TECHNISCHE
UNIVERSITÄT
DARMSTADT

System and Parallel Programming

Prof. Dr. Felix Wolf, Sebastian Rinke

ORGANIZATION

General information

Please register for this class via TUCaN

- Makes it easier for us to notify you of changes
- Gives you access to the course material in Moodle

Language

- Slides: English
- Everything else: German

Slides will be available in PDF in Moodle

- Only for the purpose of this class
- Redistribution not permitted

Outline

- Overview
- Student group formation
- Exercise
- Laboratory
- Moodle
- Lichtenberg cluster
- Laboratory for Parallel Programming

TECHNISCHE UNIVERSITÄT DARMSTADT

OVERVIEW

Team

Lecturer

Prof. Dr. Felix Wolf

Organization

Sebastian Rinke

C/C++

Dr. Alexandru Calotoiu

Team (2)

Tutors

- Simon Daniel
- Philipp Geiger
- Jascha Hellwig
- Vladimir Romanenko
- Fabian Rückeshäuser
- Tobias Stensbeck
- Davide Toldo
- Emmanouil Vergopoulos
- Christoph Weckert
- Alexander Ziesing

Authors of exercises

- Arya Mazaheri
- Mohammad Norouzi
- Sergei Shudler

Contents (according to course manual)

- Programming languages for system programming
- Foundations of parallel systems
- Parallel architectures, multi- and manycore systems, clusters
- Programming paradigms and models for parallel computing
- Parallel algorithms
- In-depth study of the course contents via lab assignments of significant extent

Syllabus

- Introduction
- C/C++
- Parallel architectures
- Foundations & analysis of parallel performance
- Parallel programming models
  - Shared-memory programming with OpenMP
  - Message passing with MPI
  - GPU programming with CUDA
- Parallel algorithms & design patterns

Learning objectives

After the successful completion of the course, students should be able to

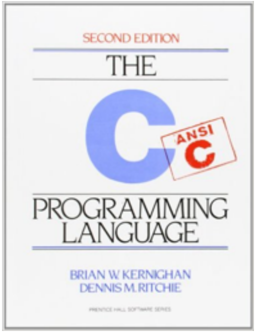
- Understand the foundations of parallel systems
- Program them efficiently
- Develop & analyze simple (parallel) applications via C on selected platforms

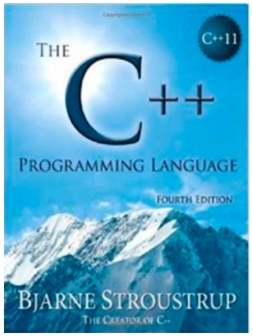
Philosophy

- Physical presence strongly recommended but not monitored
  - No video recording
- No full coverage of programming standards – rather in-depth study of key concepts
- Sound track not always mirrored on slides – please take notes or rely on books for reference

Literature - C/C++

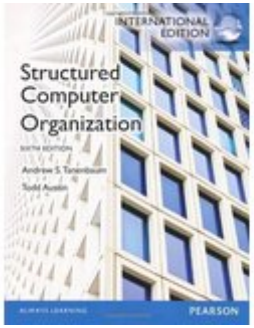
- The C Programming Language
  - Brian W. Kernighan, Dennis M. Ritchie
- The C++ Programming Language
  - Bjarne Stroustrup



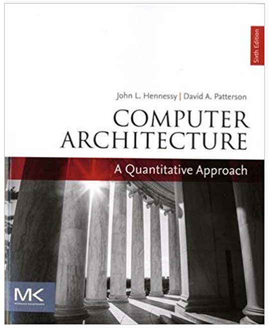


Literature – computer architecture

- Structured Computer Organization
  - Andrew S. Tanenbaum, Todd Austin, Pearson
- Computer Organization and Design
  - David A. Patterson, John L. Hennessy, Morgan Kaufmann
- Computer Architecture – A Quantitative Approach
  - John L. Hennessy, David A. Patterson, Morgan Kaufmann

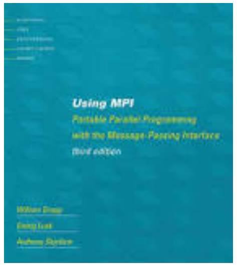


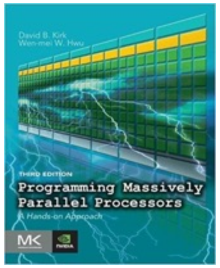
COMPUTER ORGANIZATION AND DESIGN
THE HARDWARE/SOFTWARE INTERFACE
FIFTH EDITION
DAVID A. PATTERSON
JOHN L. HENNESSY
MK



