Technische Universität Dresden

Faculty of Computer Science Institute of Systems Architecture, Operating Systems Group

ADVANCED OPERATING SYSTEMS

Summary and Outlook

https://tud.de/inf/os/studium/vorlesungen/betriebssystembau

HORST SCHIRMEIER

Agenda

- Summary
- Evaluation
- Exam
- Outlook
- Get Involved

Agenda

- **Summary**
- Evaluation
- Exam
- Outlook
- Get Involved

Advanced Operating Systems (AOS) – Contents

- **High-level OS topics:**
  - OS architectures, modern file systems, real-time systems, cloud computing / virtualization, mobile OSs, synchronization, trusted computing
- **Low-level, HW/SW interface topics:**
  - Memory consistency models, novel memory technologies, high-performance I/O, side channels

Agenda

- Summary
- **Evaluation**
- Exam
- Outlook
- Get Involved

Evaluation Results

- [switch to evaluation PDF]

You ha[illegible]e to say?

[illegible]me

→ Use[illegible] “Anonymer B[illegible]fkasten”

Evaluation

- **Please participate.** It only takes a few minutes. The results help us improve the course over time, and serve as a proof of teaching efforts towards faculty and rectorate.
- Extended deadline: 2026-07-22 (Wednesday!)

https://tud.link/bfvme2

You have more to say?
→ Contact me
→ Use our “Anonymer Briefkasten”

Agenda

- Summary
- Evaluation
- **Exam**
- Outlook
- Get Involved

Exam (1)

DSE was missing in the intro slides.

- Master Computer Science and Master DSE (modules INF-25-Ma-FSA-AOS and INF-DSE-20-E-DOS)
  - **Written exam** (60 minutes) after the lecturing period
  - more info on the AOS website: https://tud.de/inf/os/studium/vorlesungen/aos/written-exam-information
- Others (e.g. module INF-BAS4, INF-VERT4): **Oral exam**, after the lecturing period
  - Contact os@mailbox.tu-dresden.de for an exam appointment **early**
    - Matriculation number; degree program (Master Informatik, DSE, IST, …); Module name (e.g., INF-BAS4); “AOS w/ Dr. Weinhold” (or: Prof. Schirmeier); date wishes + constraints (e.g., “between X and Y”)
    - for INF-BAS4 / INF-VERT4: list other lectures + examiners
  - Expect delays due to second-examiner sync. (*Komplexprüfung*), vacation period, etc.
- Prerequisite for exam: **registration with the examination office** (deadline has passed)

Exam (2)

- **General Info**
  - **Lecture and Exercises** are relevant – Be able to solve exercise tasks!
  - Language: German or English (or, if necessary, a mix)

Exam (3)

- **Written Exam Procedure**
  - Bring required documents (valid official photo ID, e.g., **passport**; valid **student ID**)
  - Allowed aids:
    - one sheet of A4 paper (both sides) with your own **hand-written** notes (**NOT printed; NOT copied**)
    - permanent-ink pen (NOT pencil)
    - printed language dictionary
    - Drinks, food – as long as you don’t disturb others
  - **No electronic devices allowed**

Exam (4)

- **Oral Exam Procedure**
  - Closely **listen** to the question (ask if it was unclear!)
  - Answer the question as **completely and precisely** as possible
  - Feel free to **anticipate** follow-up questions
  - If applicable: Use pen & paper (provided by us), make examples, refer to your experiments from the exercises, …

