Concurrent Systems

Nebenläufige Systeme

XIV. Pickings

Wolfgang Schröder-Preikschat

February 5, 2019
Agenda

Recapitulation
  Concurrent Systems

Perspectives
  Parallel Systems
  Computing Equipment
  Further Education
Recapitulation

Concurrent Systems

Perspectives

Parallel Systems
Computing Equipment
Further Education
Content of Teaching and Cross-References

- Transactional memory
- Elementary operations
- Critical sections
- Simultaneous (concurrent/interacting) processes
- Concurrency
- Lock
- Semaphore
- Monitor
- Deadly embrace
- Guarded sections
- Non-blocking synchronisation
- Transactional memory
- Progress guarantee

© wosch, thoenig CS (WS 2018/19, LEC 14) Recapitulation – Concurrent Systems
Recapitulation
Concurrent Systems

Perspectives
Parallel Systems
Computing Equipment
Further Education
Main Research at the Chair

- **composability** and **configurability**
  - application-oriented (varying, type-safe) system software

- **specialisation**
  - dedicated operating systems: integrated, adaptive, parallel

“concurrent systems” is more or less cross-cutting thereto...
Main Research at the Chair

- **composability** and **configurability**
  - application-oriented (varying, type-safe) system software

- **specialisation**
  - dedicated operating systems: integrated, adaptive, parallel

- **reliability**
  - gentle fault and intrusion tolerance

- **thriftiness**
  - resource-aware operation of computing systems

- **timeliness**
  - migration paths between time- and event-triggered real-time systems
Main Research at the Chair

- **composability** and **configurability**
  - application-oriented (varying, type-safe) system software

- **specialisation**
  - dedicated operating systems: integrated, adaptive, parallel

- **reliability**
  - gentle fault and intrusion tolerance

- **thriftiness**
  - resource-aware operation of computing systems

- **timeliness**
  - migration paths between time- and event-triggered real-time systems

- **concurrency**
  - coordination of cooperation and competition between processes
Main Research at the Chair

- **composability** and **configurability**
  - application-oriented (varying, type-safe) system software

- **specialisation**
  - dedicated operating systems: integrated, adaptive, parallel

- **reliability**
  - gentle fault and intrusion tolerance

- **thriftiness**
  - resource-aware operation of computing systems

- **timeliness**
  - migration paths between time- and event-triggered real-time systems

- **concurrency**
  - coordination of cooperation and competition between processes

“concurrent systems” is more or less **cross-cutting** thereto...
Latency Awareness in Operating Systems

- Latency prevention
- Lock- and wait-free synchronisation
- Integrated generator-based approach
- Latency avoidance
- Interference protection
- Race-conflict containment
- Latency hiding
- Operating-system server cores
- Asynchronous remote system operation

Experiments with different operating-system architectures
- Process-/event-based and hardware-centric operating-system kernels

LAKE, Sloth

DFG: 2 doctoral researchers, 2 student assistants

1\textsuperscript{http://univis.uni-erlangen.de → Research projects → LAOS}
Latency Awareness in Operating Systems

- **latency prevention**
  - lock- and wait-free synchronisation
  - integrated generator-based approach

- **latency avoidance**
  - interference protection
  - race-conflict containment

- **latency hiding**
  - operating-system server cores
  - asynchronous remote system operation

---

1. [http://univis.uni-erlangen.de](http://univis.uni-erlangen.de) → Research projects → LAOS
Latency Awareness in Operating Systems

- **latency prevention**
  - lock- and wait-free synchronisation
  - integrated generator-based approach

- **latency avoidance**
  - interference protection
  - race-conflict containment

- **latency hiding**
  - operating-system server cores
  - asynchronous remote system operation

- experiments with different *operating-system architectures*
  - process-/event-based and hardware-centric operating-system kernels
  - LAKE, Sloth

---

¹[http://univis.uni-erlangen.de](http://univis.uni-erlangen.de) → Research projects → LAOS
Latency Awareness in Operating Systems

- **latency prevention**
  - lock- and wait-free synchronisation
  - integrated generator-based approach

- **latency avoidance**
  - interference protection
  - race-conflict containment

- **latency hiding**
  - operating-system server cores
  - asynchronous remote system operation

