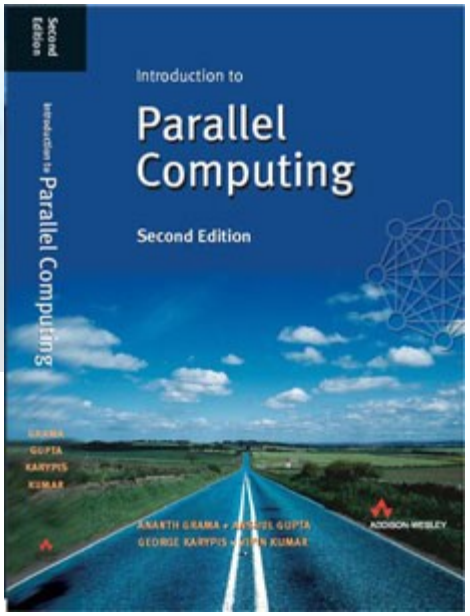
- Grama, Gupta, Karypis, Kumar: Introduction to Parallel Computing. Second edition. Pearson 2003
- Michael J. Quinn: Parallel Programming in C with MPI and OpenMP. McGraw-Hill, 2004
- Calvin Lin, Lawrence Snyder: Principles of parallel programming. Addison-Wesley, 2008
- Peter Pacheco: An introduction to parallel programming. Morgan Kaufmann, 2011

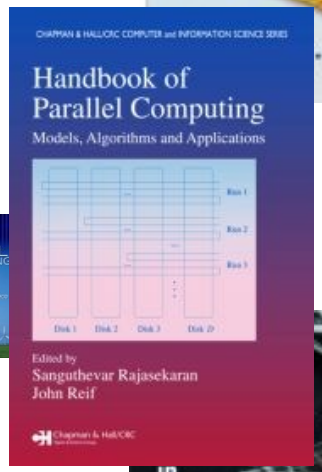
- Randal E. Bryant, David R. O'Hallaron: Computer Systems. Prentice-Hall, 2011