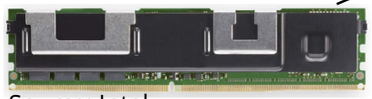
Agenda

- Summary
- Evaluation
- Exam
- **Outlook**
- Get Involved

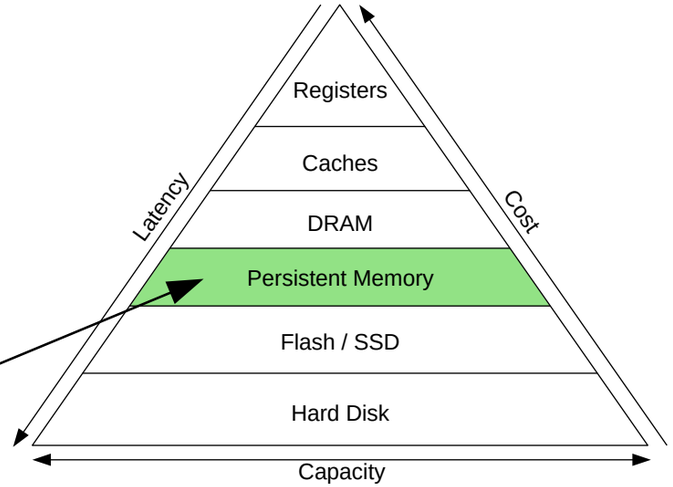
Challenge: New Memory Technologies

NVRAM: New class of memory between SSD & RAM

- Almost as fast as DRAM
- *Maintains its state if turned off*
- Available for servers since 2019
  - Optane DCPMMs (discontinued)



Source: Intel



Research Questions:

- File abstraction vs. direct, byte-wise access?
- Persistent data structures, processes, systems?
- Reliability? Reboot doesn't „fix" the system anymore!
- 1D memory hierarchy? *Demand Paging*?

Other emerging technologies:
Processing-in-memory (PIM)
High-bandwidth Memory (HBM)

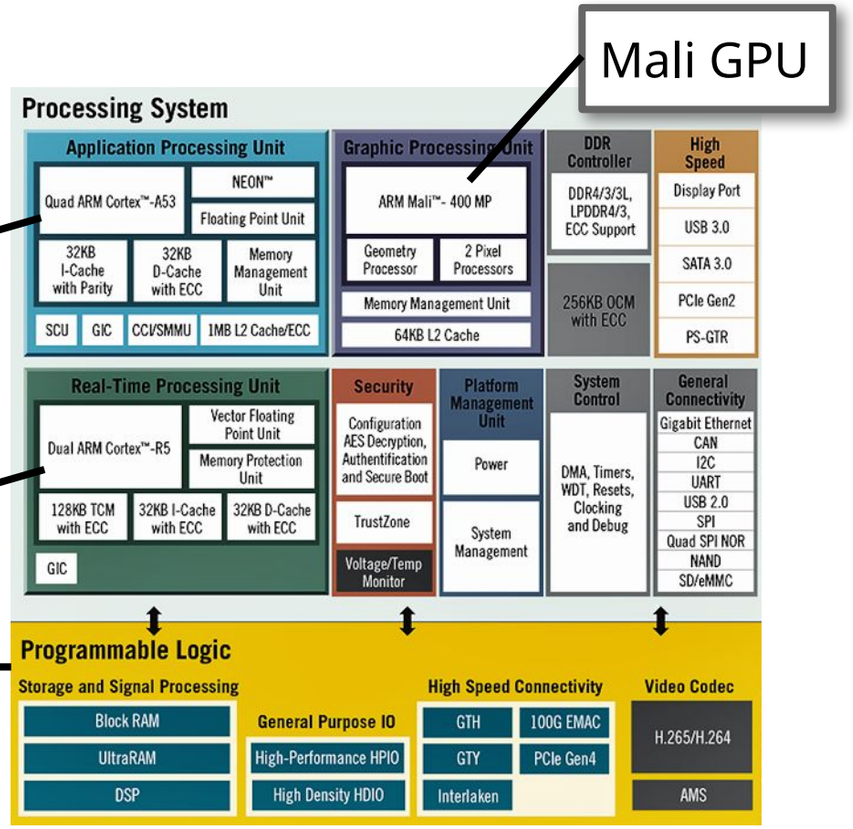
Challenge: Hardware Heterogeneity

- Example: Xilinx Ultrascale+

4 ARM Cortex A53 processor cores

2 ARM Cortex R5 real-time processors

Programmable logic (FPGA)



