Parallel Programming: Design of an Overview Class

Christoph von Praun
University of Applied Sciences
Nuremberg, Germany
praun@acm.org

This work was supported by an IBM Innovation Award grant.

Summary

- Design of a 3rd year introductory parallel programming class in the Bachelor curriculum: 'Orientation' class

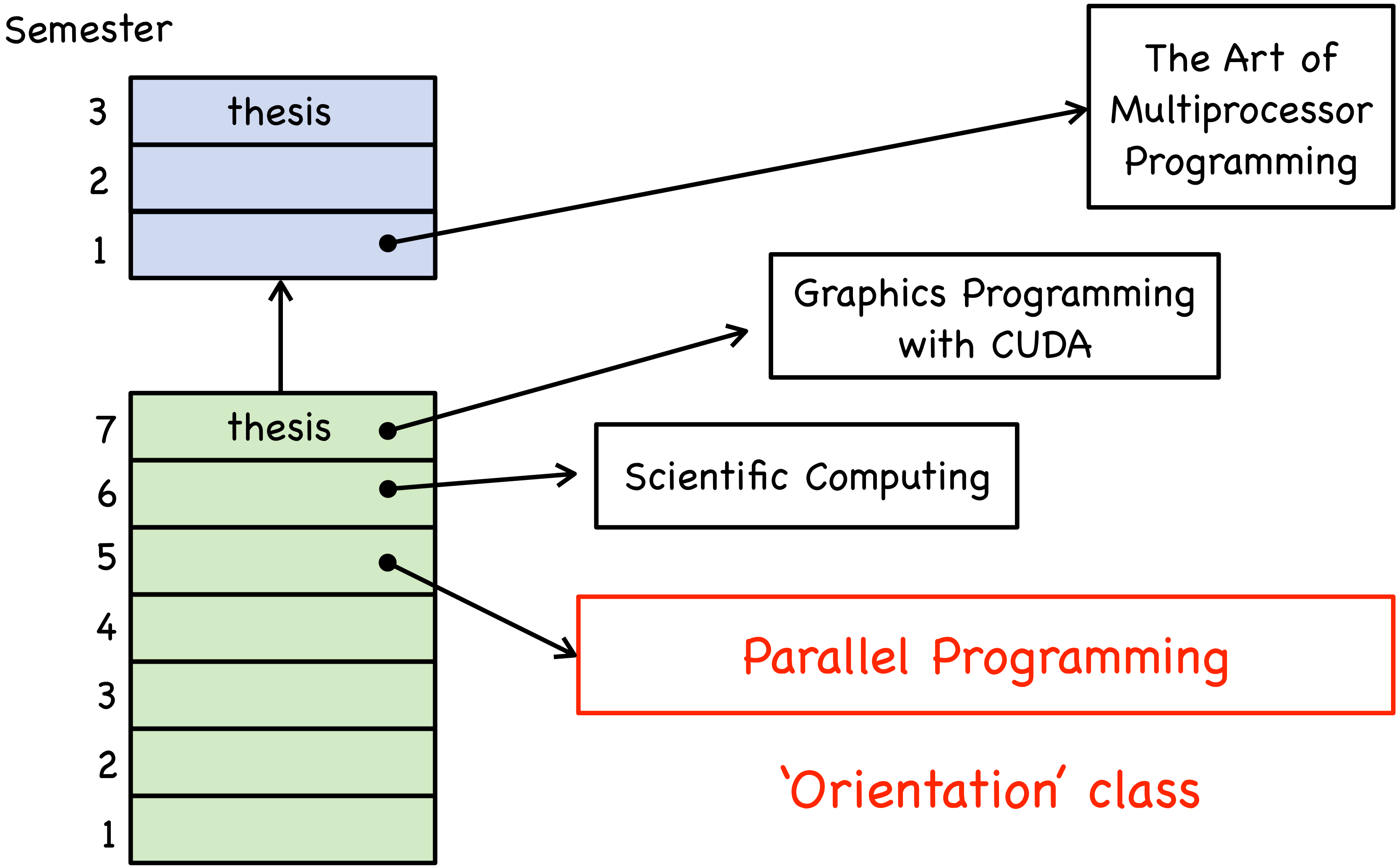
- Key characteristics of the class
  - Organization of topics follows the **Tiers of parallelism**
  - Uses programming language **X10**
  - Strong focus on **lab sessions**