Literature – parallel programming models

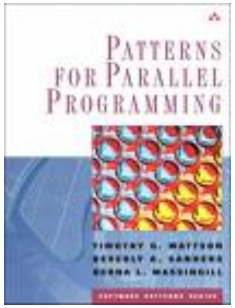
- Using MPI
  - William Gropp, Ewing Lusk, Anthony Skjellum, MIT Press
- Parallel Programming in OpenMP
  - R. Chandra, L. Dagum, D. Kohr, D. Maydan, J. McDonald, R. Meno, Morgan Kaufmann
- Programming Massively Parallel Processors
  - David B. Kirk, Wen-mei W. Hwu, Morgan Kaufmann

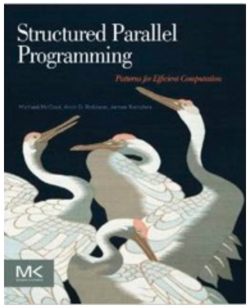

Using MPI
Portable Parallel Programming
with the Message-Passing Interface
third edition
William Gropp
Ewing Lusk
Anthony Skjellum



Literature – parallel algorithms and design patterns

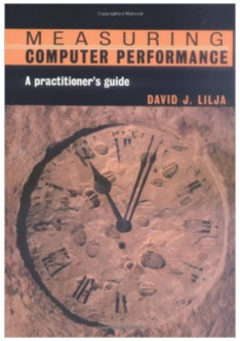
- An Introduction to Parallel Algorithms
  - Joseph JáJá, Addison Wesley
- Patterns for Parallel Programming
  - Timothy G. Mattson, Beverly Sanders, Berna Massingill, Addison-Wesley
- Structured Parallel Programming
  - Michael McCool, Arch D. Robison, James Reinders, Morgan Kaufmann

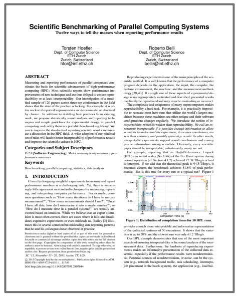




Literature – performance analysis

- Measuring Computer Performance - A Practitioner's Guide
  - David J. Lilja, Cambridge University Press
- Torsten Hoefler, Roberto Belli: Scientific benchmarking of parallel computing systems: twelve ways to tell the masses when reporting performance results
  - Proc. of the SC15 Conference, 2015





Scientific Benchmarking of Parallel Computing Systems
Twelve ways to tell the masses when reporting performance results

Torsten Hoefler
Dept. of Computer Science
ETH Zurich
Zurich, Switzerland
htor@inf.ethz.ch

Roberto Belli
Dept. of Computer Science
ETH Zurich
Zurich, Switzerland
bellir@inf.ethz.ch

ABSTRACT

Measuring and reporting performance of parallel computers constitutes the basis for scientific advancement of high-performance computing (HPC). Most scientific reports show performance improvements of new techniques and are thus obliged to ensure reproducibility or at least interpretability. Our investigation of a stratified sample of 120 papers across three top conferences in the field shows that the state of the practice is lacking. For example, it is often unclear if reported improvements are deterministic or observed by chance. In addition to distilling best practices from existing work, we propose statistically sound analysis and reporting techniques and simple guidelines for experimental design in parallel computing and codify them in a portable benchmarking library. We aim to improve the standards of reporting research results and initiate a discussion in the HPC field. A wide adoption of our minimal set of rules will lead to better interpretability of performance results and improve the scientific culture in HPC.

Categories and Subject Descriptors

D.2.8 [**Software Engineering**]: Metrics—*complexity measures, performance measures*

Keywords

Benchmarking, parallel computing, statistics, data analysis

1. INTRODUCTION

Correctly designing insightful experiments to measure and report performance numbers is a challenging task. Yet, there is surprisingly little agreement on standard techniques for measuring, reporting, and interpreting computer performance. For example, common questions such as "How many iterations do I have to run per measurement?", "How many measurements should I run?", "Once I have all data, how do I summarize it into a single number?", or "How do I measure time in a parallel system?" are usually answered based on intuition. While we believe that an expert's intuition is most often correct, there are cases where it fails and invalidates expensive experiments or even misleads us. Bailey [3] illustrates this in several common but misleading data reporting patterns that he and his colleagues have observed in practice.