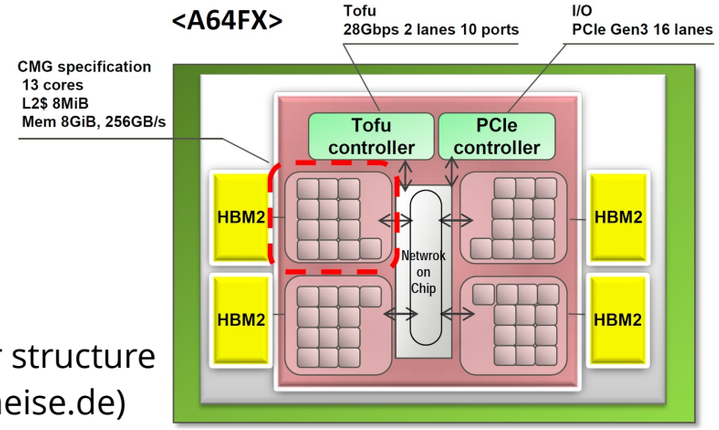
Research Questions:

- Common control-flow abstraction?
- How does scheduling work here?

Challenge: Manycore Hardware

- Example: Fujitsu A64FX (#1 top500.org)
  - 48+4 cores (2.7 Tflops) per chip
  - 4 High-Bandwidth Memories (1 TB/s); 4 NUMA regions
  - 7.299.072 cores in the supercomputer
  - *Remote DMA* (RDMA) for communication

Fugaku supercomputer (Source: Fujitsu)



Processor structure
(Source: heise.de)

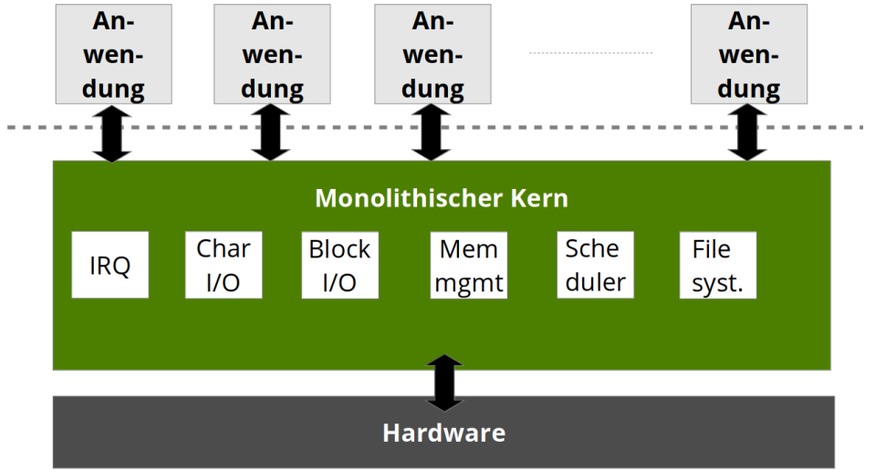
Research Questions:

- Do we still need CPU multiplexing?
- How to place control flows and data objects?