- experiments with different operating-system architectures
  - process-/event-based and hardware-centric operating-system kernels
  - LAKE, Sloth

- DFG: 2 doctoral researchers, 2 student assistants

---

1[http://univis.uni-erlangen.de → Research projects → LAOS](http://univis.uni-erlangen.de → Research projects → LAOS)
Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Coherency Kernel

Cohe
Coherency Kernel

event-based minimal kernel
- cache-aware main-memory footprint
- hyper-threading of latent actions

http://univis.uni-erlangen.de → Research projects → COKE
Coherency Kernel

- **event-based minimal kernel**
  - cache-aware main-memory footprint
  - hyper-threading of latent actions
- **featherweight agreement protocols**
  - overall kernel-level synchronisation
  - families of consistency kernels

2http://univis.uni-erlangen.de → Research projects → COKE

© wosch, thoenig          CS (WS 2018/19, LEC 14)          Perspectives – Parallel Systems
Coherency Kernel

- **event-based minimal kernel**
  - cache-aware main-memory footprint
  - hyper-threading of latent actions
- **featherweight agreement protocols**
  - overall kernel-level synchronisation
  - families of consistency kernels
- **problem-oriented consistency**
  - sequential, entry, release consistency
  - functional hierarchy of consistency domains
  - memory domains for NUMA architectures

2http://univis.uni-erlangen.de → Research projects → COKE
Coherency Kernel

- **event-based minimal kernel**
  - cache-aware main-memory footprint
  - hyper-threading of latent actions
- featherweight **agreement protocols**
  - overall kernel-level synchronisation
  - families of consistency kernels
- **problem-oriented consistency**
  - sequential, entry, release consistency
  - functional hierarchy of consistency domains
  - memory domains for NUMA architectures
- implementation as to different **processor architectures**
  - partial or total, resp. \{in,\}coherent shared memory

\[^2\text{http://univis.uni-erlangen.de → Research projects → COKE}\]
Coherency Kernel

- event-based minimal kernel
  - cache-aware main-memory footprint
  - hyper-threading of latent actions
- featherweight agreement protocols
  - overall kernel-level synchronisation
  - families of consistency kernels
- problem-oriented consistency
  - sequential, entry, release consistency
  - functional hierarchy of consistency domains
  - memory domains for NUMA architectures
- implementation as to different processor architectures
  - partial or total, resp. \{in,\}coherent shared memory
- DFG: 2 doctoral researchers (1 FAU, 1 BTU)

²http://univis.uni-erlangen.de → Research projects → COKE
Run-Time Support System for Invasive Computing

Octo borrowed from the designation of a creature that:

i is highly parallel in its actions and
ii excellently can adapt oneself to its environment

the kraken (species Octopoda) can operate in parallel by virtue of its eight tentacle
is able to do customisation through camouflage and deimatic displays and
comes with a highly developed nervous system in order to attune to dynamic ambient conditions and effects

POS abbrv. for parallel operating system
an operating system that not only supports parallel processes but that also functions inherently parallel thereby

DFG: 2.5 doctoral researchers, 1 research/3 student assistants

http://univis.uni-erlangen.de → Research projects → iRTSS
Run-Time Support System for Invasive Computing

**Octo**

borrowed from the designation of a creature that:

i. is highly parallel in its actions and  
ii. excellently can adapt oneself to its environment

the kraken (species *Octopoda*)

- can operate in parallel by virtue of its eight tentacle  
- is able to do customisation through camouflage and deimatic displays and  
- comes with a highly developed nervous system  
  - in order to attune to dynamic ambient conditions and effects

**POS**

abbrv. for *parallel operating system*  

- an operating system that not only supports parallel processes  
- but that also functions *inherently parallel* thereby

³http://univis.uni-erlangen.de → Research projects → IRTSS
Run-Time Support System for Invasive Computing

**Octo**
- borrowed from the designation of a creature that:
  1. is highly parallel in its actions and
  2. excellently can adapt oneself to its environment
- the kraken (species *Octopoda*)
  - can operate in parallel by virtue of its eight tentacle
  - is able to do customisation through camouflage and deimatic displays and
  - comes with a highly developed nervous system
    - in order to attune to dynamic ambient conditions and effects