- Teaching materials are available online

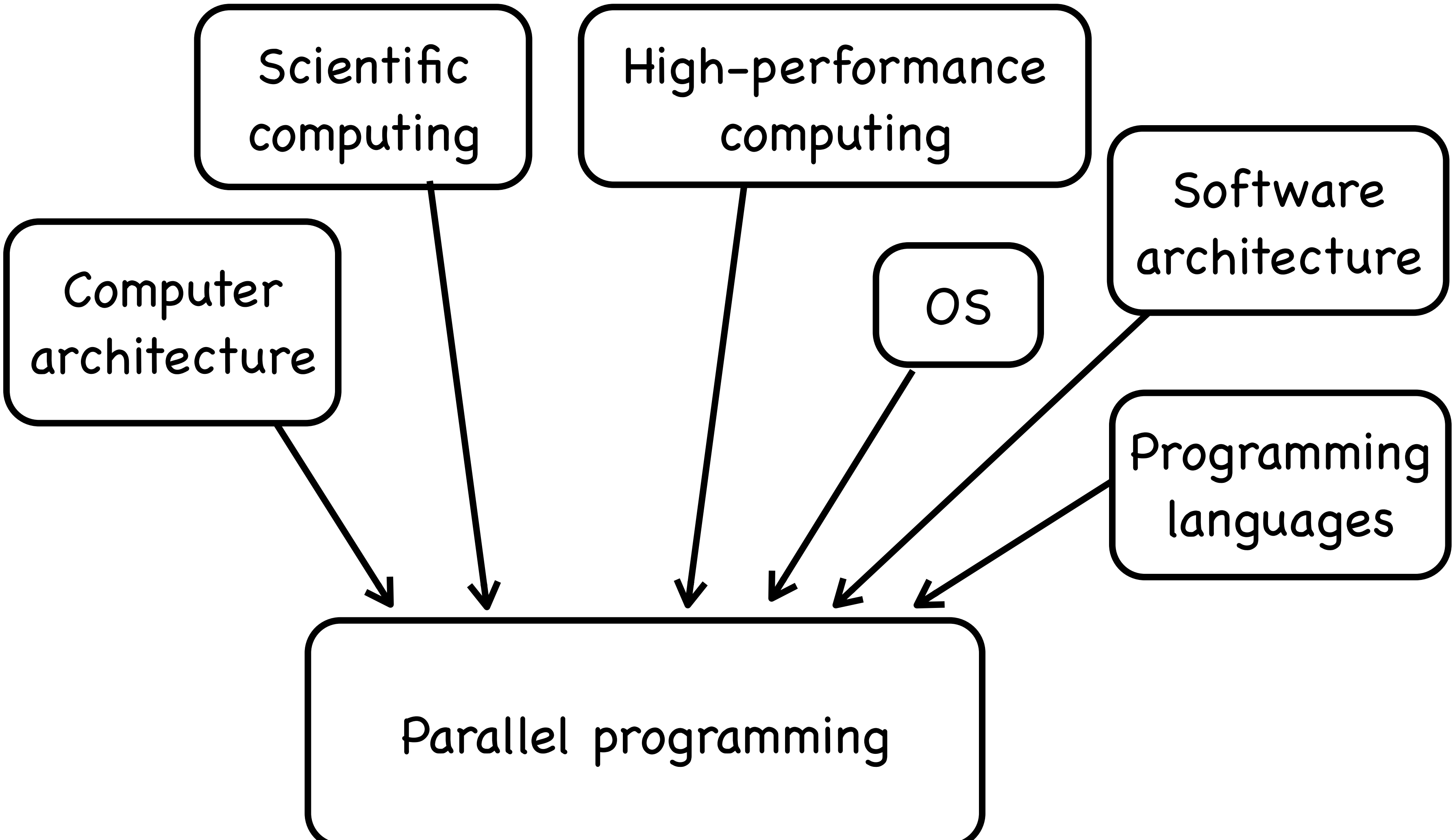
Computer Science

Master + Bachelor Curriculum

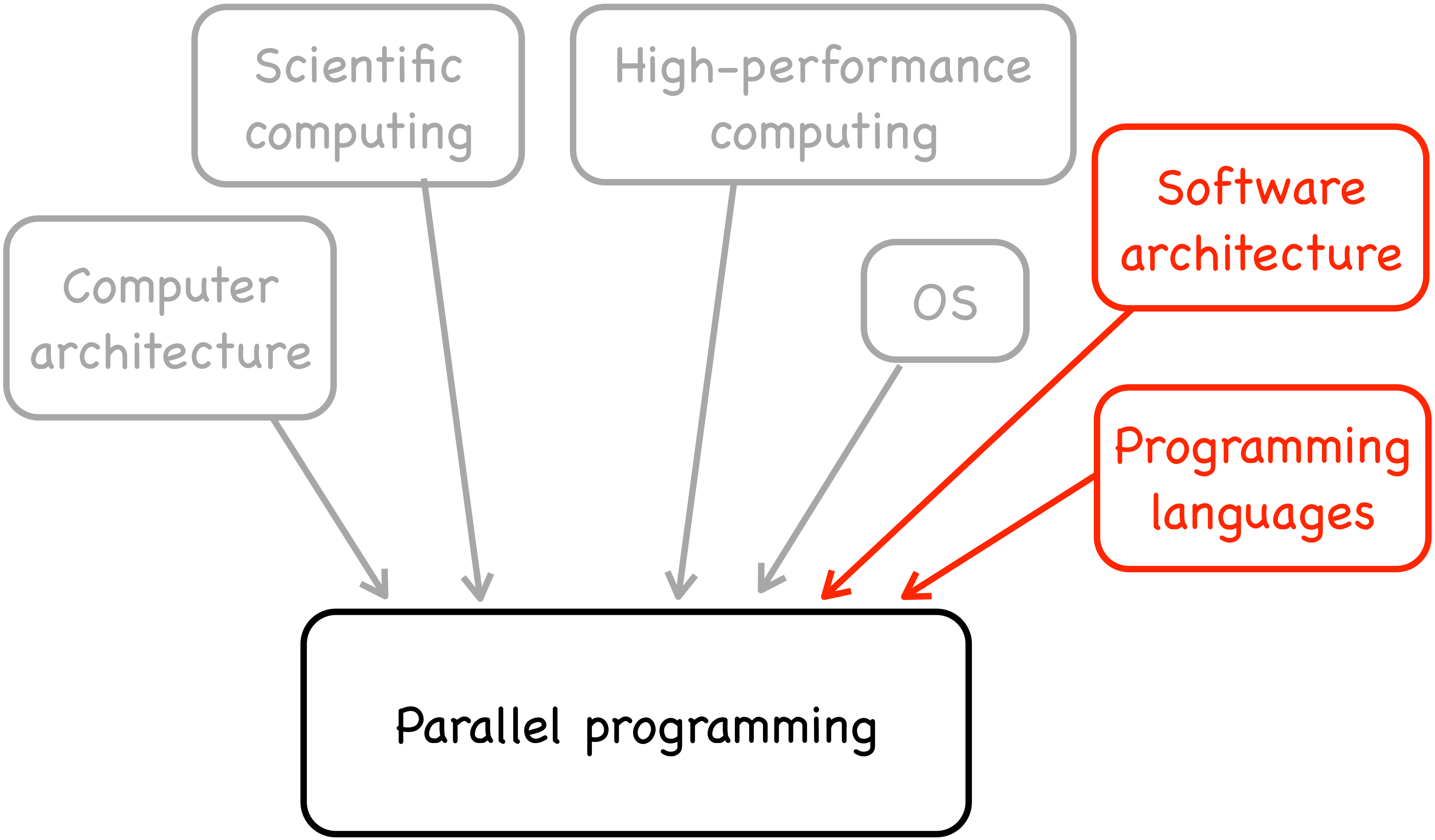
Elective classes



Influences on parallel programming classes



Influences on parallel programming classes



Outline

- **Tiers of parallelism**
- Course structure and contents
- Role of X10
- Student feedback and experience

Tiers of parallelism

- Original idea due to Michael L. Scott:
  - "Don't start with Dekker's algorithm …" [1]
  - "Making the simple case simple" [2]

