Parallel Computing „Einführung in paralleles Rechnen“

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

Lecturers:

Prof. Dr. Jesper Larsson Träff
Ass. Prof. Dr. Sascha Hunold

Tutors:

Marc Ecker
David Fischak
Samuel Pilz
Jakob Pinterits
Clemens Pircher
Hannes Siebenhandl

Technical support:

Markus Spreitzer

General plan

- Mondays 15:15 – 17:00 (Informatik Hörsaal, HERE): Lectures (Träff)
- Tuesdays (some) 15:15 – 17:00 (Informatik Hörsaal): "How to use the systems", Exercise discussion, Project discussion (Hunold)
- Mondays (backup & EXAM) 17:00-19:00 (Informatik Hörsaal)

Mandatory

Administration

Capacity limitation because of resource constraints: 300 students

We will handle this, for now, stay tuned…

Falls Sie im Sommersemester 2019 in der LVA 184.710 „Parallel Computing„ keinen Platz bekommen haben, können Sie stattdessen ein zusätzliches Wahlmodul aus dem Bachelorstudium „Software und Information Engineering“ im Ausmaß von 6 ECTS absolvieren und dieses für Ihren Studienabschluss anstatt des Moduls „Einführung in paralleles Rechnen (Parallel Computing)“ verwenden. Bitte beachten Sie, dass diese Regelung Ihren ehest möglichen Studienabschluss unterstützen soll und zeitlich bis Ende Wintersemester 2019/2020 befristet ist.

Administration

Sign up via TISS (until 11.3.2019)

Sign off via TISS if you do not want to complete the VU (until 18.3.2019)

Course homepage
www.par.tuwien.ac.at/teaching/2019s/184.710.html

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

Check regularly!

This VU consists of

- Lectures
- Exercises ("Übungsblätter") and solution's discussion
- Programming Projects
- Self-study

auf Deutsch

- 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

ECTS Breakdown

- Lectures: 1.5 ECTS
- Study: 1.5 ECTS
- Exercises, Project work (implementation, test, benchmarking): 3 ECTS

In hours to invest…

- Lectures: 13x2h = 26h
- Exercises plenary: 3x2h = 6h
- Self-study: 30h
- Written exam: 10+2h = 12h
- Home exercises: 3x2h = 6h
- Projects: 2x35h = 70h

Total: 150h = 6 ECTS

Give feedback by the end of the course

Why parallel computing?

Parallel computers are everywhere, every computer scientist ("Informatiker") 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, Cilk

- 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)

4.3: Intro, basics (I)
11.3: Principles (II): time, work, cost, speed-up, Amdahl
18.3: Principles (III): Patterns, tasks
25.3: Examples&Algorithms
1.4: Shared-memory systems, (p)threads
8.4: OpenMP
29.4: OpenMP
6.5: Distributed memory systems
13.5: MPI
20.5: MPI
27.5: MPI
3.6: Cilk/OpenMP tasks
17.6: More efficient shared-memory programming

Easter

Whitsun

Attendance is mandatory

Detailed plan: Exercises ("Übungsblätter")

Monday 25.3: First exercise ("Übungsblatt 1")
Tuesday 2.4: Hand-in
Tuesday 9.4: Solutions' presentation (+ "how to use the systems")

Monday 29.4: Second exercise ("Übungsblatt 2")
Tuesday 7.5: Hand-in
Tuesday 14.5 Solutions' presentation

Monday 20.5: Third exercise ("Übungsblatt 3")
Tuesday 28.5: Hand-in
Tuesday 4.6: Solutions' presentation (and Gödel lecture)

Hand-in dates are fixed, no exceptions

Attendance (9.4, 14.5, 4.6) mandatory

Detailed plan: Programming projects

Monday 8.4: Project 1 (OpenMP)
Tuesday 14.5: Hand-in

Monday 13.5: Project 2 (MPI)
Tuesday 4.6: Hand-in

Hand-in dates are fixed, no exceptions

Detailed plan: Special exercise

Monday 18.3: Special exercise ("Übungsblatt 0"), get account on system, submit 4K ssh-key via TUWEL
Monday 25.3: Last chance for submitting key

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: Exam ("Klausur")

Monday 24.6: 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, exercises (0,1,2,3) and projects (1,2) must be handed in and get at least 50% of the points, and the klausur 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

In order to pass course, exercises (and projects) must get at least 50% of the points in total

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 projects, details

Own implementation of a parallel algorithm for a problem discussed in the lecture in OpenMP (Project 1) and MPI (Project 2)

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

Same rules as for the exercises:
Hand-in via TUWEL, the projects (and exercises) must get at least 50% of the points in total. Groups may hand in only one solution, if two, weakest one counts. Groups need not be fixed, can change from project 1 to project 2

Read carefully and follow instructions. And start early!