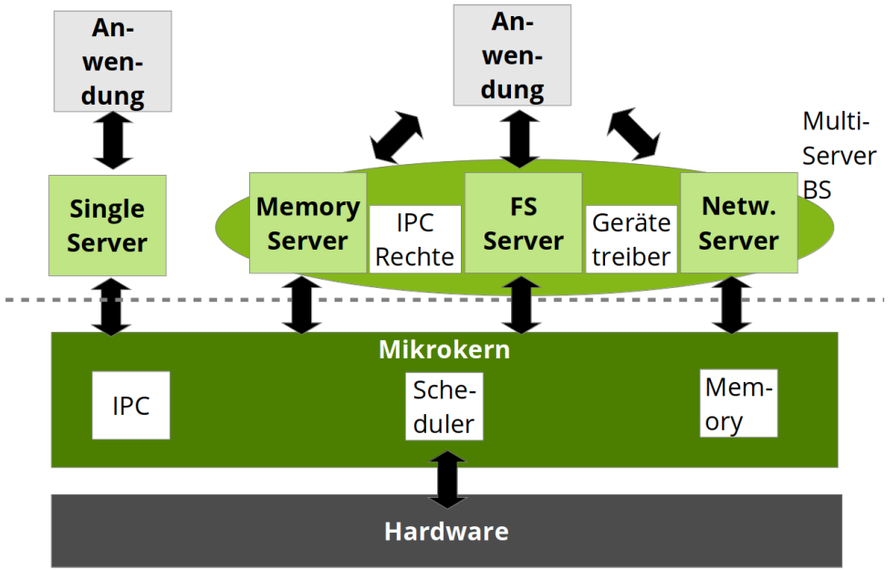
Research Topics at the OS Chair

- Dealing with **complexity**
  - constructively (projects L4, M^3)
  - analytically (project LockDoc) 
- **Non-functional properties**
  - *Security*
  - *Safety*/fault tolerance (projects DanceOS, FAIL*) 
  - Timing behavior
  - Energy and Scheduling (project HARP)
- **Hardware** developments
  - Disruptive memory technologies (projects VAMPIR, FOSSIL)

Complexity: Monolith vs. Microkernel



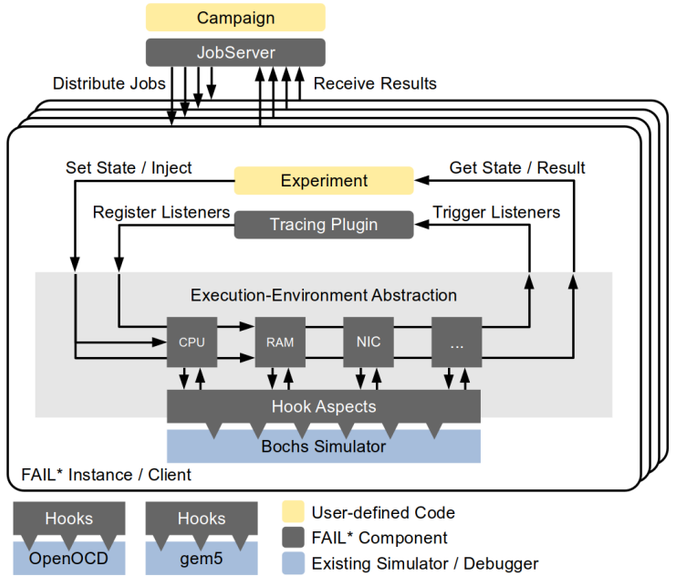
- Fine-grained locking is error-prone: **LockDoc** project
- *Security:* Take over a kernel component = Game Over
- but: Performance, lots of legacy code



- **L4Re** project
- Minimal, application specific ***Trusted Computing Base***
- Constructive complexity control (*divide & conquer*)

