Advanced Multiprocessor Programming

Vorbesprechung

Jesper Larsson Träff

Thomas Schlögl, Stephan Felber

traff@par. ...

Research Group Parallel Computing 191-4

Faculty of Informatics, Institute of Computer Engineering

TU Wien

The takeaway

Lectures:
Mondays, 13:00 (c.t)-15:00, EI 5 “Hohenegg”.

Exercises:
Two batches, exercises from the book(s), hand-in 24.4 and 8.5

Exercise Presentations, projects and progress presentation:
Thursdays, after Easter, 11:00 (c.t) – 13:00

Project hand-in 19.6, NO EXTENSION. Machine account 27.3.
Exam: 26.6 to 30.6. Sign up in TISS (May)

Further information, all material: TUWEL (and TISS)

Team

- Jesper Larsson Träff: Lectures, everything
- Thomas Schlögl, Stephan Felber: Exercises, Projects, Exam

- Markus Hinkel: Technical support

The facts and the problems

Modern multi-core processors (2, 4, …, 80 cores + multi/hyper-threading)

- do not really correspond to standard theoretical models (PRAM)
- are very, very difficult to program effectively and efficiently: Performance and correctness

This course:

- Advanced programming techniques in theory (the possible and the impossible) and practice for modern multi-core processors (not GPUs):
- How to implement traditional constructs like locks and barriers efficiently
- How to program without locks and barriers: Data structures and algorithms

Formalities

VU (Lecture-Exercises-Project)

4.5 ECTS (=112.5 hours of work)

Breakdown:

- Lecture 1.5 ECTS
- Exercises 1.0 ECTS
- Programming Project: 2.0 ECTS

Participation MANDATORY, credit given based on Participation, Blackboard Exercises, Programming Project, and Exam

Detailed break-down

- Planning, intro (“Vorbesprechung”): 2h
- Lectures: 15 x 2h = 30h
- Preparation: 15 x 2.5h = 22.5h
- Project/Exercises: 50h
- Exam, including preparation: 8h

Total: 112.5h = 4.5 ECTS

Lecture:

Monday, 13:00 – 15:00, EI 5, "Hohenegg"
Thursdays (from 27.4), 11:00 – 13:00, EI 5, "Hohenegg"

"Sprechstunde" Jesper Larsson Träff, Thomas Schlögl, Stephan Felber: by appointment

Email: traff@par. …, Thomas.e191-02.schloegl@ …
stephan.felber@ …

Sign-up required (deadline 31.3, in TISS)

Sign-out if you don't follow the lecture (before 24. April)

- Theory exercises should be done individually (discussions encouraged…)
- Project in groups of ≤2 members

Now: ≤3

- Get machine account via TUWEL: 27.3 (will be enabled soon, TUWEL exercise to upload 4K public ssh key)

Topics, Goals

Basic understanding of principles and practice of thread-based shared-memory multiprocessor parallel programming

Principles/theory:

- Synchronization and coordination mechanisms
- Scope and limitations
- Correctness: safety and liveness

C/C++, threads, OpenMP, CilkPlus, …

Practice:

- Implementation of basic synchronization mechanisms
- Fundamental (lock- and wait-free) data structures
- Memory models

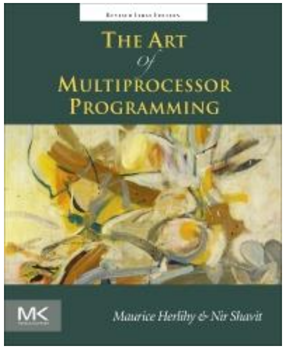
Supporting higher-level shared memory programming models:

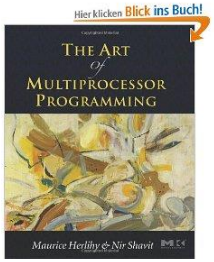
- Task parallel models by work-stealing
- (Transactional memory)

Literature/Material

Book:

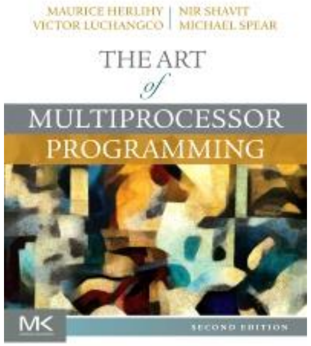
Maurice Herlihy (Brown), Nir Shavit (Tel Aviv): The Art of Multiprocessor Programming. Morgan Kaufmann Publishers, 2008, revised 1st edition, 2012, second edition, 2021 (now with Victor Luchangco and Michael Spear)


THE ART of MULTIPROCESSOR PROGRAMMING
MK
Maurice Herlihy & Nir Shavit


Hier klicken Blick ins Buch!
THE ART of MULTIPROCESSOR PROGRAMMING
Maurice Herlihy & Nir Shavit
MK

Recommended: buy it!

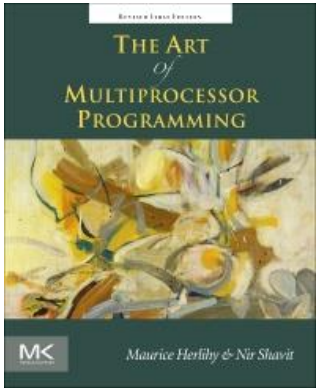
...despite Elsevier


MAURICE HERLIHY | NIR SHAVIT
VICTOR LUCHANGCO | MICHAEL SPEAR
THE ART of MULTIPROCESSOR PROGRAMMING
MK
SECOND EDITION

Lecture slides, additional papers… all on TUWEL)

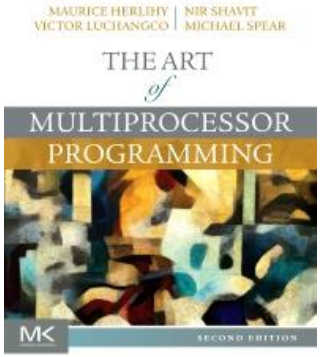
Approx. Coverage

Chapters 1-5 (6), Chapters 7, 9, 10, 11, (12?), 13-16, (17?)
Work-stealing and memory models from other sources


The Art of Multiprocessor Programming
MK
Maurice Herlihy & Nir Shavit

<u>Prerequisites</u>:

- "Introduction to Parallel Computing"
- Algorithms and data structures
- C/C++ (or Java) programming


Maurice Herlihy | Nir Shavit
Victor Luchangco | Michael Spear
The Art of Multiprocessor Programming
MK
Second Edition

<u>Possible follow-up</u>:

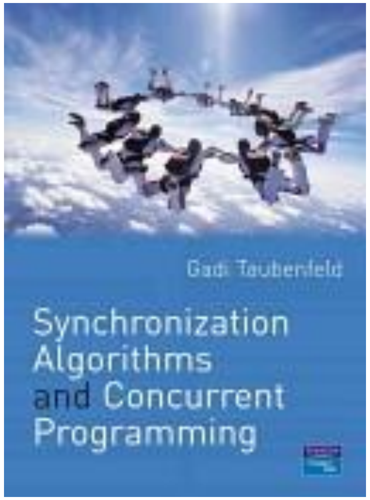
- Parallel Algorithms (PRAM, Scheduling)
- HPC
- Distributed Algorithms (Ulrich Schmid)
- Seminars, Project, Master's thesis

Michel Raynal

Concurrent Programming: Algorithms, Principles, and Foundations

Springer

Michel Raynal:
Concurrent programming: Algorithms, Principles, and Foundations. Springer, 2013

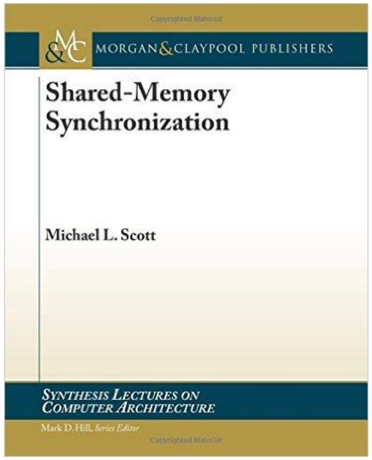


Gadi Taubenfeld

Synchronization Algorithms and Concurrent Programming

Gadi Taubenfeld:
Synchronization Algorithms and Concurrent Programming.
Pearson/Prentice Hall 2006

