Concurrent Systems

Nebenläufige Systeme

I. Introduction

Wolfgang Schröder-Preikschat

October 14, 2015
Outline

Preface

Contents

Organisation

Summary
Abstract Concept

meaning of the lecture labelling in linguistic terms [3]:
con·cur·rent (lat.) concurrens: preposition of concurrere

sys·tems plural of (gr.) systēmas: to place together
meaning of the lecture labelling in linguistic terms [3]:

**con·cur·rent** (lat.) *concurrent*:

1. occurring at the same time; existing together
2. meeting in or going toward the same point; converging
3. acting together; cooperating
4. in agreement; harmonious
5. exercised equally over the same area
Abstract Concept

- meaning of the lecture labelling in linguistic terms [3]:

**systems** plural of (gr.) *systēmas*: to place together

1. a set of arrangements of things so related or connected as to form a unity or organic whole
2. a set of facts, principles, rules, etc. classified or arranged in a regularly, orderly form so as to show a logical plan linking the various parts
3. a method or plan of classification or arrangement

...
Abstract Concept

- in terms of computer science: a system of several computations who are executing simultaneously, potentially interacting with each other
Abstract Concept

- meaning of the lecture labelling in linguistic terms [3]:
  **con·cur·rent** (lat.) *concurrēns*: preposition of *concurrere*
  1. occurring at the same time; existing together
  2. meeting in or going toward the same point; converging
  3. acting together; cooperating
  4. in agreement; harmonious
  5. exercised equally over the same area

**sys·tems** plural of (gr.) *systēmas*: to place together
  1. a set of arrangements of things so related or connected as to form a unity or organic whole
  2. a set of facts, principles, rules, etc. classified or arranged in a regularly, orderly form so as to show a logical plan linking the various parts
  3. a method or plan of classification or arrangement

- in terms of computer science: a system of several computations who are executing simultaneously, potentially interacting with each other
Concurrency as a System Property

- simultaneous execution of potentially interacting computations
  - with the latter being logical (cooperating) or contending (incidental)
Concurrency as a System Property

- simultaneous execution of potentially interacting computations
  - with the latter being logical (cooperating) or contending (incidental)

concurrence in the program flow is due to:

multiplication of processing units
- real parallelism
- instruction set architecture level
- partitioning in space
Concurrency as a System Property

- **Simultaneous execution** of potentially interacting computations
- With the latter being logical (cooperating) or contending (incidental)

Concurrence in the program flow is due to:

- **Multiplication** of processing units, but also
  - Real parallelism
  - Instruction set architecture level
  - Partitioning in space

- **Multiplexing** (partial virtualisation [2])
  - Pseudo-parallelism
  - Operating-system machine level
  - Partitioning in time
Concurrency as a System Property

simultaneous execution of potentially interacting computations

with the latter being logical (cooperating) or contending (incidental)

concurrence in the program flow is due to:

multiplication of processing units, but also

- real parallelism
- instruction set architecture level
- partitioning in space

multiplexing (partial virtualisation [2])

- pseudo-parallelism
- operating-system machine level
- partitioning in time

functionally equal, but non-functionally unequal, characteristics

however, each of the two “concurrency dimensions” originates in different functions to coordinate/synchronise concurrent processes
Concurrency as a System Property

- **simultaneous execution** of potentially interacting computations
  - with the latter being logical (cooperating) or contending (incidental)
- concurrency in the program flow is due to:
  - **multiplication** of processing units, but also
    - real parallelism
    - instruction set architecture level
    - partitioning in space
  - **multiplexing** (partial virtualisation [2])
    - pseudo-parallelism
    - operating-system machine level
    - partitioning in time
- functionally equal, but non-functionally unequal, characteristics
  - however, each of the two “concurrency dimensions” originates in different functions to coordinate/synchronise concurrent processes
- focus is on **parallel processing** of the same non-sequential program
Parallel Processing
Parallel Processing

clustered & symmetric
parallel-computer engineering is pervasive

- multi-core  ■ conventional characteristic
- uni-core   ■ rather unconventional, but rife

by the way: multi-core ⊂ many-core

- multi      ■ little tens ("handful") of cores
- many       ■ several tens of cores and more
  - hundreds or even thousands

exposure to parallelism is indispensable [4]
- mandatory at least for operating systems