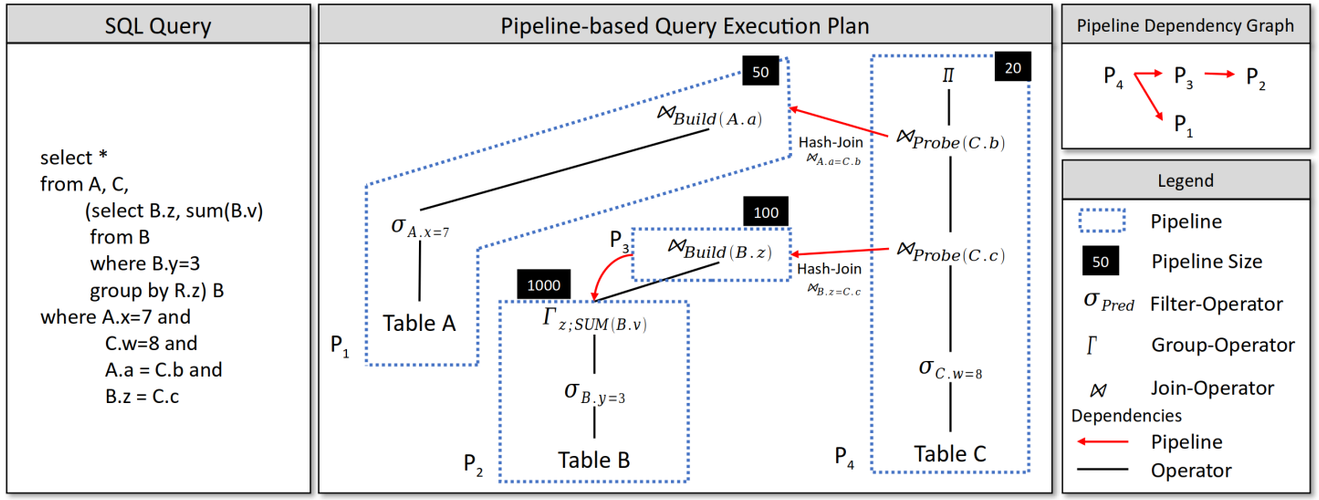
Fault-Tolerant Operating Systems

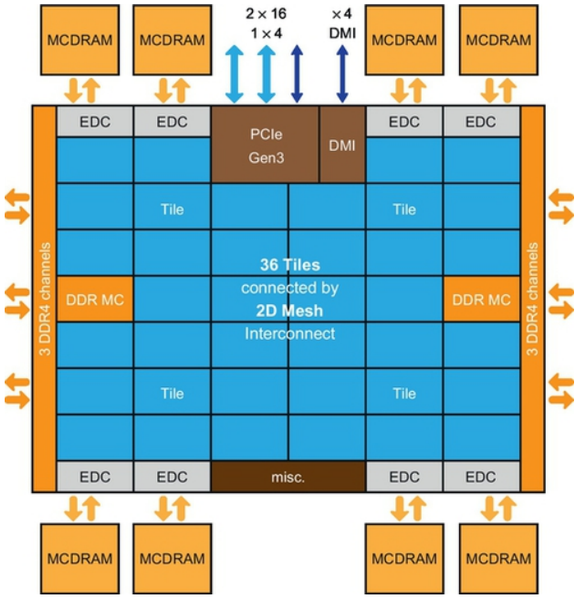
- *Soft Errors* can cause e.g. bit-flips in memory or the CPU
- How can we **extend** operating systems – or **design** them ground-up – so that they still work?
  - **DanceOS** project
- How can we (systematically) determine whether we were successful?
  - Fault injection: **FAIL*** project



Disruptive Memory Technologies

- Dealing with heterogeneous memories: **VAMPIR** project
  - Latency, throughput, persistency, fault tolerance, wearout, energy consumption, PIM capabilities, …
- Use case: databases
  - Predictions much easier!