Synthesis lectures on computer architecture. Morgan&Claypool



Copyrighted Material

MORGAN & CLAYPOOL PUBLISHERS

Shared-Memory Synchronization

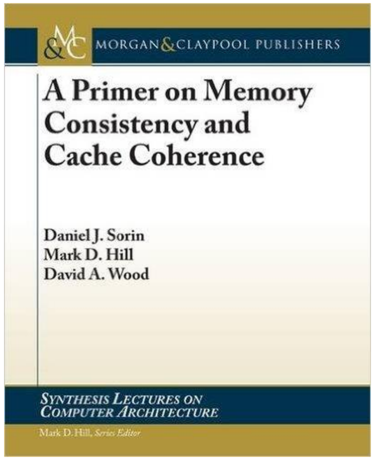
Michael L. Scott

SYNTHESIS LECTURES ON COMPUTER ARCHITECTURE

Mark D. Hill, Series Editor

Copyrighted Material

Michael L. Scott: Shared-Memory Synchronization, 2013



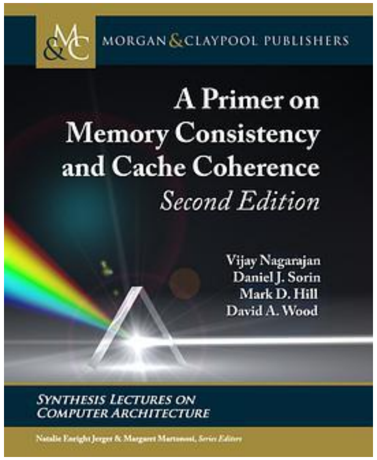
MORGAN & CLAYPOOL PUBLISHERS

A Primer on Memory Consistency and Cache Coherence

Daniel J. Sorin
Mark D. Hill
David A. Wood

SYNTHESIS LECTURES ON COMPUTER ARCHITECTURE

Mark D. Hill, Series Editor



MORGAN & CLAYPOOL PUBLISHERS

A Primer on Memory Consistency and Cache Coherence *Second Edition*

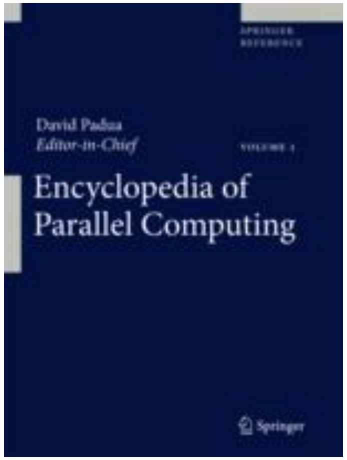
Vijay Nagarajan
Daniel J. Sorin
Mark D. Hill
David A. Wood

SYNTHESIS LECTURES ON COMPUTER ARCHITECTURE

Natalie Enright Jerger & Margaret Martonosi, Series Editors

Daniel J. Sorin, Mark. D. Hill, David A. Wood: A Primer on Memory Consistency and Cache Coherence, 2011, second edition 2020

Parallel computing background (also wikipedia.org)



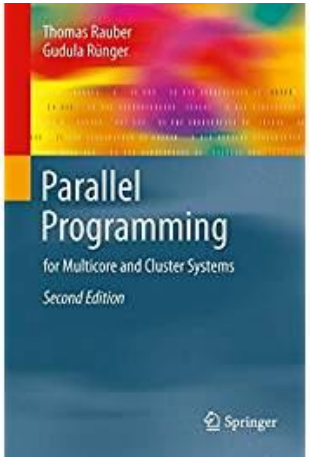
SPRINGER REFERENCE

David Padua
Editor-in-Chief

VOLUME 1

Encyclopedia of Parallel Computing

Springer



Thomas Rauber
Gudula Rünger

Parallel Programming

for Multicore and Cluster Systems

Second Edition

Springer

Exercises/Project

Two slots

"Theoretical" exercises from book, hand-in and discussion/presentation on blackboard

Small programming project:
Implementation and benchmarking (comparison) of lock-free data structure(s) and other material from the lectures