28 cores, uniformly distributed across four tiles 😊
Multiplication of Processing Units

parallel-computer engineering is pervasive

multi-core  ■ conventional characteristic
uni-core  ■ rather unconventional, but rife

by the way: multi-core ⊂ many-core

multi  ■ little tens (“handful”) of cores
many  ■ several tens of cores and more
- hundreds or even thousands

exposure to parallelism is indispensable [4]
■ mandatory at least for operating systems

many-core processors make core multiplexing almost superfluous
■ unless latency hiding becomes an issue within a parallel process

28 cores, uniformly distributed across four tiles 😊
Parallel Processor: CPU

AMD, Intel, Tilera

2 cores
Parallel Processor: CPU

2 cores

4 cores
## Parallel Processor: CPU

<table>
<thead>
<tr>
<th>Cores</th>
<th>Image 1</th>
<th>Image 2</th>
<th>Image 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td><img src="image1" alt="2 cores" /></td>
<td><img src="image2" alt="4 cores" /></td>
<td><img src="image3" alt="8 cores" /></td>
</tr>
</tbody>
</table>

AMD, Intel, Tilera
Parallel Processor: CPU

AMD, Intel, Tilera

<table>
<thead>
<tr>
<th>Cores</th>
<th>Image</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 cores</td>
<td><img src="image1" alt="2 cores" /></td>
</tr>
<tr>
<td>4 cores</td>
<td><img src="image2" alt="4 cores" /></td>
</tr>
<tr>
<td>8 cores</td>
<td><img src="image3" alt="8 cores" /></td>
</tr>
<tr>
<td>16 cores</td>
<td><img src="image4" alt="16 cores" /></td>
</tr>
</tbody>
</table>
Parallel Processor: CPU

2 cores  4 cores  8 cores  16 cores  32 cores

AMD, Intel, Tilera
Parallel Processor: CPU

AMD, Intel, Tilera

<table>
<thead>
<tr>
<th>Cores</th>
<th>Image 1</th>
<th>Image 2</th>
<th>Image 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 cores</td>
<td><img src="image1.png" alt="Image 1" /></td>
<td><img src="image2.png" alt="Image 2" /></td>
<td><img src="image3.png" alt="Image 3" /></td>
</tr>
<tr>
<td>4 cores</td>
<td><img src="image4.png" alt="Image 4" /></td>
<td><img src="image5.png" alt="Image 5" /></td>
<td><img src="image6.png" alt="Image 6" /></td>
</tr>
<tr>
<td>8 cores</td>
<td><img src="image7.png" alt="Image 7" /></td>
<td><img src="image8.png" alt="Image 8" /></td>
<td><img src="image9.png" alt="Image 9" /></td>
</tr>
<tr>
<td>16 cores</td>
<td><img src="image10.png" alt="Image 10" /></td>
<td><img src="image11.png" alt="Image 11" /></td>
<td><img src="image12.png" alt="Image 12" /></td>
</tr>
<tr>
<td>48 cores</td>
<td><img src="image13.png" alt="Image 13" /></td>
<td><img src="image14.png" alt="Image 14" /></td>
<td><img src="image15.png" alt="Image 15" /></td>
</tr>
<tr>
<td>32 cores</td>
<td><img src="image16.png" alt="Image 16" /></td>
<td><img src="image17.png" alt="Image 17" /></td>
<td><img src="image18.png" alt="Image 18" /></td>
</tr>
</tbody>
</table>
Parallel Processor: CPU

- 2 cores
- 4 cores
- 8 cores
- 16 cores
- 80 cores
- 48 cores
- 32 cores

AMD, Intel, Tilera
## Parallel Processor: CPU

<table>
<thead>
<tr>
<th>Cores</th>
<th>AMD, Intel, Tilera</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td></td>
</tr>
<tr>
<td>32</td>
<td></td>
</tr>
<tr>
<td>48</td>
<td></td>
</tr>
<tr>
<td>80</td>
<td></td>
</tr>
<tr>
<td>100</td>
<td></td>
</tr>
</tbody>
</table>