Own implementation of a parallel algorithm for a problem discussed in the lecture in OpenMP (project 1) and MPI (project 2). Compile, test, run, experiment, improve…

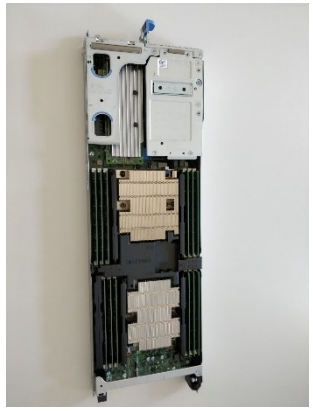
Goal: Correct and efficient implementation (according to expectations)

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

Systems

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) systems

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



Serverroom, Favoritenstr. 9-11

Systems

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) systems

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

Tuesday 9.4: Introduction to the systems, "How to use…" (slurm)

Mandatory

Monday 18.3: Special exercise ("Übungsblatt 0"), get
Monday 25.3: Last chance for submitting key

Projects, systems

The compute cluster is a rare, limited resource (1 cluster, 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, exercises (0,1,2,3) and projects (1,2) must be handed in
- Exercises and project solutions must get at least 50% of the assigned points
- Valid account on systems (special exercise)

Exercises/Projects 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, exercises (0,1,2,3) and projects (1,2) must be handed in.
- Exercises and project solutions must get at least 50% of the assigned points
- Valid account on systems (special exercise)

Exercises/Projects 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

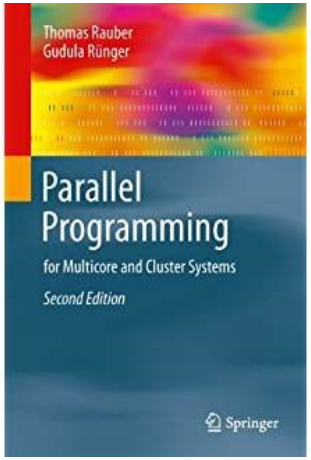
"Wir haben alle Ausreden analysiert.
Schwarzfahren kostet so oder so…"

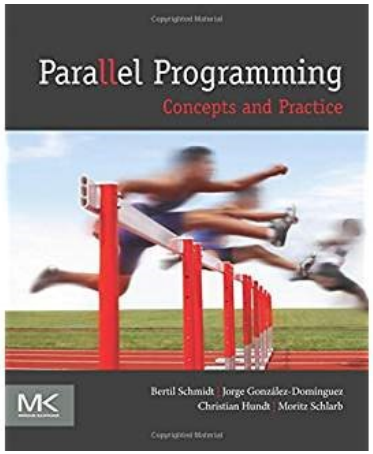
Constructive criticism is always welcome. You can evaluate the course by the end of the semester (TISS)

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.

