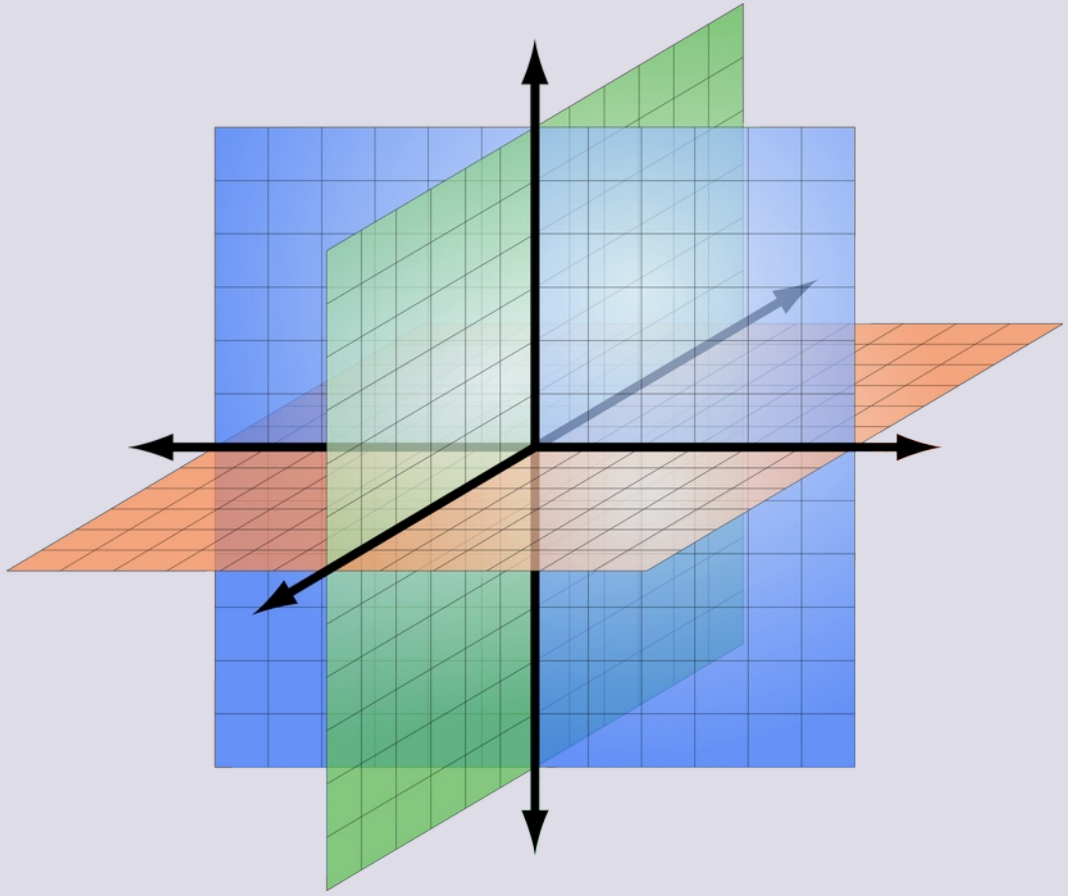
Advanced Operating Systems
Summer Semester 2024/2025