© wosch CS (WS 2015, LEC 1) Preface 11
Parallel Processor: GPU

512 cores
Parallel System: HPC

Tianhe-2

3,120,000 cores
Characteristic

- **nature** of the overall processor architecture

- address-space **organisation**

- cache **coherency**: memory **property**

- memory (also: cache) **consistency**: memory **state**
Characteristic

- **nature** of the overall processor architecture
  - **homogeneous**
    - in functional terms: instruction set architecture (ISA)
    - but also non-functional: latency, clock speed, energy use
  - **heterogeneous**
    - different in at least one of those aspects
address-space organisation

shared
- globally direct memory access: load/store operations
- maybe partitioned global address space (PGAS)

distributed
- globally indirect memory access: message passing
Characteristic

- **Parallel Systems**

  - **nature** of the overall processor architecture
    - **homogeneous** in functional terms: instruction set architecture (ISA)
    - **heterogeneous** different in at least one of those aspects
  
  - **address-space organisation**
    - **shared** globally direct memory access: load/store operations
    - **distributed** globally indirect memory access: message passing

- **cache coherency**: memory property
  - **coherent**
    - any read evaluates to the last write to the same address
    - temporary (memory/cache) inconsistencies are tolerated
  - **non-coherent**
    - else
memory (also: cache) **consistency**: memory *state*

- **strict**
  - all accesses are seen in order in which they were issued

- **otherwise**
  - loosened models, differentiate between read and write
  - sequential, processor, weak, entry, or release consistency
**Characteristic**

- **nature** of the overall processor architecture
  - **homogeneous**: in functional terms: instruction set architecture (ISA)
    - but also non-functional: latency, clock speed, energy use
  - **heterogeneous**: different in at least one of those aspects

- **address-space organisation**
  - **shared**: globally direct memory access: load/store operations
    - maybe partitioned global address space (PGAS)
  - **distributed**: globally indirect memory access: message passing

- **cache coherency**: memory *property*
  - **coherent**: any read evaluates to the last write to the same address
    - temporary (memory/cache) inconsistencies are tolerated
  - **non-coherent**: else

- **memory (also: cache) consistency**: memory *state*
  - **strict**: all accesses are seen in order in which they were issued
  - **otherwise**: loosened models, differentiate between read and write
    - sequential, processor, weak, entry, or release consistency
Outline

Preface

Contents

Organisation

Summary
Introduction:

1. overview, organisation—today’s lecture...
Introduction:

1. overview, organisation—today’s lecture.

General topics and basic principles:

2. notion of “concurrency” against the background of resource sharing
   - causality (“cause and effect”), synchronisation, indivisibility
3. notion of “process” and difference to “program”
   - sequential, non-sequential, concurrent, interacting
4. critical (program) sections and their typical patterns
   - race conditions/hazards: lost update, lost wakeup
5. elementary operations and other hardware aspects
   - TAS, CAS, and LL/SC versus caches, coherence, and interference
Synchronisation: Blocking

Pessimistic methods

Classic and folklore:

6. lock algorithms
   - contention, backoff, ticket, interference

7. semaphore
   - binary (vs. “mutex”), general/counting, bolt, set

8. monitor and condition variable
   - signalling semantics: Hansen, Hoare, Mesa, Java

9. deadlock and livelock
   - prevention, avoidance and detection & resolution
Synchronisation: Non-Blocking

Avant-garde and other:

10. algorithms based on indivisible memory-write instructions
   - assuming vertical (stack-like) overlapping
   - interrupt-transparent synchronisation

11. algorithms based on dedicated machine instructions
   - assuming horizontal (congeneric) overlapping
   - compare and swap (CAS), load linked (LL) and store conditional (SC)

12. transactional memory
   - AMD’s advanced synchronisation facility (ASF)
   - Intel’s transactional synchronisation extensions (TSX)

13. progress guarantees
   - obstruction-, lock- and wait-free behaviour
   - constructive (favoured) and analytical (neglected) approaches
State of the art and recapitulation:

14. right from the rummage table...
   - software combining, procedure chaining, combining funnels
   - read-copy update, remote-core locking

15. wrap-up and words in a personal matter
   - retrospection and lessons learned
   - research projects on these topics at the chair
   - perspectives for advanced training: bachelor, master, doctoral thesis
State of the art and recapitulation:

14. right from the rummage table...
   - software combining, procedure chaining, combining funnels
   - read-copy update, remote-core locking

15. wrap-up and words in a personal matter
   - retrospection and lessons learned
   - research projects on these topics at the chair
   - perspectives for advanced training: bachelor, master, doctoral thesis

Hint (Lecture)

Main objective is to impart knowledge on concurrent systems from the system programming point of view. Wide emphasis is on the internals of synchronisation concepts and primitives as well as the implications of the respective implementations. Application of these methods for parallel programming takes a back seat.
depends on the German linguistic abilities of the participants
Language of Instruction

depends on the German linguistic abilities of the participants

**German**
- if all attendees do agree on a German-speaking class
- will be asked for at the beginning of each lesson

**English**
- if at least one attendee does not agree on German

© wosch  CS (WS 2015, LEC 1)  Organisation
Language of Instruction

- depends on the German linguistic abilities of the participants
  - German: if all attendees do agree on a German-speaking class
    - will be asked for at the beginning of each lesson
  - English: if at least one attendee does not agree on German
- in case of doubt or missing answer, German is fallback position

1 Studying abroad also means *living* abroad—and to take part and share in Franconian social life. The latter *soft skills* cannot be overestimated.
Language of Instruction

depends on the German linguistic abilities of the participants

- German  ■ if all attendees do agree on a German-speaking class
  ■ will be asked for at the beginning of each lesson
- English ■ if at least one attendee does not agree on German

in case of doubt or missing answer, German is fallback position\(^1\)

written material (slides or handouts, resp.) will be English
- with technical terms also stated in German, where applicable

\(^1\)Studying abroad also means *living* abroad—and to take part and share in Franconian social life. The latter *soft skills* cannot be overestimated.
Lecture

Meaningful Learning

- acquire new knowledge
- relate it with previous knowledges
Lecture

Meaningful Learning

- acquire new knowledge
  - prepare next reading on one's own initiative
  - attend presentation, listen, and discuss topics treated
  - reinforce learning matter, reflect
Meaningful Learning

- relate it with previous knowledges
  - parallel programming (PFP)
  - computer architecture (GRA)
  - system programming (SP, SPiC, GSPiC)
  - operating systems (BS), operating-systems engineering (BST)
  - real-time systems (EZS)
Lecture

Meaningful Learning

- acquire new knowledge
  - prepare next reading on one's own initiative
  - attend presentation, listen, and discuss topics treated
  - reinforce learning matter, reflect

- relate it with previous knowledges
  - parallel programming (PFP)
  - computer architecture (GRA)
  - system programming (SP, SPIc, GSPiC)
  - operating systems (BS), operating-systems engineering (BST)
  - real-time systems (EZS)

- teaching material presented in the lecture room:
  - follow “Lehre” (Eng. teaching) at https://www4.cs.fau.de
  - copies of the slides are made available as handouts free of charge
  - supplemented by secondary literature as and when required
    - see the bibliography at the bottom of each handout

© wosch  CS (WS 2015, LEC 1)  Organisation  22
deepen knowledge by means of direct experience

Acquisition of virtuous behaviour and operational ability is less a matter of easy instruction but rather functional copy, practise, and use. (Aristotle [1])
Exercise

- deepen knowledge by means of direct experience

*Acquisition of virtuous behaviour and operational ability is less a matter of easy instruction but rather functional copy, practise, and use. (Aristotle [1])*

- discussion of assignments, outline of approaches
- consolidation of the lecture, clarification of open questions
deepen knowledge by means of direct experience

*Acquisition of virtuous behaviour and operational ability is less a matter of easy instruction but rather functional copy, practise, and use.* (Aristotle [1])

- discussion of assignments, outline of approaches
- consolidation of the lecture, clarification of open questions