- Development of parallel software (parallel programming) can be based on techniques at different abstraction layers
  - progressively less complexity at higher abstraction layers

high-level (simpler)

parallelization	**techniques**
(1) automatic or implicit	parallelizing compiler
(2) deterministic	fully independent computations or serialization
(3) explicitly synchronized (data race free)	critical sections, transactions
(4) low-level (with race conditions)	implementation of threads, synchronization mechanisms, non-blocking data structures

low-level (more complex)

parallelization	techniques
(1) automatic or implicit	parallelizing compiler
(2) deterministic	fully independent computations or serialization
(3) explicitly synchronized (data race free)	critical sections, transactions
(4) low-level (with race conditions)	implementation of threads, synchronization mechanisms, non-blocking data structures

parallelization	techniques
(1) automatic or implic	parallelizing compiler
(2) de	tion
(3) ex synch (data	
(4) lo (with race conditions)	s, synchronization mechanisms, non-blocking data structures

Goal of the class:

Students should be conscious about 'their' tier when developing a parallel program.

Encourage students to move programming activity to higher tiers in the abstraction hierarchy.

Tier-1: automatic or implicit parallelism

- **Auto-parallelization** through compilers
- **Parallel kernels**: parallelism encapsulated in libraries
  – LAPACK, etc.
- **Parallel frameworks**: framework organizes parallelism, synchronization and communication, programmer supplies sequential kernels
  – Map-reduce
  – Web application frameworks, e.g. WebSphere
  – etc.

Tier-1: automatic or implicit parallelism

- **Auto-parallelization** through compilers
- **Parallel kernels**: parallelism encapsulated in libraries
  - LAPACK, etc.
- **Parallel frameworks**: framework organizes parallelism, synchronization and communication, programmer supplies sequential kernels
  - Map-reduce
  - Web application frameworks, e.g. WebSphere

Sequential semantics.

Tier-2: deterministic parallelism

- **Independent computations**:
  - parallel array languages (FORALL loops)
  - parallel containers (e.g., STAPL, Intel Concurrent Collections, Hierarchically Tiled Arrays)

- Concurrent computations with dependencies that follow **deterministic idioms**:
  - reduction, scan

Tier-2: deterministic parallelism

- **Independent computations**:
  - parallel array languages (FORALL loops)
  - parallel containers (e.g., STAPL, Intel Concurrent Collections, Hierarchically Tiled Arrays)

- Concurrent computations with dependencies that follow **deterministic idioms**:
  - reduction, scan

Semantics through serialization + sequential reasoning.

Tier-3: explicitly synchronized, data-race-free

Three principal programming models

- **Event-based**
- **Thread-parallel with shared memory**
  - critical sections
  - condition variables
- **Message-based**
  - send/receive
  - collective communication

Tier-3: explicitly synchronized, data-race-free

Three principal programming models

- **Event-based**
- **Thread-parallel with shared memory**
  - critical sections
  - condition variables
- **Message-based**
  - send/receive
  - collective communication

Semantics through interleaving of program blocks

Tier-4: low-level, with race conditions

- Programming with shared memory
  - atomic load and store
  - atomic compare and swap
- Platform-specific (Java, X86, …)
- Sequential consistency is often a simplifying assumption
  - e.g. teaching Dekker's algorithm