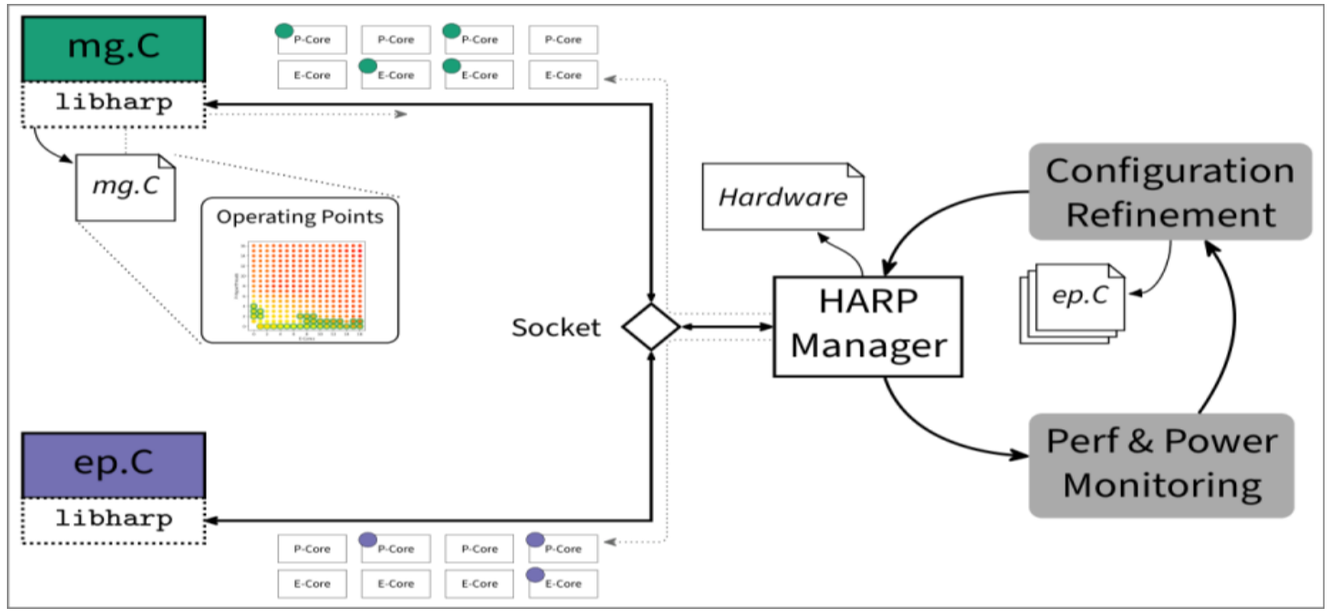




Energy-Aware Resource Management

- How to distribute tasks efficiently in a modern system? → **HARP**
  - Understand and integrate hardware properties → e.g. core types (P-cores vs. E-cores)
  - Understand and integrate application behavior → e.g. core preferences, resource sharing, ...

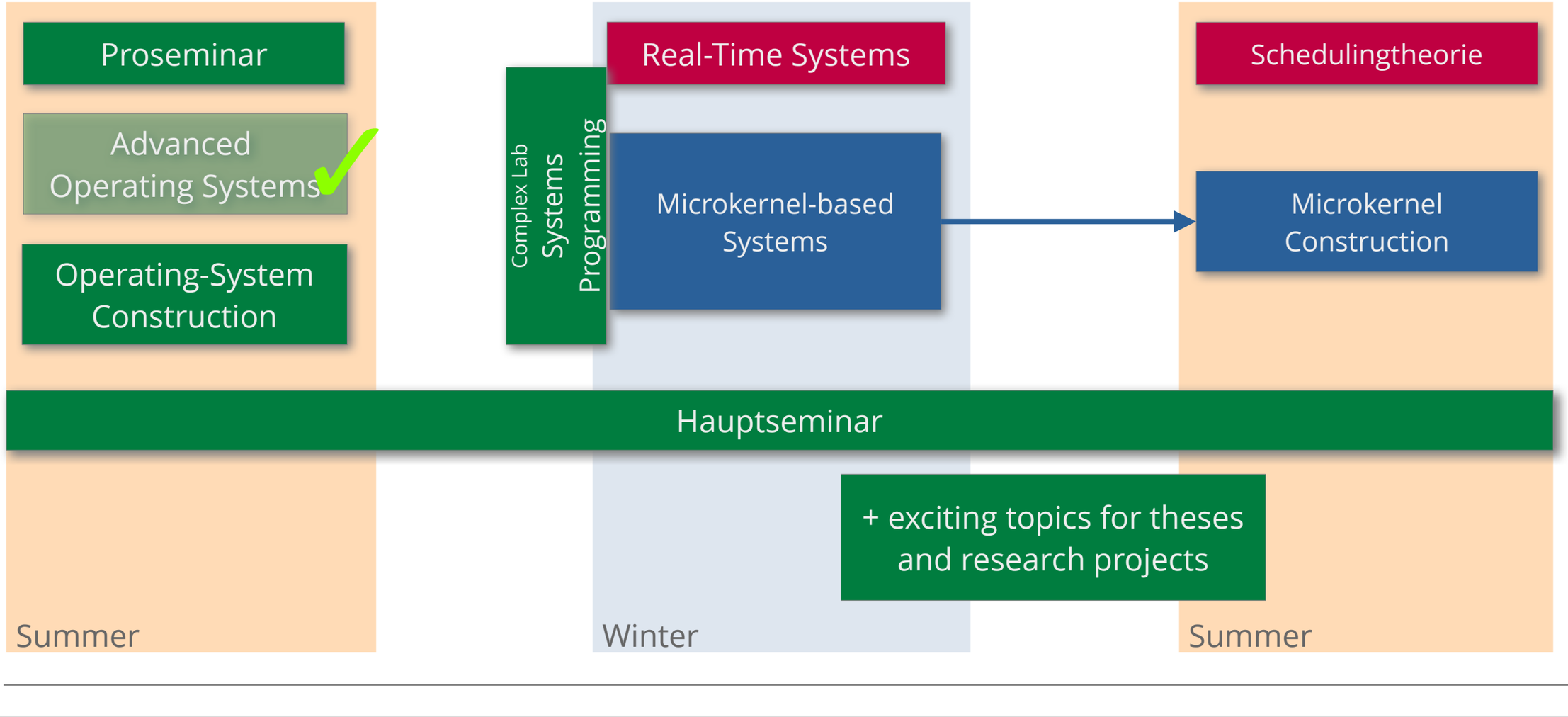
  → Find optimal solution at runtime (or at least as close as possible)



