Concurrent Systems
Nebenläufige Systeme

I. Introduction

Wolfgang Schröder-Preikschat

October 16, 2018

Abstract Concept
meaning of the lecture labelling in linguistic terms [?]:
con·cur·rent (lat.) concurrens: 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 which
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 incidental (contending)
- 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)
    - 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

- master/slave
- asymmetric
- clustered & symmetric
Parallel Processing

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 [7]
  - mandatory at least for operating systems
  - 28 cores, uniformly distributed across four tiles
- many-core processors make core multiplexing almost superfluous
  - unless latency hiding becomes an issue within a parallel process

Parallel Processor: CPU

- AMD, IBM, Intel, Tilera

<table>
<thead>
<tr>
<th>Cores</th>
<th>AMD</th>
<th>IBM</th>
<th>Intel</th>
<th>Tilera</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Parallel Processor: GPU

- NVIDIA

<table>
<thead>
<tr>
<th>Cores</th>
</tr>
</thead>
<tbody>
<tr>
<td>512</td>
</tr>
<tr>
<td>1536/3072</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Cores</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
</tr>
<tr>
<td>80</td>
</tr>
<tr>
<td>48</td>
</tr>
<tr>
<td>32</td>
</tr>
</tbody>
</table>
Parallel System: HPC

### Characteristic Parallel Systems

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

- **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
  - non-coherent: temporary (memory/cache) inconsistencies are tolerated

- **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

Outline

- Preface
- Contents
- Organisation
- Summary
Fundamentals

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

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

Pickings

State of the art and recapitulation:
14. current research work and advances in modern operating systems
    - remote-core locking [], unlocking energy []
    - read-copy update [], big kernel lock
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.
Language of Instruction

- depends on the German linguistic abilities of the participants
  - English: preferred working language
  - German: strict choice 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.

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 \([?]\))

  - 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

\(^2\)abbr. 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

- **soft** (personal, social, methodical) skills
  - staying power, capacity of teamwork, structured problem solving

---

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. 22) **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](https://www.campus.fau.de)

---

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 I

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

[2] **Corbató, F. J.; Merwin-Daggett, M.; Daley, R. C.:**
An Experimental Time-Sharing System.

[3] **Falsafi, B.; Guerraoui, R.; Picorel, J.; Trigonakis, V.:**
Unlocking energy.

[4] **Lozi, J.-P.; David, F.; Thomas, G.; Lawall, J. L.; Muller, G.:**
Remote core locking: migrating critical-section execution to improve the performance of multithreaded applications.

Reference List II

[5] **McKenney, P. E.; Slingwine, J. D.:**
Read-copy update: Using execution history to solve concurrency problems.

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

[7] **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