Martin Děcký

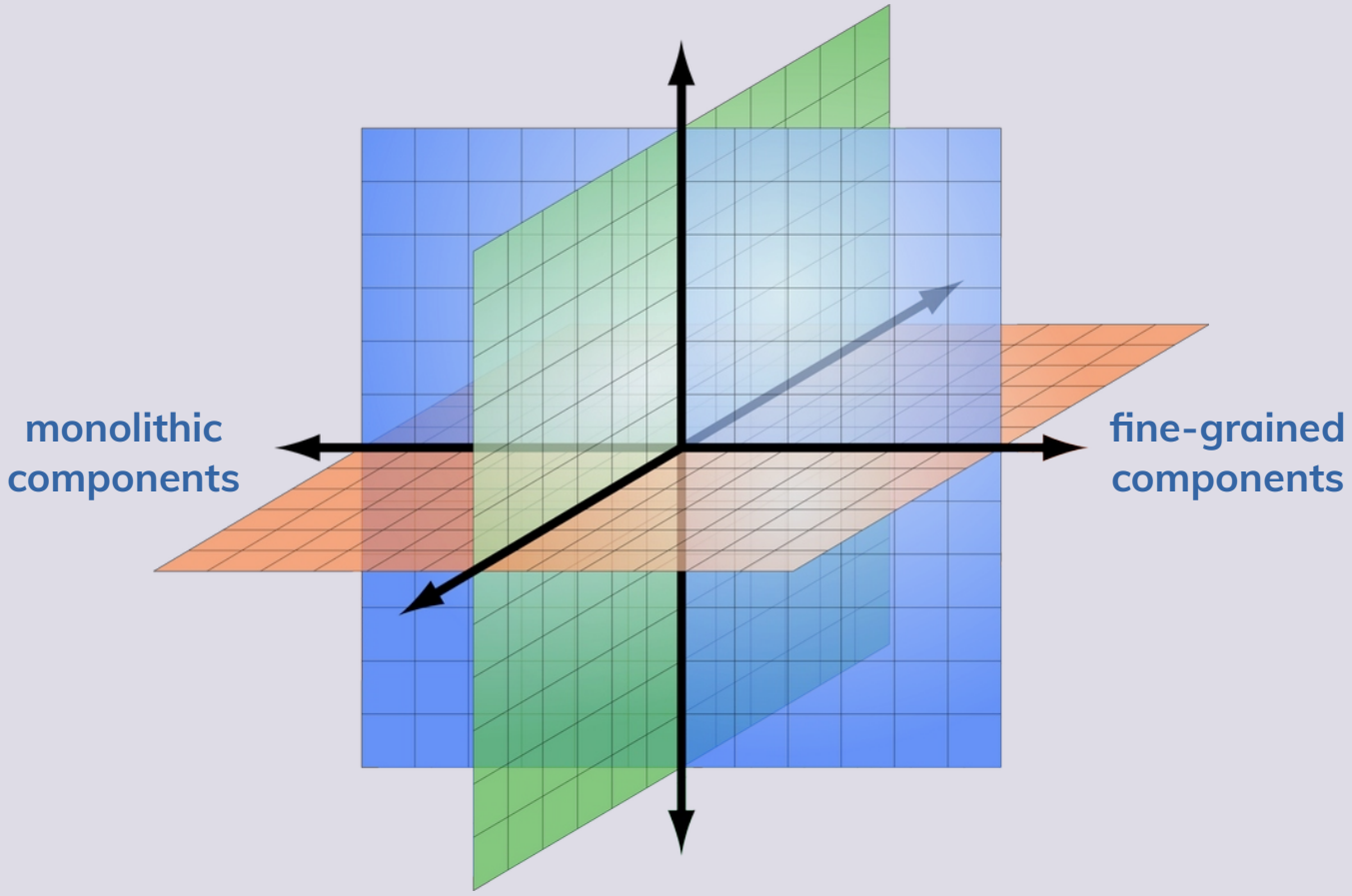
7

Architecture

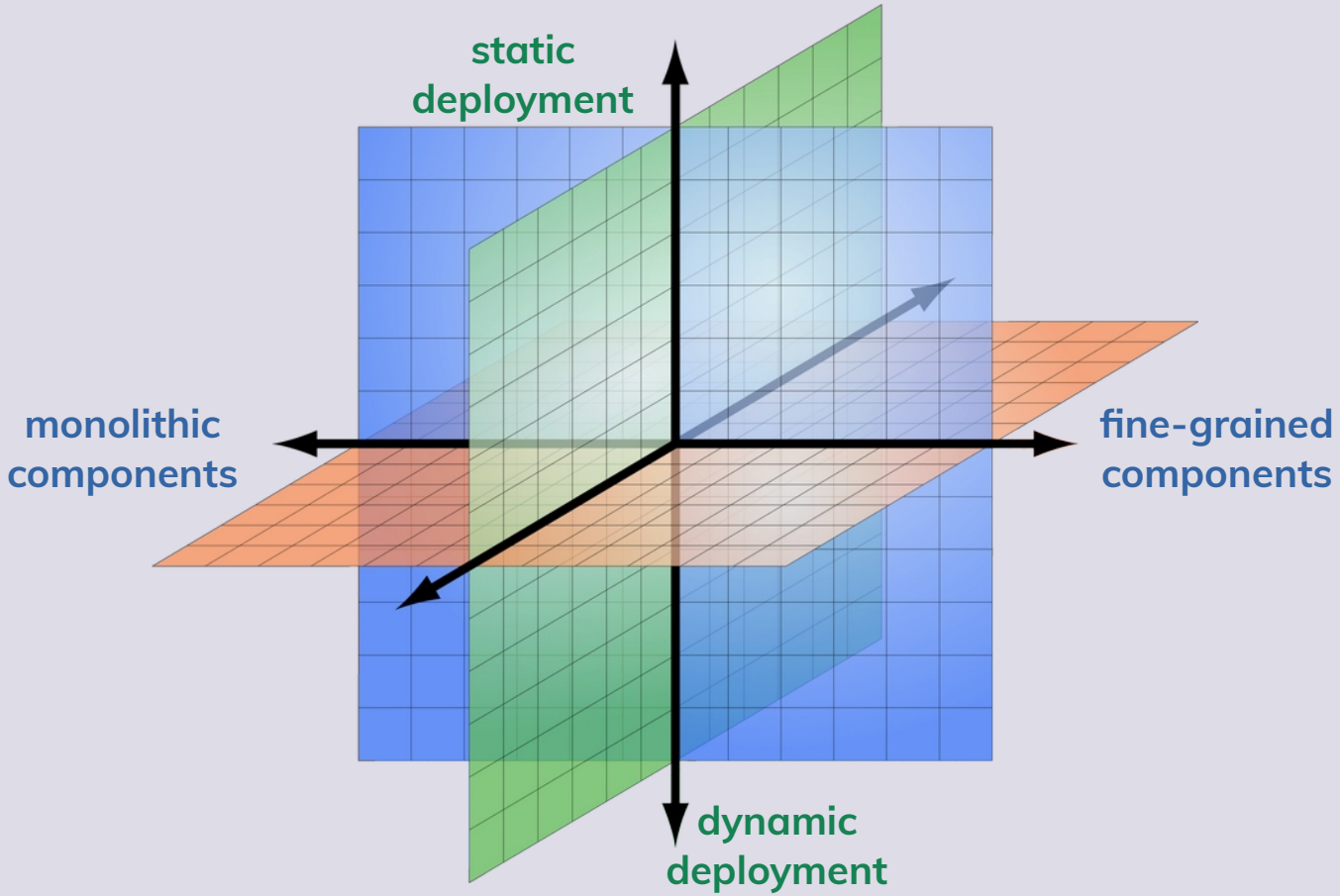
Operating Systems Design Space



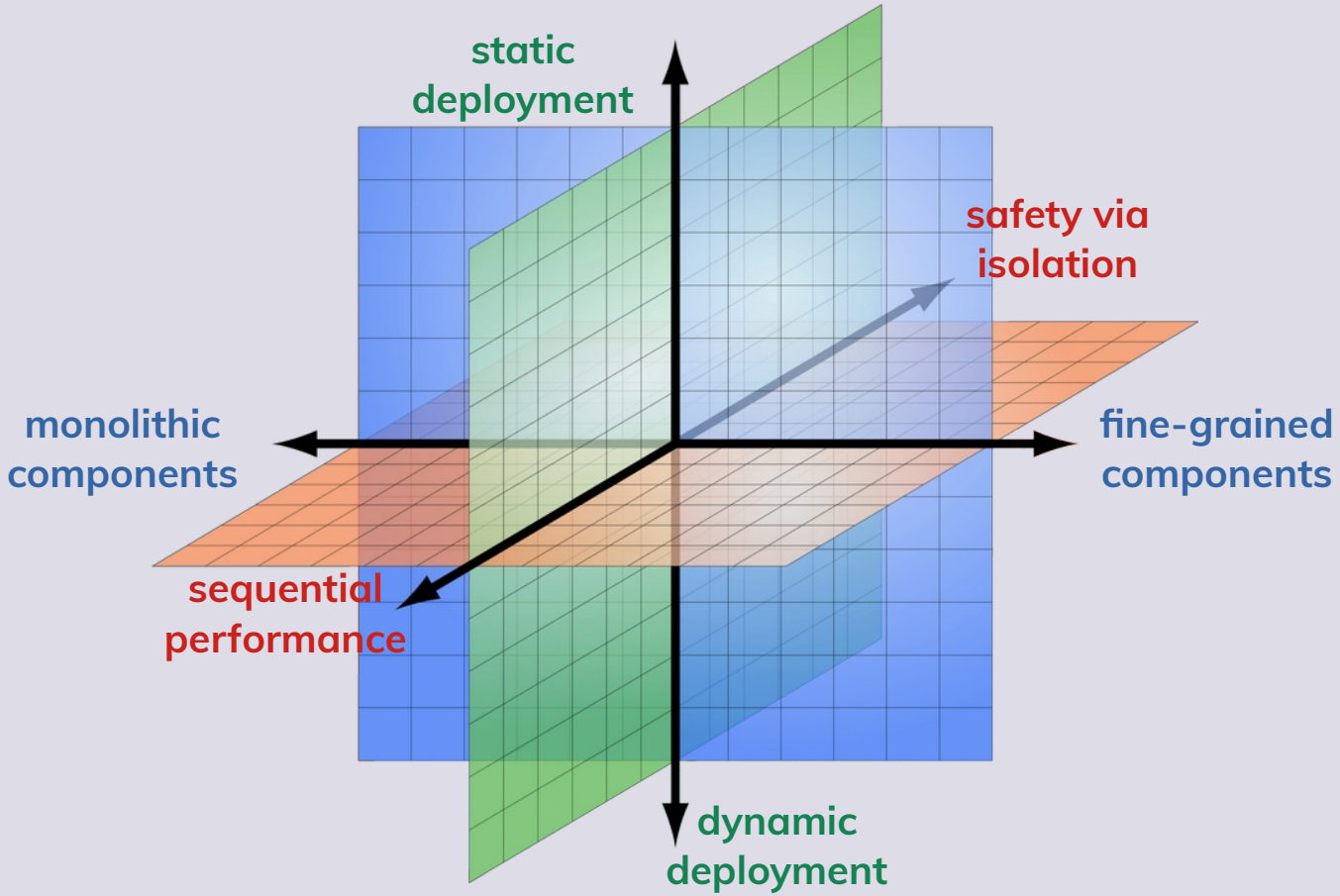
Operating Systems Design Space



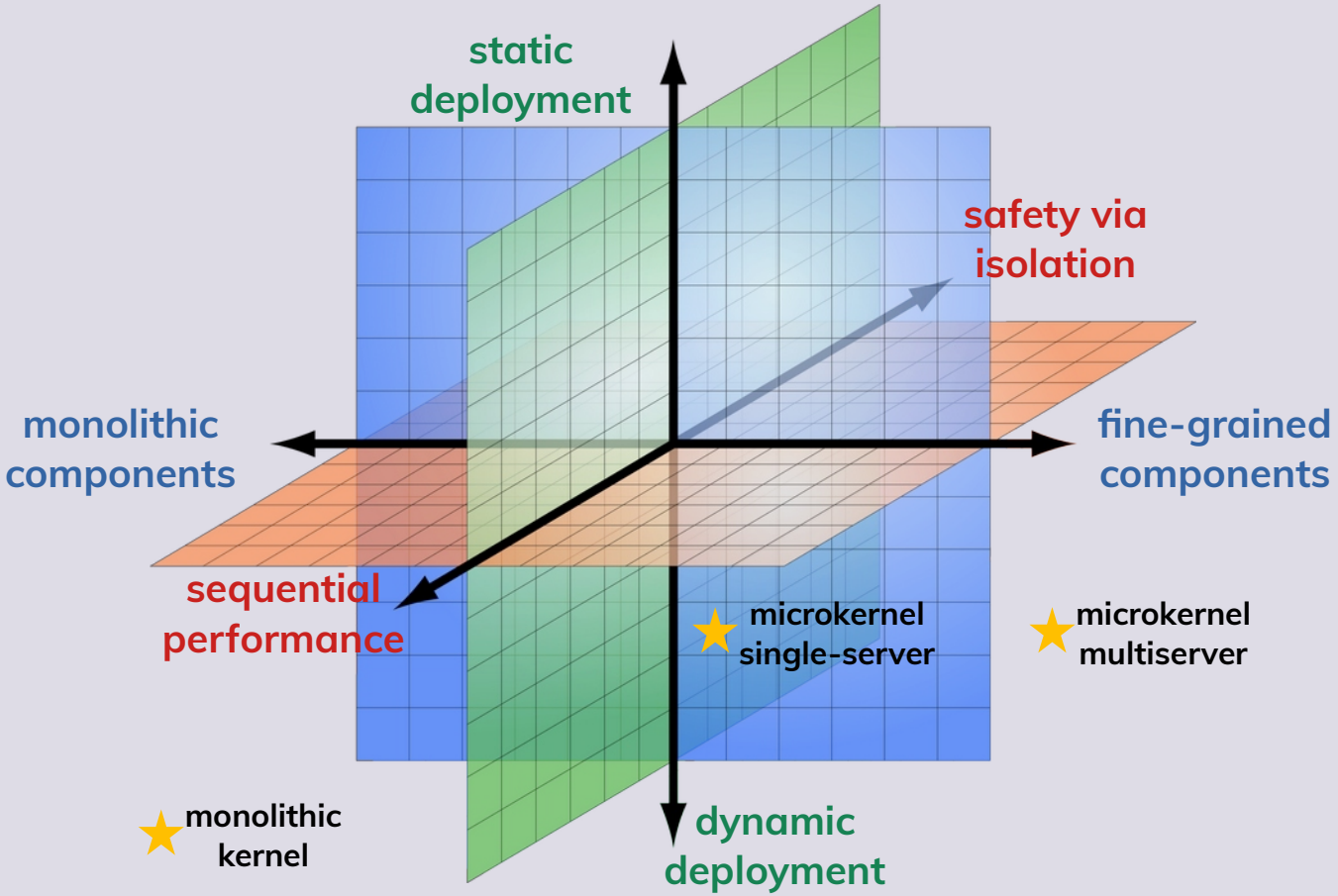
Operating Systems Design Space