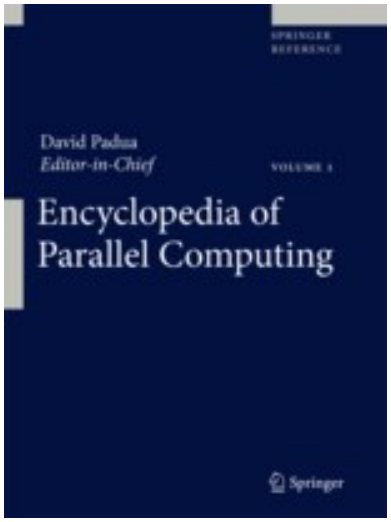


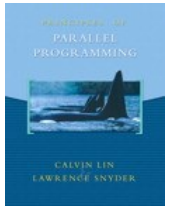


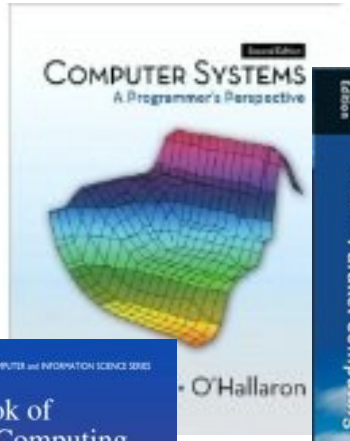
Additional Literature

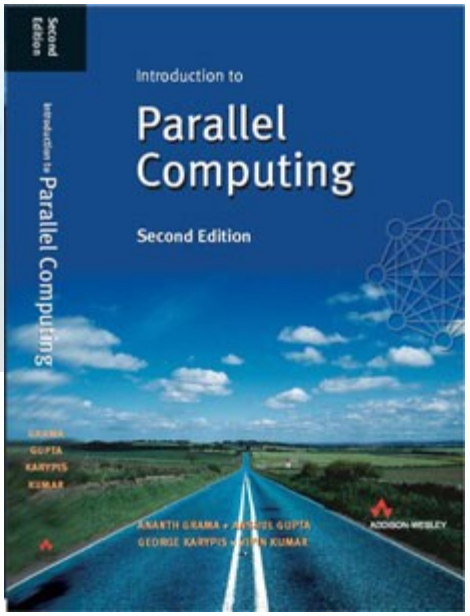
- Grama, Gupta, Karypis, Kumar: Introduction to Parallel Computing. Second edition. Pearson 2003
- Michael J. Quinn: Parallel Programming in C with MPI and OpenMP. McGraw-Hill, 2004
- Calvin Lin, Lawrence Snyder: Principles of parallel programming. Addison-Wesley, 2008
- Peter Pacheco: An introduction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

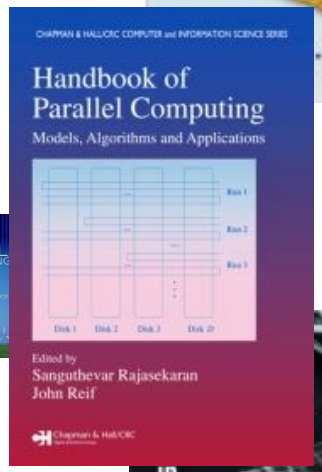
- Randal E. Bryant, David R. O'Hallaron: Computer Systems. Prentice-Hall, 2011