**POS**
- abbrv. for *parallel operating system*
  - an operating system that not only supports parallel processes
  - but that also functions *inherently parallel* thereby

**DFG:** 2.5 doctoral researchers, 1 research/3 student assistants

---

³http://univis.uni-erlangen.de → Research projects → iRTSS
Power-Aware Critical Sections

- Scalable synchronisation on the basis of agile critical sections
- Infrastructure load-dependent and self-organised change of protection against race conditions
- Linguistic support
- Preparation, characterisation, and capturing of declared critical sections
- Automated extraction of critical sections
- Notation language for critical sections
- Program analysis and LLVM integration/adaptation
- Power-aware system programming
- Mutual exclusion, guarded sections, transactions
- Dynamic dispatch of synchronisation protocols or critical sections, resp.
- Tamper-proof power-consumption measuring
- Instruction survey and statistics based on real and virtual machines
- Energy-consumption prediction or estimation, resp.

DFG: 2 doctoral researchers, 2 student assistants

4 http://univis.uni-erlangen.de → Research projects → PAX
Power-Aware Critical Sections

- scalable synchronisation on the basis of **agile critical sections**
  - infrastructure
    - load-dependent and self-organised change of protection against race conditions
- linguistic support
  - preparation, characterisation, and capturing of declared critical sections

---

4 [http://univis.uni-erlangen.de → Research projects → PAX](http://univis.uni-erlangen.de)
Power-Aware Critical Sections

- scalable synchronisation on the basis of **agile critical sections** infrastructure
- load-dependent and self-organised change of protection against race conditions

**linguistic support**
- preparation, characterisation, and capturing of declared critical sections

- automated extraction of critical sections
  - notation language for critical sections
  - program analysis and LLVM integration/adaptation

---

Power-Aware Critical Sections

scalable synchronisation on the basis of agile critical sections infrastructure

load-dependent and self-organised change of protection against race conditions

linguistic support

preparation, characterisation, and capturing of declared critical sections

automated extraction of critical sections

notation language for critical sections

program analysis and LLVM integration/adaptation

power-aware system programming

mutual exclusion, guarded sections, transactions

dynamic dispatch of synchronisation protocols or critical sections, resp.

http://univis.uni-erlangen.de → Research projects → PAX
Power-Aware Critical Sections

- scalable synchronisation on the basis of agile critical sections infrastructure
  - load-dependent and self-organised change of protection against race conditions

- linguistic support
  - preparation, characterisation, and capturing of declared critical sections

- automated extraction of critical sections
  - notation language for critical sections
  - program analysis and LLVM integration/adaptation

- power-aware system programming
  - mutual exclusion, guarded sections, transactions
  - dynamic dispatch of synchronisation protocols or critical sections, resp.

- tamper-proof power-consumption measuring
  - instruction survey and statistics based on real and virtual machines
  - energy-consumption prediction or estimation, resp.

---

4http://univis.uni-erlangen.de → Research projects → PAX
Power-Aware Critical Sections

- scalable synchronisation on the basis of agile critical sections infrastructure
  - load-dependent and self-organised change of protection against race conditions

- linguistic support
  - preparation, characterisation, and capturing of declared critical sections

- automated extraction of critical sections
  - notation language for critical sections
  - program analysis and LLVM integration/adaptation

- power-aware system programming
  - mutual exclusion, guarded sections, transactions
  - dynamic dispatch of synchronisation protocols or critical sections, resp.

- tamper-proof power-consumption measuring
  - instruction survey and statistics based on real and virtual machines
  - energy-consumption prediction or estimation, resp.

- DFG: 2 doctoral researchers, 2 student assistants

4http://univis.uni-erlangen.de → Research projects → PAX
Latency- and Resilience-Aware Networking

Auffassung von der kausalen [Vor]bestimmtheit
allen Geschehens bzw. Handelns (Duden)

latency-aware communication endpoints, optimised protocol stack
specialised resource management, predictable run-time behaviour

DFG: doctoral researchers, 2 student assistants (1 FAU, 1 Uni SB)

http://univis.uni-erlangen.de → Research projects → LARN
Latency- and Resilience-Aware Networking

- real-time capable network communication
  - transport channel for cyber-physical systems
  - predictable transmission latency
  - in a certain extent guaranteed quality criteria