Reproducing experiments is one of the main principles of the scientific method. It is well known that the performance of a computer program depends on the application, the input, the compiler, the runtime environment, the machine, and the measurement methodology [20,43]. If a single one of these aspects of *experimental design* is not appropriately motivated and described, presented results can hardly be reproduced and may even be misleading or incorrect.

The complexity and uniqueness of many supercomputers makes reproducibility a hard task. For example, it is practically impossible to recreate most hero-runs that utilize the world's largest machines because these machines are often unique and their software configurations changes regularly. We introduce the notion of *interpretability*, which is weaker than reproducibility. *We call an experiment interpretable if it provides enough information to allow scientists to understand the experiment, draw own conclusions, assess their certainty, and possibly generalize results.* In other words, interpretable experiments support sound conclusions and convey precise information among scientists. Obviously, every scientific paper should be interpretable; unfortunately, many are not.

For example, reporting that an High-Performance Linpack (HPL) run on 64 nodes (N=314k) of the Piz Daint system during normal operation (cf. Section 4.1.2) achieved 77.38 Tflop/s is hard to interpret. If we add that the theoretical peak is 94.5 Tflop/s, it becomes clearer, the benchmark achieves 81.8% of peak performance. But is this true for every run or a typical run? Figure 1 provides a much more interpretable and informative representation of the collected runtimes of 50 executions. It shows that the variation is up to 20% and the slowest run was only 61.2 Tflop/s.

Figure 1: Distribution of completion times for 50 HPL runs.

Our HPL example demonstrates that one of the most important aspects of ensuring interpretability is the sound analysis of the measurement data. Furthermore, the hardness of reproducing experiments makes an informative presentation of the collected data essential, especially if the performance results were nondeterministic. Potential sources of nondeterminism, or *noise*, can be the system (e.g., network background traffic, task scheduling, interrupts, job placement in the batch system), the application (e.g., load bal-

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
SC '15, November 15 - 20, 2015, Austin, TX, USA
© 2015 Copyright held by the owner/author(s). Publication rights licensed to ACM. ISBN 978-1-4503-3723-6/15/11…$15.00
DOI: http://dx.doi.org/10.1145/2807591.2807644

Exam (Studienleistung)

- Exercises
  (3x, each 1/9 of total points)
- Lab assignments & colloquium
  (3x, each 2/9 of total points)

Completed in groups of 3

- Passing the exam requires **at least 50%** of the points.
- Formation of groups via Moodle

Studienleistung
Systemnahe und parallele Programmierung

- Registration procedure for the Studienleistung in TUCaN is the same as for exams
- Studienleistung comprises Teilleistungen (exercises, labs)

- **Register for the Studienleistung until 4.11.18**
    - Registration open until 31.01.19, but registration after 4.11.18 can imply loss of Studienteilleistung submitted before registration
- **Deregistration of Studienleistung possible until 28.10.18**

Studienleistung (2)
Systemnahe und parallele Programmierung

- Information for the following study programs with **examination regulation before WS 2015/16**:
  - B.Sc. Informatik
  - B.Ed. Informatik
  - B.Sc. iST

  Studienleistung Systemnahe und parallele Programmierung graded as passed / not passed

Schedule

Week	# Hours	Lecture	Exercise
42	3	Introduction	
43	3	C	
44	3	C/C++	E1 online
45	3	Parallel architectures	E1 discussion of solutions
46	1 2	Processes & threads, parallel programming models Shared memory programming with OpenMP	E2 online
47	3	Shared memory programming with OpenMP C & OpenMP lab assignment	E2 discussion of solutions
48	1 1	Shared memory programming with OpenMP Parallel performance I	Q&A - C & OpenMP
49	3	Message passing with MPI	Q&A - C & OpenMP
50	3	Parallel performance II MPI lab assignment	Q&A - MPI
51	2 1	Parallel algorithms Parallel design patterns	Q&A - MPI

Schedule

Week	# Hours	Lecture	Exercise
3	3	GPU programming with CUDA	E3 online
4	3	GPU programming with CUDA CUDA lab assignment	E3 discussion of solutions
5			Q&A - CUDA
6			Q&A - CUDA
7			