**blackboard practice** under guidance of an exercise instructor

- registration through *WAFFEL*² (URL see CS web page)
- assignments are to be processed in teamwork: discretionary clause
  - depending on the number of participants
Exercise

- deepen knowledge by means of direct experience

  *Acquisition of virtuous behaviour and operational ability is less a matter of easy instruction but rather functional copy, practise, and use. (Aristotle [1])*

- discussion of assignments, outline of approaches
- consolidation of the lecture, clarification of open questions

- **blackboard practice** under guidance of an exercise instructor
  - registration through **WAFFEL**² (URL see CS web page)
  - assignments are to be processed in teamwork: discretionary clause
    - depending on the number of participants

- **computer work** under individual responsibility
  - registration is not scheduled, reserved workplaces are available
  - in case of questions, a CS exercise instructor is available

²abbr. for (Ger.) Webanmeldefrickelformular Enterprise Logic
Requirements

- **hard skills (computer-science expertise)**
  - mandatory
    - structured computer organisation
    - algorithm design and development
    - principles of programming in C or C++
    -> knowledge gaps will not be closed actively: no extra tuition
  - optional
    - assembly language (absolute) programming
    - system programming
    - operating systems
    -> as appropriate, knowledge gaps will be closed on demand by the instructors
Requirements

- **hard skills** (computer-science expertise)
  - mandatory
    - structured computer organisation
    - algorithm design and development
    - principles of programming in C or C++
    ↞ knowledge gaps will not be closed actively: no extra tuition
  - optional
    - assembly language (absolute) programming
    - system programming
    - operating systems
    ↞ as appropriate, knowledge gaps will be closed on demand by the instructors

- **soft** (personal, social, methodical) **skills**
  - staying power, capacity of teamwork, structured problem solving
achievable credit points
- 5 ECTS (*European Credit Transfer System*)
achievable credit points

- 5 ECTS (*European Credit Transfer System*)
- corresponding to a face time of 4 contact hours per week
  - lecture and practice, with 2 SWS\(^3\) (i.e., 2.5 ECTS) each

\(^3\)abbr. for (Ger.) *Semesterwochenstunden*
Major Course Assessment

- achievable credit points
  - 5 ECTS (*European Credit Transfer System*)
  - corresponding to a face time of 4 contact hours per week
    - lecture and practice, with 2 SWS\(^3\) (i.e., 2.5 ECTS) each

- German or English (cf. p. 21) **oral examination**
  - date by arrangement: send e-mail to wosch@cs.fau.de
  - propose desired date within the official audit period
    - the exception (from this very period) proves the rule...

\(^3\) abbr. for (Ger.) *Semesterwochenstunden*
Major Course Assessment

- achievable credit points
  - 5 ECTS (European Credit Transfer System)
  - corresponding to a face time of 4 contact hours per week
    - lecture and practice, with 2 SWS\textsuperscript{3} (i.e., 2.5 ECTS) each

- German or English (cf. p. 21) oral examination
  - date by arrangement: send e-mail to wosch@cs.fau.de
  - propose desired date within the official audit period
    - the exception (from this very period) proves the rule...

- examination subjects
  - topics of lecture, blackboard practice, but also computer work
  - brought up in the manner of an “expert talk”
    - major goal is to find out the degree of understanding of inter-relations

\textsuperscript{3}abbr. for (Ger.) Semesterwochenstunden
achievable credit points

- 5 ECTS (European Credit Transfer System)
- corresponding to a face time of 4 contact hours per week
  - lecture and practice, with 2 SWS\(^3\) (i.e., 2.5 ECTS) each

German or English (cf. p. 21) oral examination

- date by arrangement: send e-mail to wosch@cs.fau.de
- propose desired date within the official audit period
  - the exception (from this very period) proves the rule...

examination subjects

- topics of lecture, blackboard practice, but also computer work
- brought up in the manner of an “expert talk”
  - major goal is to find out the degree of understanding of inter-relations

registration through “mein campus”: https://www.campus.fau.de

\(^3\)abbr. for (Ger.) Semesterwochenstunden
Outline

Preface

Contents

Organisation

Summary
Subject Matter

- Coordination of cooperation and concurrency
  - Between interacting (i.e., control- or data-flow dependent) processes
  - With emphasis on explicit synchronisation