Tier-4: low-level, with race conditions

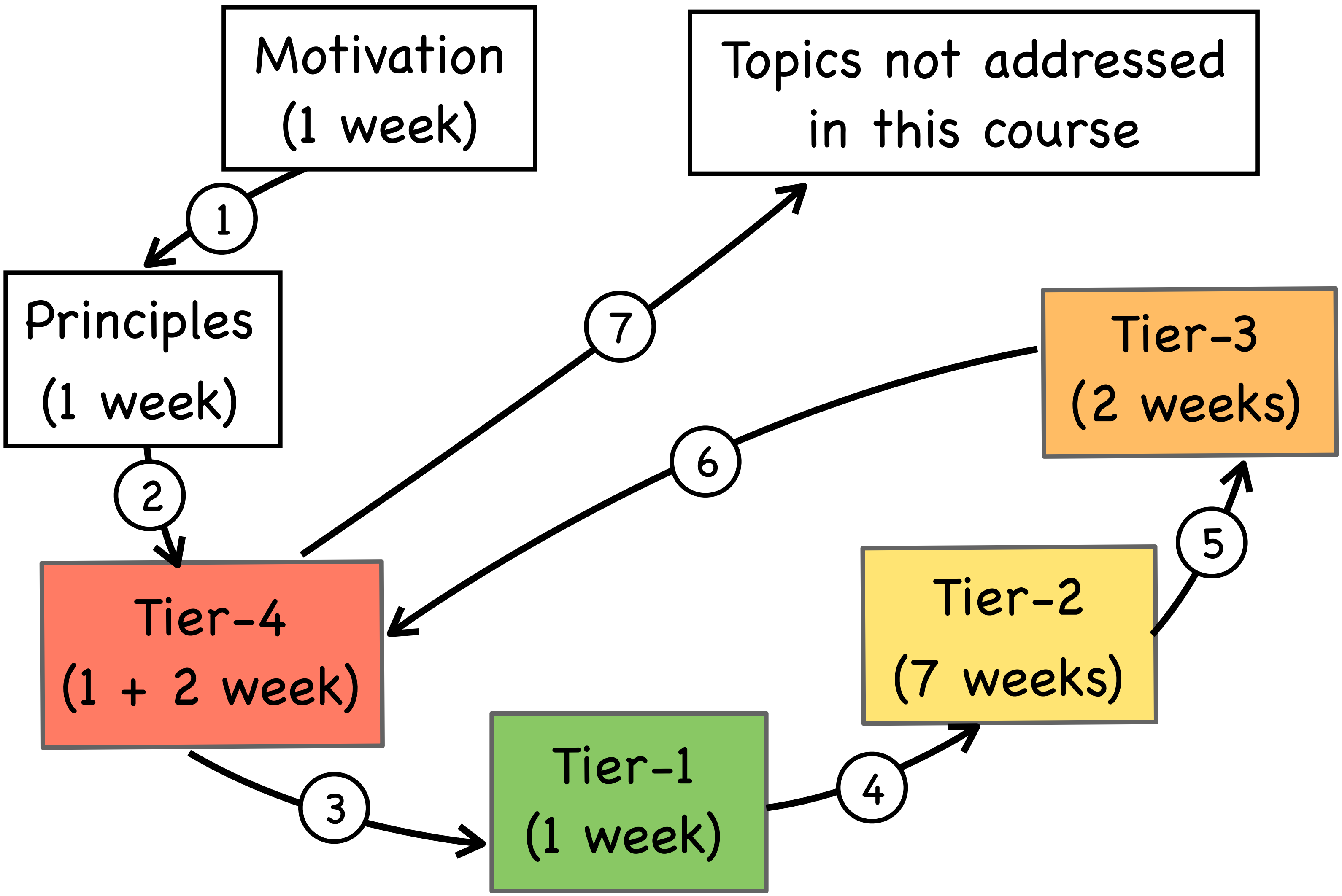
- Programming with shared memory
  - atomic load and store
  - atomic compare and swap
- Platform-specific (Java, X86, …)
- Sequential consistency is often a simplifying assumption
  - e.g. teaching Dekker's algorithm

Semantics through interleaving of statements, possibly not sequentially consistent

Outline

- Tiers of parallelism
- **Course structure and contents**
- Role of X10
- Student feedback and experience

Roadmap of the class (15 weeks)



Motivation (1 week)

- Hardware trend:
  - Moore's Law continues
  - frequency scaling limited by power density: multicores
- Performance: Software need to be parallel
  - challenges (Amdahl's Law)
  - opportunities (Gustafson's Law)
- Energy: Throughput-oriented computing can save energy

Lab session

- Pencil and paper

Principles (1 week)

- Simple model for concurrent computations
  - partial orders of operations
  - synchronization vs ordinary operations
  - happens-before relation
- Explain semantics of X10 language constructs
  - async
  - finish, for-async
  - atomic

Lab session

- Parallel prime number testing

Tier-4 (1 week)
[low-level with race conditions]

- Race conditions
- Non-determinacy
  - associative non-determinism (floating point)
  - atomicity violation: lost-update problem
- “Interleaving” semantics

Lab sessions

- Numeric integration

Tier-1 (1 week)
[automatic or implicit parallelism]

- Challenges of loop parallelization
  - intro to data dependencies
  - difficulties and limitations of dependence analysis on some loop scenarios
- Parallel frameworks
  - Map-Reduce, Web-applications