Lecture time: Tuesday, 13:30 – 16:05h

Lab colloquia start ~3 weeks after assignment

Exercise – time and location

Exercise group	Time	Location	Seats
Ü 01	Mo, 13:30 – 15:10	S103/116	32
Ü 02	Fr, 15:20 – 17:00	S103/175	52
Ü 03	Mo, 13:30 – 15:10	S103/107	40
Ü 04	Mo, 15:20 – 17:00	S103/110	18
Ü 05	Di, 09:50 – 11:30	S105/22	48
Ü 06	Di, 11:40 – 13:20	S215/204K	18
Ü 07	Mo, 17:00 – 18:40	S110/211	46
Ü 08	Fr, 15:20 – 17:00	S110/211	46
Ü 09	Mo, 14:25 – 16:05	S105/23	48
Ü 10	Do, 11:40 – 13:20	S103/100	44
Ü 11	Mi, 17:10 – 18:50	S113/118	40
Ü 12	Do, 18:05 – 19:45	S103/125	24

Detailed schedules

- Course schedule: https://docs.google.com/spreadsheets/d/13XfuHmMYMLM3lQTdRa7gY1wHNYXql9aATbBROHbjZiA/edit?usp=sharing

- Exercise schedule: https://docs.google.com/spreadsheets/d/13kTlNqYSWfknylWsMPbLI7Wt9eLohdLc9jtI2c2a2Z4/edit?usp=sharing

STUDENT GROUP FORMATION

Student group formation

- Students team up in groups of size 3
- Group formation via Moodle until **23.10.18**
  - Find teammates (Moodle: Forum Suche Gruppenpartner)
  - You and your teammates register for same group (Moodle: 3er Gruppe)
  - After deadline only groups of 3 valid (at most one group of size 1-2)
  - All groups of size smaller than 3 deleted & random group formation
- Group assignment is fixed after deadline and cannot be changed
- Contact Sebastian Rinke (rinke@cs.tu-darmstadt.de) if members leave the course and thus the group or do not contribute to the assignments

Student group formation

- Student solves all exercises and labs in the same group
  - One assignment submission per group
  - **All group members must ensure submission of correct version in time**

EXERCISE

Exercise

Three exercises

- C
- Parallel architectures, processes & threads
- Parallel algorithms, patterns, and performance

Exercise

Steps per exercise

1. Task sheet online in Moodle (Monday 13:30)
2. Each group submits one solution within **one week** (next Monday 13:30)
3. Tutor discusses solution after submission deadline in class
4. Tutors grade solutions in Moodle
   - Brief description of mistakes, no correction of errors
   - Grade in points
   - All group members receive same grade

- **Exercises count like an exam**

LABORATORY

Laboratory

Three laboratories

- C & OpenMP
- MPI
- CUDA

Laboratory

Steps per laboratory

1. Task sheet online in Moodle (after presentation in lecture)
2. Empty exercise slots are Q&A slots for the lab
3. Each group submits one solution within **three weeks**
4. After submission deadline, three weeks for *colloquium*
   - Tutor responsible for your group will arrange meeting with you
5. Grade in points for colloquium immediately told and later entered in Moodle together with points for the written solution

Laboratory

Colloquium

- Tutor will arrange time and location with students
- Eleven computers available in CS C-pool
- Students download their solution from Moodle and copy it to cluster
- Students execute each programming solution on cluster
  - Submit job to cluster
- Tutor asks every student additional questions
- Students receive different scores based on answered questions
- Tutor tells where students lost points and their score for the colloquium
- **Labs count like an exam**

MOODLE

Moodle

- The tool for course organization
  - Access granted once registered for course in TUCaN
  - Course schedule + exercise schedule
  - Lecture slides
  - Exercise tasks + submission with deadline
  - Lab tasks + submission with deadline
  - All announcements
  - Forums
- Moodle from Hochschulrechenzentrum
  URL: https://moodle.tu-darmstadt.de/course/view.php?id=13966