Agenda

- Summary
- Evaluation
- Exam
- Outlook
- **Get Involved**

Other Lectures



Thesis Topics

{Bachelor,Master,Diploma} theses, Beleg, Forschungsprojekt, …

- **Fl4mer:** Application Tracing via Intel PT on L4Re
- **TEECore:** Core Isolation via Cache Separation
- **NUMA-HARP:** Dynamic Congestion Detection via PEBS
- Dynamic Adaptive Polling in MPI
- Distributed Gang Scheduling for HPC Applications
- Program Analysis for Memory-Access-Pattern Deduction

More on our website

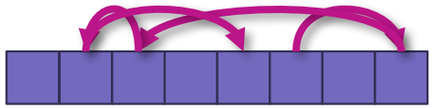
https://tud.de/inf/os/studium/abschlussarbeiten

Program Analysis for Memory-Access-Pattern Deduction

What are memory access patterns?

Sequential Access

Random Access



Partial Sequential Access

...

Why do they matter?

Access Pattern	Data Throughput [GB/s]			
	Singlethreaded		Multithreaded	
	DRAM	HBM	DRAM	HBM
Seq. Read	17.1		75.0	
Seq. Write	6.7		60.0	
Rand. Read	0.82		7.5	
Rand. Write	0.93		4.0	

So given the access pattern and concurrency: **we can optimize the programs throughput**, by placing program data in the right memory

So what can we optimize (from an OS-perspective)?

- Access Pattern?
- Concurrency?
- Data Placement!

Moderns HPC-Systems provide different types of main memory.

e.g.

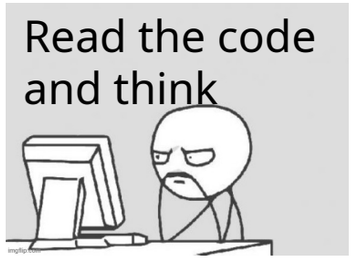
- DDR5 DRAM
- HBM2e
- CXL (attached memory)

Program Analysis for Memory-Access-Pattern Deduction

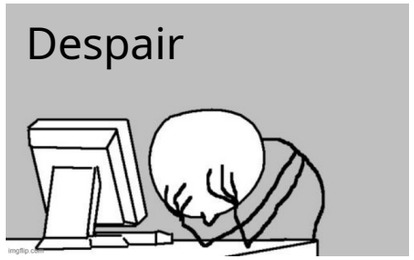
Goal: Reduce a program runtime by optimizing data throughput

Problem: What if the access pattern is not given?