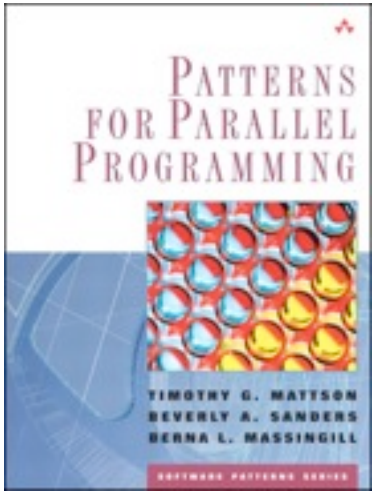
Lab sessions

- Development of Map-Reduce applications (framework provided)

Tier-2 (7 weeks)
[deterministic parallelism]

Patterns for algorithmic problem decomposition:

according to T. Mattson, B. Sanders, B. Massingill, "Patterns for parallel programming", AW 2005.

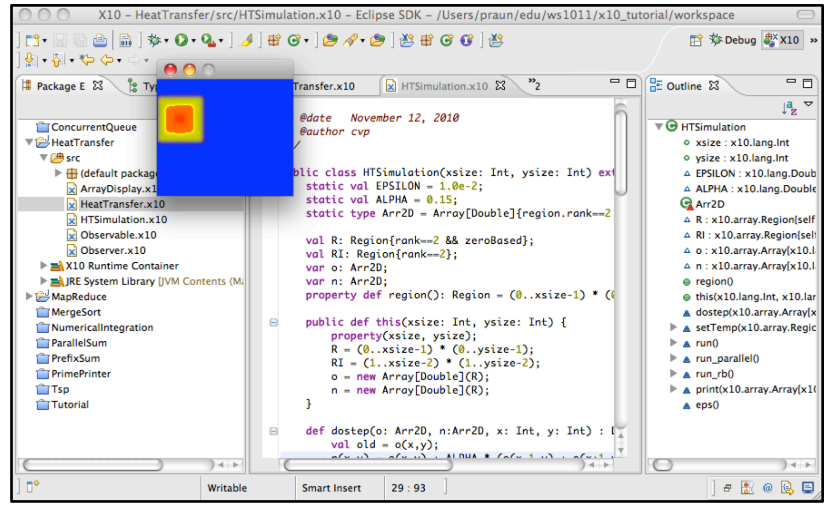


- Data parallelism
  - geometric decomposition, recursive data
  - **data locality issues**
- Task parallelism
  - task parallel, divide and conquer
  - **task scheduling / load balancing issues**

Tier-2 (7 weeks)
[deterministic parallelism]

Lab sessions

- Data parallel
  - heat-transfer
  - matrix multiply
  - algorithms for reduction and prefix-sum



- Task parallel
  - map-reduce framework implementation
  - merge-sort
  - traveling salesman

Tier-3 (2 weeks)
[explicitly synchronized]

- Pattern: Pipeline parallelism
- Producer-consumer communication through concurrent queues
  - critical sections
  - conditional synchronization

Lab sessions

- Array-based concurrent queue with explicit synchronization (atomic blocks)

Tier-4 (2 weeks)
[low-level with race conditions]

- Programming with race conditions
- Memory models (SC, TSO)

Lab sessions

- Lamport's concurrent non-blocking queue
  (1 consumer / 1 produce non-blocking queue)
- Observe non-SC behavior of Java

Topics not addressed in the course

- Patterns for …
  - … locality / reducing data access latency
  - … load balancing / distribution of work
  - … enhancing parallelism
  - … distribution data

- Performance debugging

Outline

- Tiers of parallelism
- Course structure and contents
- **Role of X10**
- Student feedback and experience

X10

- Pragmatic choice:
  - Syntax familiar to students: "extension of sequential Java"
  - Simple things can be expressed with succinct syntax
  - X10 can express programs at tiers (1)-(3)
    memory model not specified -> use Java at tier (4)

- Class was not X10 'only'
  - students could choose their own language for projects
  - X10 language tutorial provided separately

Student feedback on X10

“The language should not be used in future classes , since parallel programming is simplified significantly, and for that reason, one does not run into issues and problems that occur when conventional programming languages are used for parallel programming.”

“Takes a while to be familiar with the type system / type inference.”