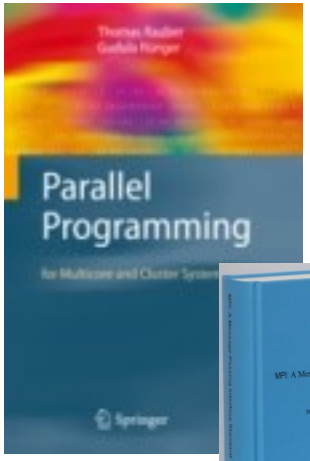


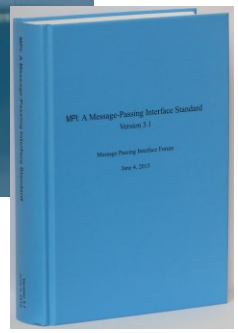


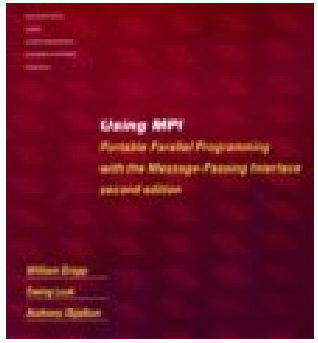


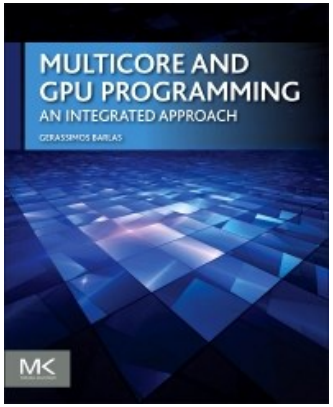


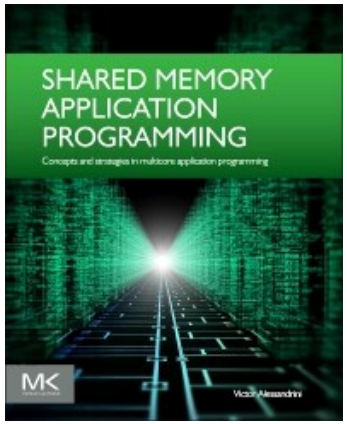


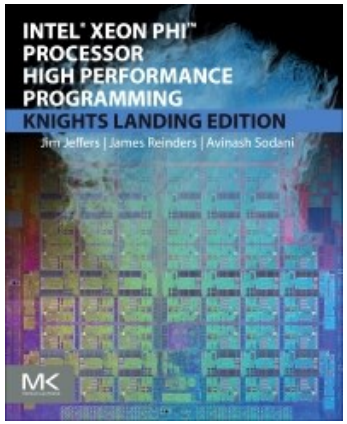


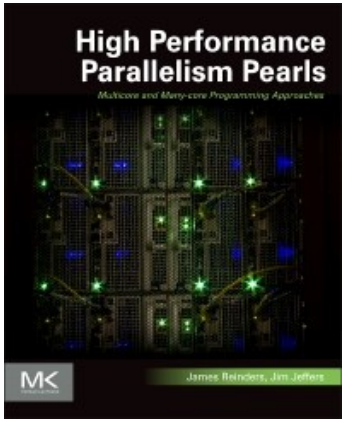


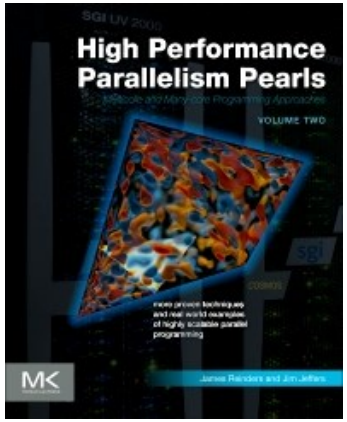


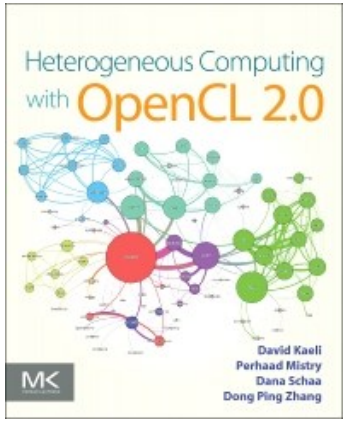

MULTICORE AND
GPU PROGRAMMING
AN INTEGRATED APPROACH
GERASSIMOS BARLAS


SHARED MEMORY
APPLICATION
PROGRAMMING


INTEL® XEON PHI™
PROCESSOR
HIGH PERFORMANCE
PROGRAMMING
KNIGHTS LANDING EDITION
Jim Jeffers | James Reinders | Avinash Sodani


High Performance
Parallelism Pearls
James Reinders, Jim Jeffers


High Performance
Parallelism Pearls
VOLUME TWO


Heterogeneous Computing
with OpenCL 2.0
David Kaeli
Perhaad Mistry
Dana Schaa
Dong Ping Zhang

Follow-up

Bachelor thesis (Träff, Hunold):

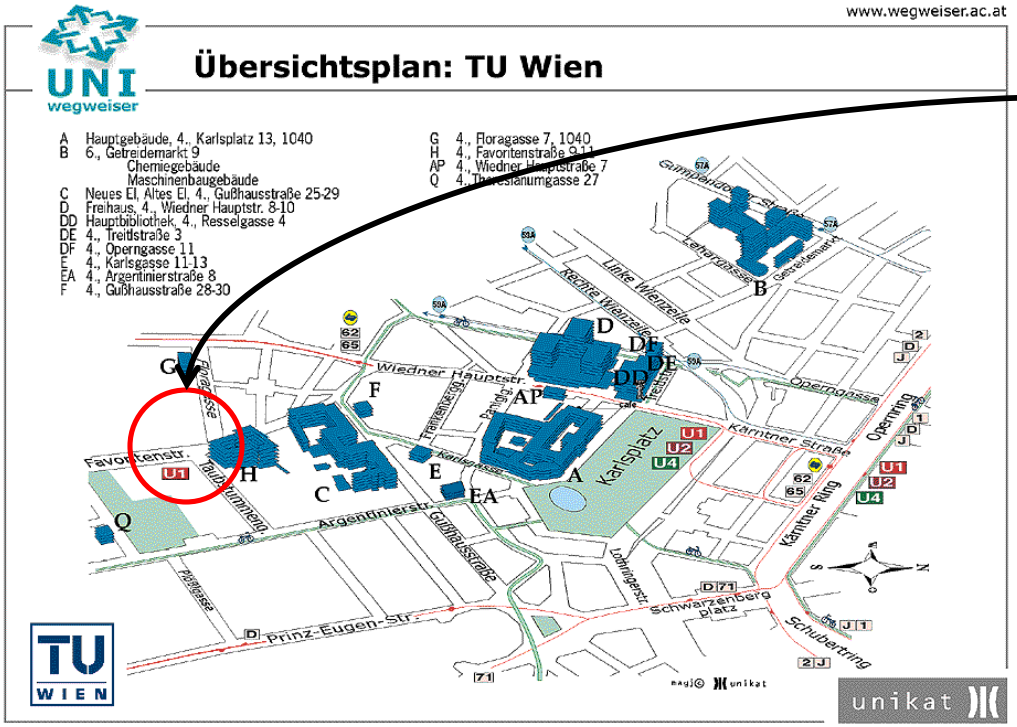
- All aspects of parallel computing (implementations, benchmarking, applications, algorithms, models, …)

Master:

- High Performance Computing, VU, 4.5ECTS
- Advanced Multiprocessor Programming, VU, 4.5 ECTS
- Parallel Algorithms, VU, 3 ECTS
- Projects, 6+6, ECTS
- Seminars in Software Engineering, Algorithms, Theoretical computer science, computer engineering, 3 ECTS
- Master Thesis, 30 ECTS

Research Division Parallel Computing, 191-4 (Träff, Hunold)

See www.par.tuwien.ac.at



Favoritenstrasse 16, 3rd floor

Next to U1, Taubstummengasse, exit Floragasse