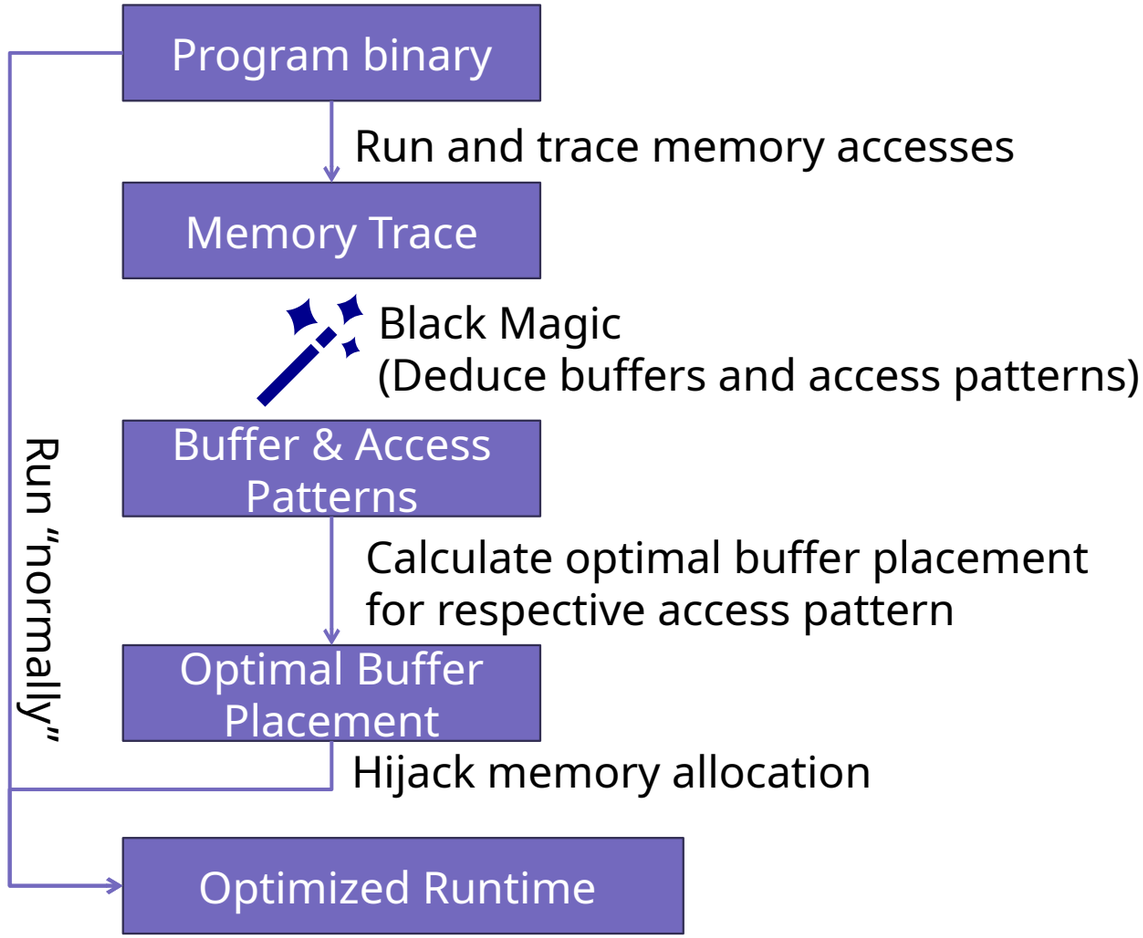
If we have access to the source code



If we don't have access to the source code (OS-perspective)



Idea: Trace programs memory interaction to deduce the access pattern



That’s It!

Thank you!
I hope we’ll meet again.

Don’t forget:
Side-channel lab part 2 (today, 13:00, APB/E042)