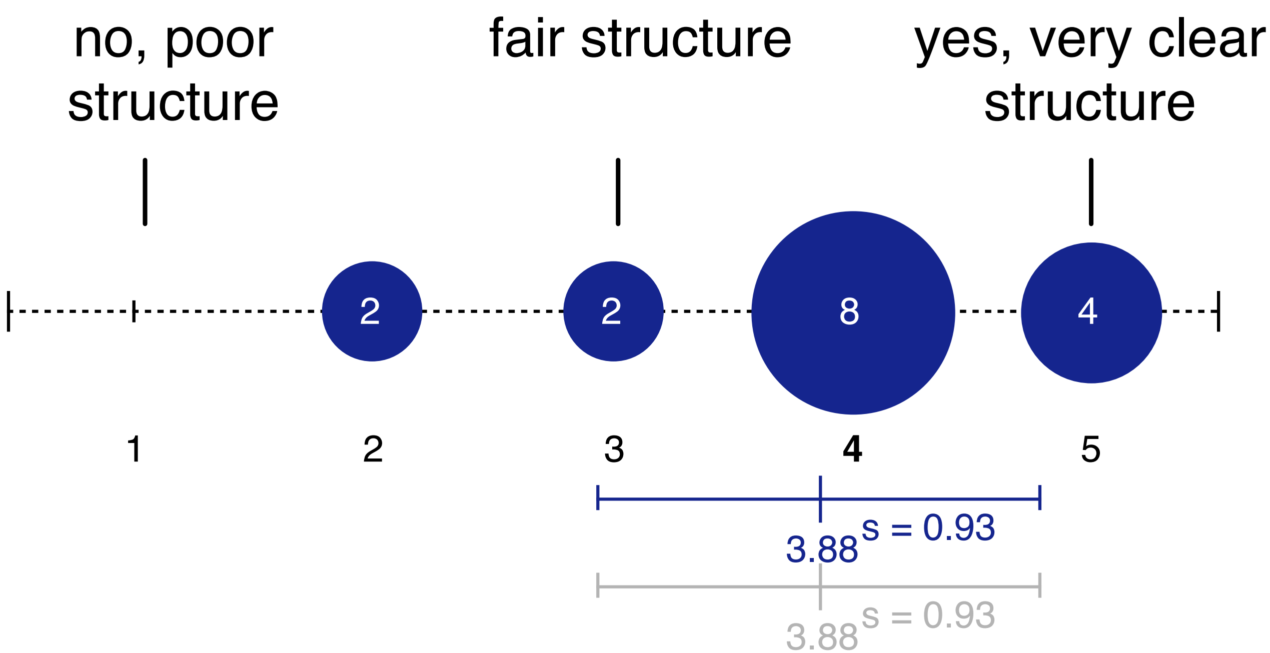
“Usability of X10 IDE needs to be improved” [March-June 2010]

Outline

- Tiers of parallelism
- Course structure and contents
- Role of X10
- **Student feedback and experience**

Student feedback (1/3)

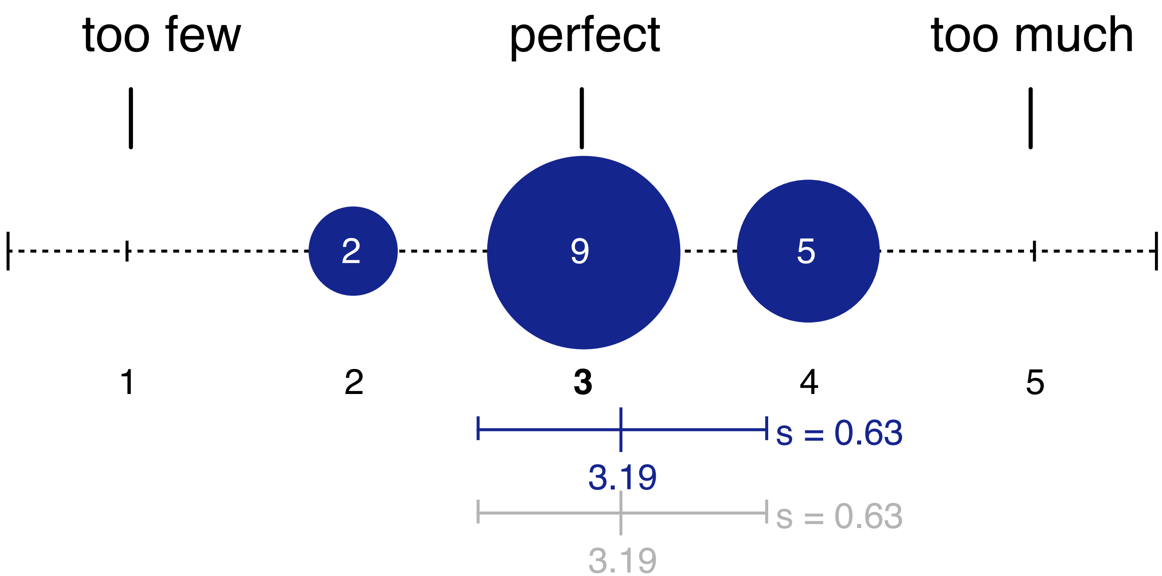
Has the course been well-structured and did the structure support your learning?