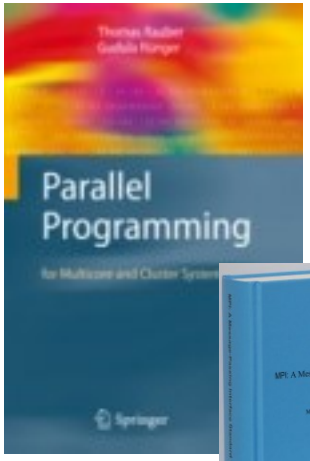


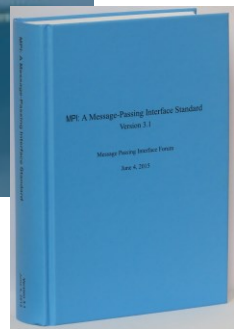


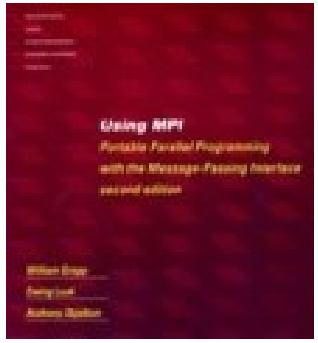


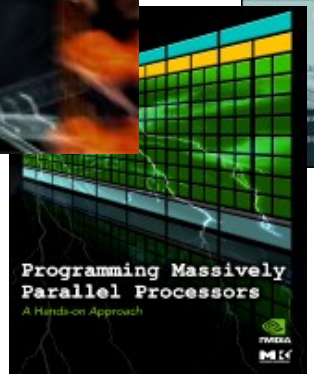


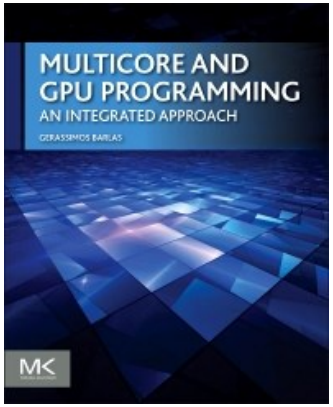


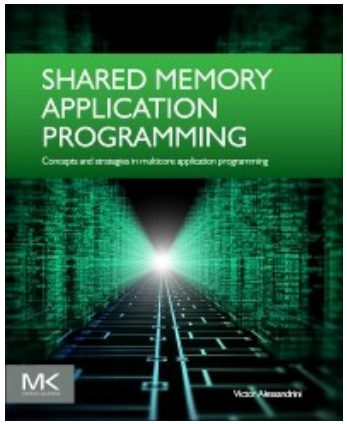


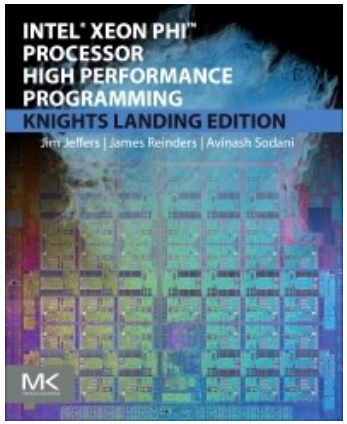


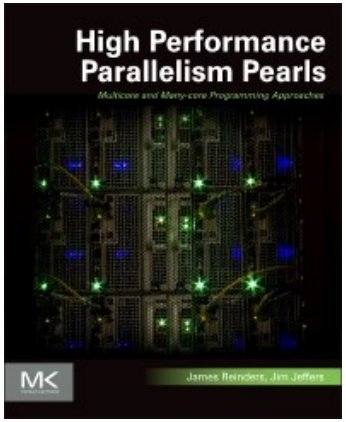


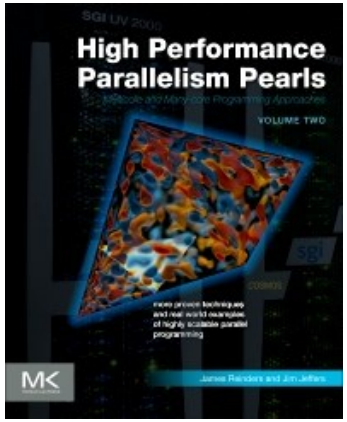


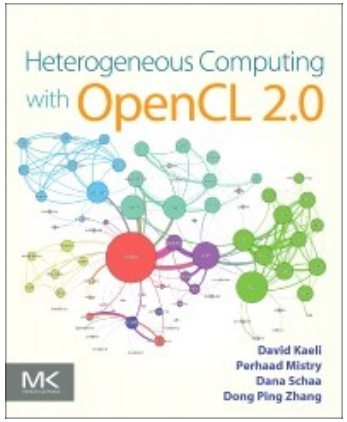

MULTICORE AND
GPU PROGRAMMING
AN INTEGRATED APPROACH
GERASSIMOS BARLAS


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):

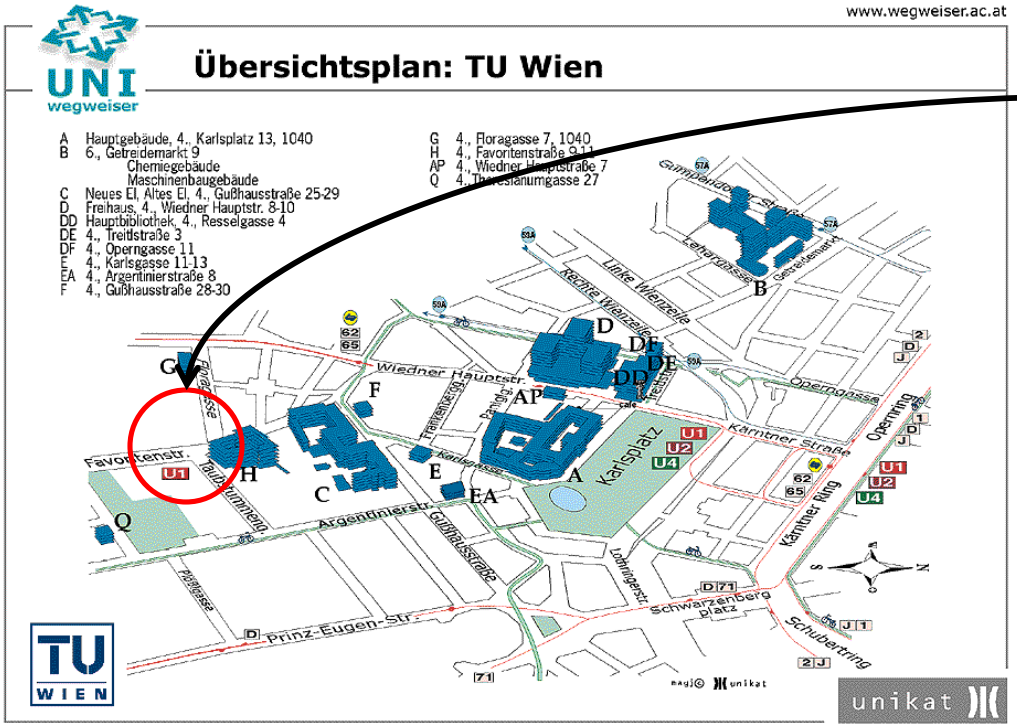
- All aspects of parallel computing (implementations, benchmarking, applications, algorithms, models, …)

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