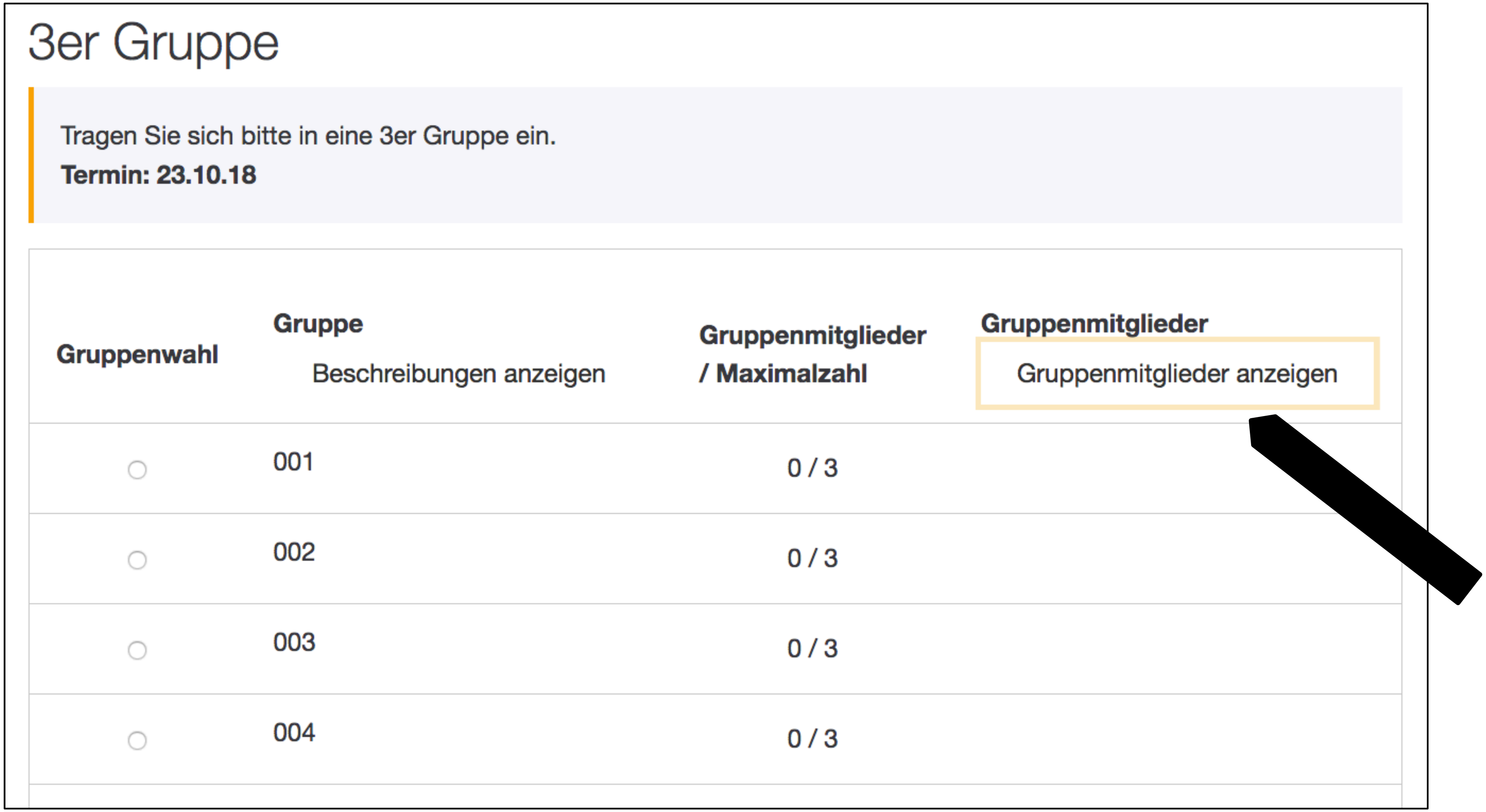
Moodle forums

- Forum Organisation (auto-subscribed)
  - Topics related to course organization (or clarification of exercise & lab tasks)
  - Moderator: Sebastian Rinke
- Forum Praktikum (auto-subscribed)
  - Topics related to contents of the labs
  - No moderation
- Forum Übung (auto-subscribed)
  - Topics related to contents of the exercises
  - No moderation
- Forum Suche Gruppenpartner (auto-subscribed)
  - Search and find team mates
  - No moderation

Moodle group formation

- Register yourself in a group with your team mates
- Collision detection
  - If first member in selected group is not one of your team mates, enroll in other group
  - Always check if someone already registered for the group you want to sign in
- Query team mates in Moodle
- **Deadline: 23.10.18**

Moodle group formation



3er Gruppe

Tragen Sie sich bitte in eine 3er Gruppe ein.
Termin: 23.10.18

Gruppenwahl	Gruppe Beschreibungen anzeigen	Gruppenmitglieder / Maximalzahl	Gruppenmitglieder Gruppenmitglieder anzeigen
○	001	0 / 3	
○	002	0 / 3	
○	003	0 / 3	
○	004	0 / 3	

LICHTENBERG CLUSTER

Lichtenberg cluster

- We have a cluster project for course
  - Three nodes exclusively
  - CUDA hardware later as soon as needed
- Students need to apply for cluster access
- Make yourself familiar with the cluster
  - URL: http://www.hhlr.tu-darmstadt.de

Nutzungsordnung des Hochleistungsrechners der TU Darmstadt für Studierende im Rahmen einer Lehrveranstaltung

1. Präambel

Diese Nutzungsordnung legt fest, nach welchen Regeln der Hochleistungsrechner von Studierenden im Rahmen einer Lehrveranstaltung der TU Darn werden darf.