Feedback collected from 16/21 participants.

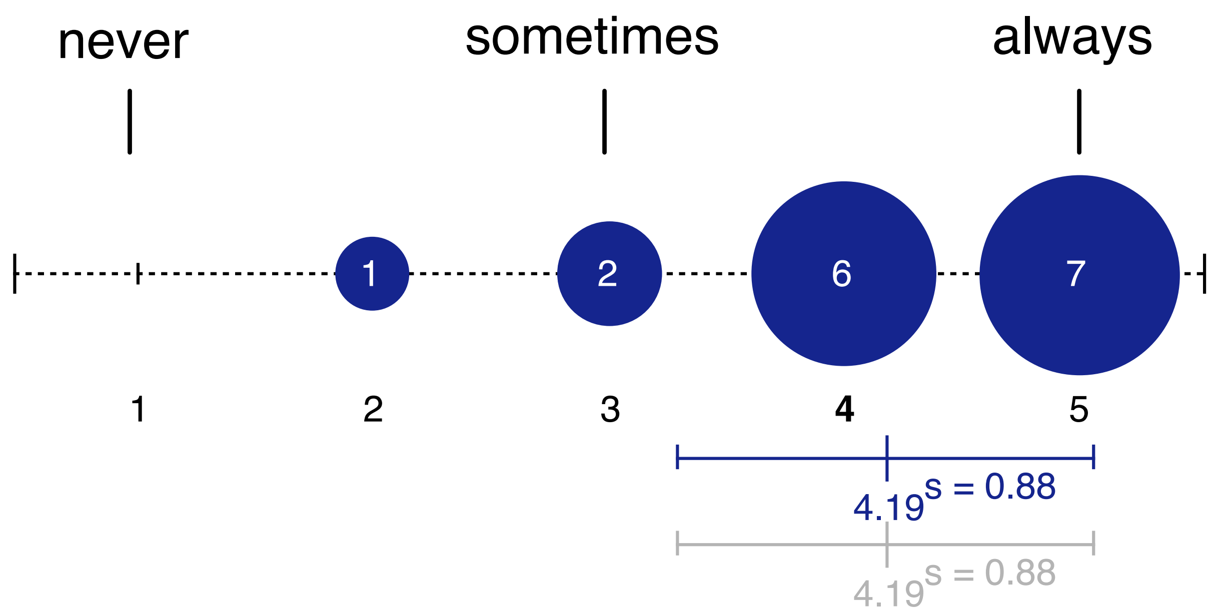
Student feedback (2/3)

The number of topics and the volume of material presented in class was …



Student feedback (3/3)

Did the lab sessions help you to learn and understand the materials presented in class?



Criticism

- Focus of discussion on correctness, not performance
- Focus on 'higher layers' in the abstraction hierarchy
  - Less complex than lower tiers
  - Assumption: People educated in our school are more likely to do parallel programming at higher rather than lower tiers
- Language X10 not widely used in practice

Conclusions

- "Tiers of parallelism" is a fruitful concept
  - course structure
  - orientation for students
- Focus on lab sessions important
  - provided skeletons and solutions for every exercise
  - few students could chose their own language (typically much more complex than X10)
- X10 turned out to be very good choice
  - succinct expression of programs at different tiers
  - steep learning curve

Sources

[1] Michael L. Scott: "Don't start with Dekker's algorithm – top-down introduction to concurrency", Multicore Programming Education Workshop, 2009.

[2] Michael L. Scott: "Making the simple case simple", Position paper, Workshop on Curricula for Concurrency, in conjunction with OOPSLA, 2009.

Thank you for your attention.

Teaching materials are available at
http://www.in.ohm-hochschule.de/professors/praun/pp