5 http://univis.uni-erlangen.de → Research projects → LARN
real-time capable network communication
- transport channel for cyber-physical systems
- predictable transmission latency
- in a certain extent guaranteed quality criteria

deterministic run-time support
- Auffassung von der kausalen [Vor]bestimmtheit allen Geschehens bzw. Handelns (Duden)
- latency-aware communication endpoints, optimised protocol stack
- specialised resource management, predictable run-time behaviour

\[5\] http://univis.uni-erlangen.de → Research projects → LARN
Latency- and Resilience-Aware Networking

- **real-time capable network communication**
  - transport channel for cyber-physical systems
  - predictable transmission latency
  - in a certain extent guaranteed quality criteria

- **deterministic run-time support**

  *Auffassung von der kausalen [Vor]bestimmtheit allen Geschehens bzw. Handelns (Duden)*

  - latency-aware communication endpoints, optimised protocol stack
  - specialised resource management, predictable run-time behaviour
    - in time (phase 1) and energy (phase 2) respect

---

5. [http://univis.uni-erlangen.de](http://univis.uni-erlangen.de) → Research projects → LARN
Latency- and Resilience-Aware Networking

- **real-time capable network communication**
  - transport channel for cyber-physical systems
  - predictable transmission latency
  - in a certain extent guaranteed quality criteria

- **deterministic run-time support**

  *Auffassung von der kausalen [Vor]bestimmtheit allen Geschehens bzw. Handelns (Duden)*

  - latency-aware communication endpoints, optimised protocol stack
  - specialised resource management, predictable run-time behaviour
    - in time (phase 1) and energy (phase 2) respect

- DFG: doctoral researchers, 2 student assistants (1 FAU, 1 Uni SB)

---

5 http://univis.uni-erlangen.de → Research projects → LARN
### Multi/Many-Core Processor Pool

<table>
<thead>
<tr>
<th>faui4*</th>
<th>clock</th>
<th>cores per domain</th>
<th>domain</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>physical</td>
<td>logical</td>
</tr>
<tr>
<td>8e</td>
<td>2.9 GHz</td>
<td>8</td>
<td>16</td>
</tr>
<tr>
<td>8f</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9big01</td>
<td>2.5 GHz</td>
<td>6</td>
<td>–</td>
</tr>
<tr>
<td>9big02</td>
<td>2.2 GHz</td>
<td>10</td>
<td>20</td>
</tr>
<tr>
<td>9phi01</td>
<td>1.2 GHz</td>
<td>6</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td>1.1 GHz</td>
<td>57</td>
<td>228</td>
</tr>
<tr>
<td>scc</td>
<td>1.5 GHz</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>800 MHz</td>
<td>2</td>
<td>–</td>
</tr>
<tr>
<td>InvasIC</td>
<td>3.5 GHz</td>
<td>8</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>25 MHz</td>
<td>4</td>
<td>–</td>
</tr>
</tbody>
</table>
## Multi/Many-Core Processor Pool

<table>
<thead>
<tr>
<th>faui4*</th>
<th>clock</th>
<th>cores per domain</th>
<th>domain</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>physical</td>
<td>logical</td>
</tr>
<tr>
<td>8e</td>
<td>2.9 GHz</td>
<td>8</td>
<td>16</td>
</tr>
<tr>
<td>8f</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9big01</td>
<td>2.5 GHz</td>
<td>6</td>
<td>–</td>
</tr>
<tr>
<td>9big02</td>
<td>2.2 GHz</td>
<td>10</td>
<td>20</td>
</tr>
<tr>
<td>9phi01</td>
<td>1.2 GHz</td>
<td>6</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td>1.1 GHz</td>
<td>57</td>
<td>228</td>
</tr>
<tr>
<td>scc</td>
<td>1.5 GHz</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>800 MHz</td>
<td>2</td>
<td>–</td>
</tr>
<tr>
<td>InvasIC</td>
<td>3.5 GHz</td>
<td>8</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>25 MHz</td>
<td>4</td>
<td>–</td>
</tr>
</tbody>
</table>

Budgeted acquisition: further $n$-core systems, transactional memory

**OctoPOS**  $n \geq 64$

**PAX**  $n \geq 16$, plus several multi-core micro-controllers
Bachelor, Master, or Doctoral Thesis