Implementation in C++ threads or C with (p)threads, OpenMP, possibly with CilkPlus (or other C-based framework)

Latex template will be available. Follow instructions on how/what to hand in

Exercises: 2 batches, hand-in

- 27.3 -> 24.4 (Monday):
- 24.4 -> 8.5 (Monday):

Mandatory to pass course (and at least 50% correct)

Project is done in groups of ≤2 (since 10.3: ≤3)

Project:

- 17.4 (Monday): Project topic presentation (by me)
- 24.4 (Monday): Project commit (by you)
- Ca. 1.6: Project status presentation (by you: each group gives a 5-10 minute overview of what it is doing)
- 19.6: Project hand-in (fixed deadline, no extension)

EXAM: From late June (26.6-30.6)

Start early on the project!

System

Possible to start developing on own PC/laptop (no lab access)

Benchmarking/testing: Nebula, new shared-memory node at TUWien

- 2 AMD EPYC 7351P 32-core processors, 2-way hyperthreading, 1.2GHz, total 64 cores, 256G main memory
- Possibly ARM system (Medusa)

Start early on the project!

More later… (get account via TUWEL on 27.3)

Grading/participation

- Attending lectures and exercises (MANDATORY)
- Active participation
- Solving the exercises, presentation on the blackboard (theoretical exercises, hand-in of practical programming exercise, MANDATORY)
- Examination based on project but can cover whole material
- NO ChatGPT

NOTE:

- You only learn by doing exercises and project by yourself.
- Copying/plagiarism will result in grade 5
- Discussion with other groups encouraged, but hand in your own solution

Don't forget: EVALUATE THE COURSE by end of semester (TISS)

Project hand-in:

- Short description of problem, your solution
- Some argument for correctness, testing procedure…
- The required tests/benchmark comparisons (plots, tables)

Both correctness and performance are important!

Grade weighting: ¼ for exercises, ½ for project, ¼ for exam

Start early on the project!

Solving in group:

- Active collaboration, "2*100%", NOT "2*50%"
- Both members get same grade (unless blatantly different)
- Both members must understand all aspects of solutions

Project hand-in:

- Short description of problem, your solution
- Some argument for correctness, testing procedure…
- The required tests/benchmark comparisons (plots, tables)

Both correctness and performance are important!

If done in group, only one hand-in counts (the worst…); groups can hand in two solutions, that should then be identical, or just one (more risky…)

Follow instructions, hand-in via TUWEL

Exam

Oral examination based on project, but can cover whole lecture

Ca. ½ hour

Now: Might be written exam instead

Sign-up in TISS later (group exam or individual to be decided)

If you are signed up for the exam, but do not show up without (or extremely late) notice, grade is 5

Detailed lecture plan (subject to change), Mondays

6.3: "Vorbesprechung". Intro. Mutual Exclusion problem
13.3: Constructions of atomic registers, register snapshot
20.3: Relative power of synchronization operations, correctness conditions
27.3: Relative power of synchronization operations, universality
17.4: Projects (description). Relative power of synchronization operations. Wrap-up.
24.4: Practical lock implementations
(27.4, Thursday, TBA)
8.5: Data structures (I): List-based set
15.5: Data structures (II): Queues and stacks
22.5: Memory consistency models, memory reclamation
5.6: Data structures (III): Skiplist, priority queues
12.6: Data structures (IV): Hash tables
19.6: Barrier synchronization, work-stealing theory (TBA)

Easter: 3.4-14.4
Whitsun: 29.5

1st of May 1.5

Detailed exercise plan (subject to change), Thursdays

27.3 to 24.4 (Monday): Exercise 1, hand-in
24.4 to 8.5 (Monday): Exercise 2, hand-in

4.5 (Thursday): Exercise feedback 1
11.5 (Thursday): Exercise feedback 2

17.4: Project topic presentation
25.5: Project Q&A
1.6: Project presentations, state-of-the-art… (by you)
15.6: Project Q&A

Follow-up

- Projects (6.0+6.0 ECTS)

- Seminar in WS23

- Parallel Algorithms (WS23: VU, 3.0 ECTS)
- High Performance Computing (WS23: VU, 4.5 ECTS)

- Master's Thesis (30.0 ECTS)