Der Hochleistungsrechner steht überdies den Wissenschaftlerinnen und Wissenschaftlern der TU Darmstadt und den Wissenschaftlerinnen und Wi anderer Universitäten zur Verfügung. Wissenschaftliches Rechnen ist gestattet, sofern die eingesetzte Software dies erlaubt. Jegliche rein kommerziell untersagt. Bestandteil dieser Nutzungsordnung sind die Bestimmungen in der Allgemeinen Benutzungsordnung für die Informationsverarl Kommunikations-Infrastruktur [1] der TU Darmstadt.

Ein Verstoß gegen diese Nutzungsordnung kann zum Entzug der Nutzungsberechtigung führen.

Alle ausgefüllten Nutzungsanträge (dieses Formular) müssen vom Veranstalter gemeinsam mit dem gesonderten Nutzungsantrag zur Lehrveranstaltu eingereicht werden. Für Fragen stehen wir Ihnen gerne auch per E-Mail: hhlr@hrz.tu-darmstadt.de zur Verfügung.

2. Aufgaben des Hochschulrechenzentrums der TU Darmstadt

2.a. Betrieb des Hochleistungsrechners

Der Hochleistungsrechner wird vom Hochschulrechenzentrum (HRZ) der Technischen Universität Darmstadt betrieben. Zu den Aufgaben des HRZ

- Students need to provide TU-ID in Moodle
  - **Deadline: 23.10.18**
- Students need to hand in signed form (available in Moodle)
  - **Deadline: 23.10. or 30.10.18**

Trouble shooting

- Questions related to lecture contents

  ➔ professor (only during or after class) or tutor

- Questions related to exercise

  ➔ tutor

- Questions related to lab assignment

  ➔ tutor

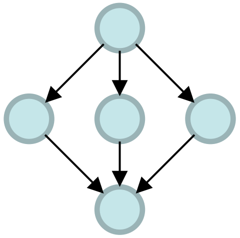
- Questions related to organization

  ➔ Moodle forum Organisation

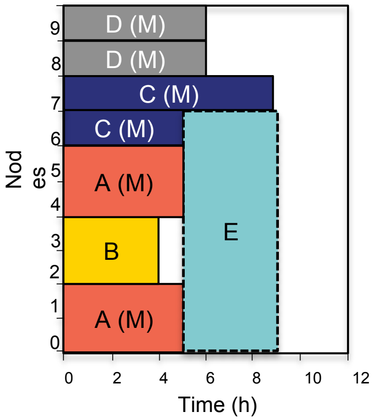
LABORATORY
FOR PARALLEL PROGRAMMING

Laboratory for Parallel Programming

- Parallelism discovery in sequential programs



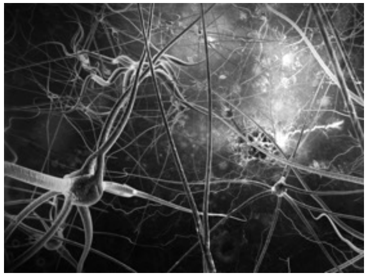
- Dynamic scheduling of cluster resources

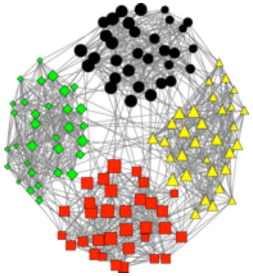


- Generation of performance models for parallel programs

$$f(p) = \sum_{k=1}^{n} c_k \cdot p^{i_k} \cdot \log_2^{j_k}(p)$$

- Scalable algorithms for specific application areas





Further parallel-programming courses

Name	Type	Offered next
Multithreading in C++	Integrated Lecture	SS 2018 (usually every summer)
Advanced Multithreading in C++	Integrated Lecture	WS 2018 / 2019 (usually every winter)
Parallel Programming Technology	Lab	SS 2018 (usually every semester)
Parallel Computing	Seminar	SS 2018 (usually every semester)

GOOD LUCK