Concurrent Programming
Rinnakkaisohjelmointi (RIO)

Teemu Kerola
University of Helsinki
Department of Computer Science

Spring 2012

Connections

Applications
Operating systems
Programming languages
Concurrent Programming (Rio)
- synchronization
- communication
- co-operation
Distributed systems
Performance
Computer Architecture

Course Connections

Computer Organization I
Tietokoneen toiminta

Concurrent Programming
Rinnakkaisohjelmointi

Operating Systems
Adjunttiopasteelit

Distributed Systems
Hajautetut dinsysteemit

Software Design (Java)
Ohjelmointiteknologia (Java)

Network Programming
Verkkosovellusten toteuttaminen

Linux System Admin
Linux-ylläpito

Intro to Specification and Verif.
Spesifioinnin ja verif. perusteet

Concurrent Programming in practice
- Discussed in lectures
- Practice with BACI (1 core) and in project (multicore)

Motivation

- To know and understand …
  - Why concurrency is important
  - What are the concurrency problems in your systems
  - How concurrency problems are usually solved
  - What are the usual tools for solving concurrency problems
  - How concurrency problems may be solved at different system levels
    - HW, OS, progr. language library, application

Learning Goals

- Concurrency and problems caused by concurrency
  - Basics, fundamental ideas, background
- Fundamental concepts and models in concurrency
  - Main emphasis is here!
  - E.g., “can explain special features of semaphores and monitors and use them properly in applications”, or “can explain how deadlocks can be prevented”.
- Concurrent programming in multicore and distributed systems
  - Get taste of it, more in later courses
- Concurrent programming in practice
  - Discussed in lectures
  - Practice with BACI (1 core) and in project (multicore)
Concurrent Programming (RIO) 10.1.2012

Lecture 0: Admin, General

**What Good is Concurrent Programming for?**
- All computer systems are inherently concurrent – one must understand concurrency to understand computer system operation.
- All new desktops are multi-core systems for parallel applications.
- Concurrency allows huge speedups for properly designed systems.
- Concurrency causes complex problems that are not easy to solve without good understanding of concurrency.

**What is Not Covered?**
- How to write efficient code for multicore systems?
- What types of applications are suitable for multicore systems?
- What programming paradigms exist to write efficient code for multicore systems?
- How to write multicore code so that it would run in many multicore systems?
- How to best utilize multicore GPU in your application?
- Programming distributed applications.

**Learning Methods**
- Lecture notes
  - Not perfect for self study – use with text book
- Summary lectures & discussions
- BACI – Ben-Ari Concurrency Interpreter
- Practice problems
- Homeworks
- Project
- Group meetings (practice sessions)
- Course exam

**Lectures**
- Summary lectures
  - Assumed: students have read the text book in advance
  - Basic knowledge of today’s topic
- Discussion items
  - Goal: deeper understanding of today’s topic
  - Students discuss given topics in small groups (2-3 students sitting next to each other)
    - Given topics
    - Student suggestions/questions

**BACI – Ben-Ari Concurrency Interpreter**
- Write concurrent programs in C–
- Compile them to BACI PCode
- Execute them concurrently in BACI
- GUI, debugging environment

**Java**
- How to use Java for Concurrent Programming?
  - Threads
  - Messages
  - Semaphores
  - Monitors
- Lectures
- Project

---

10.1.2012 Copyright Teemu Kerola 2012
Study Circles

- Group work, team work
  - It is better to study in a team than alone
  - Peer student support
  - Study circles formed in the first group meeting
- Student centered learning
  - The student has responsibility on learning
  - Instructor facilitated learning
  - Instructors give good environment for learning
- Team work
  - Solving homework problems independently and then discussing them in study circles and in group meetings
  - Project
  - Any other co-operative work for this course

Creation of Study Circles

- Possibly the largest problem in study circle courses
  - “Ville promised, but did not do and he was not accessible. And then Maija did most of it. This is not right! Boohoo!”
- Study circles are formed in the 1st group meeting
  - Goals should be similar
  - Easy in real life: “you will finish it or …”
  - Think about your goals before the 1st group meeting
    - Do I want to learn a lot, or just pass the course?
    - Will I help others, or just concentrate on my own work?
  - Discuss and agree on common goals before agreeing on forming a study circle
  - Finally, sign the “Study Circle Contract”
- Keep up with your agreement
  - Inform the study circle immediately, if you will not continue
  - Get quickly rid of peer students who do not work as agreed on