- Against the background of two dimensions of concurrency
  - **Vertical**
    - Overlapped execution at operating-system machine level
    - Process preemption (partial virtualisation)
  - **Horizontal**
    - Overlapped execution at instruction set architecture level
    - Processor (core) multiplication

- In-depth study of approaches suitable (not only) for operating systems
  - Advanced studies to the range of topics on system programming
  - Basic studies to concurrent (i.e., non sequential) programming

- Fundamental understanding of different synchronisation paradigms
  - Blocking versus non-blocking synchronisation
  - Where is what paradigm mandatory, optional, beneficial, or adversely...
Subject Matter

- coordination of cooperation and concurrency
  - between interacting (i.e., control- or data-flow dependent) processes
  - with emphasis on explicit synchronisation

- against the background of two dimensions of concurrency
  - vertical
    - overlapped execution at operating-system machine level
    - process preemption (partial virtualisation)
  - horizontal
    - overlapped execution at instruction set architecture level
    - processor (core) multiplication

- in-depth study of approaches suitable (not only) for operating systems
  - advanced studies to the range of topics on system programming
  - basic studies to concurrent (i.e., non sequential) programming

- fundamental understanding of different synchronisation paradigms
  - blocking versus non-blocking synchronisation
  - where is what paradigm mandatory, optional, beneficial, or adversely...
Subject Matter

- Coordination of cooperation and concurrency
  - Between interacting (i.e., control- or data-flow dependent) processes
  - With emphasis on explicit synchronisation

- Against the background of two dimensions of concurrency
  - **Vertical**
    - Overlapped execution at operating-system machine level
    - Process preemption (partial virtualisation)
  - **Horizontal**
    - Overlapped execution at instruction set architecture level
    - Processor (core) multiplication

- In-depth study of approaches suitable (not only) for operating systems
  - Advanced studies to the range of topics on system programming
  - Basic studies to concurrent (i.e., non sequential) programming

- Fundamental understanding of different synchronisation paradigms
  - Blocking versus non-blocking synchronisation
  - Where is what paradigm mandatory, optional, beneficial, or adversely...
Subject Matter

- coordination of cooperation and concurrency
  - between interacting (i.e., control- or data-flow dependent) processes
  - with emphasis on explicit synchronisation

- against the background of two dimensions of concurrency
  - **vertical**
    - overlapped execution at operating-system machine level
    - process preemption (partial virtualisation)
  - **horizontal**
    - overlapped execution at instruction set architecture level
    - processor (core) multiplication

- in-depth study of approaches suitable (not only) for operating systems
  - advanced studies to the range of topics on system programming
  - basic studies to concurrent (i.e., non sequential) programming

- fundamental understanding of different synchronisation paradigms
  - blocking versus non-blocking synchronisation
  - where is what paradigm mandatory, optional, beneficial, or adversely...
Subject Matter

- Coordination of cooperation and concurrency
  - Between interacting (i.e., control- or data-flow dependent) processes
  - With emphasis on explicit synchronisation

- Against the background of two dimensions of concurrency
  - **Vertical**
    - Overlapped execution at operating-system machine level
    - Process preemption (partial virtualisation)
  - **Horizontal**
    - Overlapped execution at instruction set architecture level
    - Processor (core) multiplication

- In-depth study of approaches suitable (not only) for operating systems
  - Advanced studies to the range of topics on system programming
  - Basic studies to concurrent (i.e., non sequential) programming

- Fundamental understanding of different synchronisation paradigms
  - Blocking versus non-blocking synchronisation
  - Where is what paradigm mandatory, optional, beneficial, or adversely...
Reference List

[1] Aristotle:  
*Nicomachean Ethics.*  
c. 334 BC

An Experimental Time-Sharing System.  

*Webster’s New World Dictionary.*  
Simon & Schuster, Inc., 1988

[4] Sutter, H.:  
The Free Lunch is Over: A Fundamental Turn Toward Concurrency in Software.  
In: *Dr. Dobb's Journal 30* (2005), Nr. 3, S. 202–210