Practice Problems

- Practice problems
  - Self evaluation
    - Do them only after you think you know the material
    - Do I understand it now?
  - Use does not directly affect your grade
    - No bookkeeping on material use
    - No credit toward course grade

Homework Problems

- Normal homework problems
  - Learning happens when you solve the problems and discuss them
    - Study topic area first before trying out the problems
    - Reading a complete solution or giving one to peer student is wasting a good problem!
  - Homework problems are discussed at
    - Study circle own meetings before group meetings
    - Group meetings with peer students at the table
  - Affects your grade
    - You get homework points (hwp, lhp) for completed problems
    - Only for those present in group meeting

Group Meetings

- Mark down the problems you have completed
- Assistant will organize you in tables
  - Each table should have a student solution for each problem
  - One or more tables may be in English in English speaking practice session
- Discuss all problems in your own table
  - You should have at least tried to solve all problems beforehand
  - You should understand all solutions at end
  - Other students in your table should understand all solutions at end
  - Correct solutions are available, consult assistant if needed
  - Usually no presentations in front of class
- Discuss additional topics given in solutions paper
  - Open ended discussions, no “correct” answers given
- Common discussion of selected problems/solutions
- Advice students in other tables, if needed

Project

- Team work with Study Circle
- Deeper understanding on practical concurrent programming with Java
  - Other course components needed as background knowledge
  - Affects your grade (20/60 points)
Studying for This Course

- Study weekly topics
  - Read the textbook on the same topics, with a different approach
  - Use lecture slides if you find them useful
  - Attend summary lectures and actively participate in discussions
- Check your learning with self evaluation
  - Do practice problems and homework
  - Participate in study circle
    - Discuss homework
    - Weekly group meeting
    - Continue projects
    - Study circle meet face-to-face or in the web
- Finish project in time
- Study for exam
- Course exam

Evaluation

- Self evaluation
  - Do practice problems after each topic
  - Does not affect your grade
  - Do homework each week
    - Do I understand or not?
    - What is there still to learn and how do I do it??
  - Affects your grade
- Course exam
  - Gives a fixed deadline for learning
  - Covers all topics
    - Topics learned in independent study as well as in study circles using various learning methods
  - Evaluates learning
    - Most of the grade based on this
    - Must reach certain level (50%) to pass the course

Grading

<table>
<thead>
<tr>
<th>Course component</th>
<th>max grade points</th>
</tr>
</thead>
<tbody>
<tr>
<td>6 homeworks (min 1 p)</td>
<td>10 p</td>
</tr>
<tr>
<td>Project (min 1 p)</td>
<td>20 p</td>
</tr>
<tr>
<td>2 course exams (min 15 p)</td>
<td>30 p</td>
</tr>
<tr>
<td>Total (min 30)</td>
<td>60 p</td>
</tr>
</tbody>
</table>

Concurrent Programming and problems caused by concurrency

Fundamental concepts and models in concurrency, Concurrent programming in practice

Course contents

- Lecture 0: Admin
- Lecture 1: Concurrency
- Lecture 2: Concurrency at Progr. Lang. Level
- Lecture 3: Critical Section Problem
- Lecture 4: Verifying Concurrent Programs
- Lecture 5: Deadlocks
- Lecture 6: OS Support for Conc. Semaphores
- Lecture 7: More on semaphores
- Lecture 8: Progr. Lang. Support for Conc. Monitors
- Lecture 9: Conc. Control in Dist. Environment
- Lecture 10: Crit. Sections in Distributed Environment
- Lecture 11: Practical Examples on Conc. Control
- Lecture 12: Current Research, Course Summary
- Project: Java Concurrent Programming

Summary

- Course administration
- Course components and learning methods
- BACI simulator
- Lecture format
- Project

Motto

"It is not good exercise, if you do not sweat"
This is not a marathon!
Top-down approach
- Total some 160 h / 6 cu (107 h / 4 cu) course
- 5 yrs / 300 cu = 1 yr / 60 cu = 1600 h / 60 cu = 26.67 h / 1 cu = 160 h / 6 cu