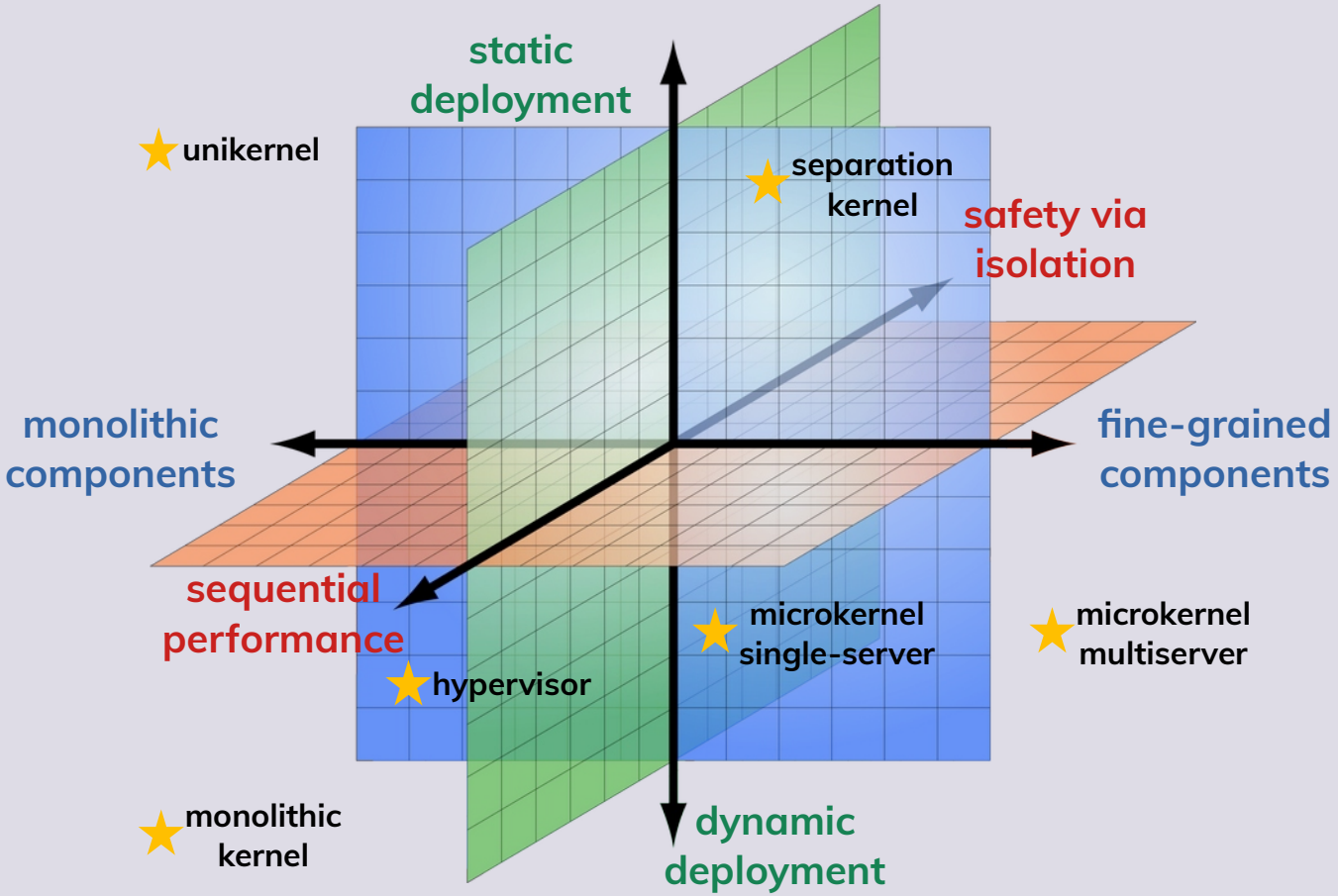
Operating Systems Design Space



Operating Systems Design Space



Operating Systems Design Space



Architecture Changing over Time

- **Windows NT**
  - Started essentially as a reimplementation of CMU Mach
    - Device drivers and file systems in user space
  - Gradually moved towards monolithic implementation
  - Recently certain device driver classes (USB, sound) in user space again
- **Linux**
  - Archetypal monolithic kernel
  - Originally with GPU drivers in user space (X11)
    - Gradually moved to kernel space
  - Other device drivers optionally in user space (FUSE, USB, etc.)
  - Proliferation of eBPF

Very Abridged Operating Systems History

- **1969**
  - *RC 4000 Multiprogramming System*
    - Per Brinch Hansen (Regnecentralen)
    - Separation of mechanism and policy, modularity via isolated concurrently running processes, message passing
    - Same year as Multics
- **1971**
  - *HYDRA*
    - William Wulf (Carnegie Mellon University)
    - Capability-based, object-oriented kernel
    - Around the same time as UNIX

Very Abridged Operating Systems History

- **1979**
  - *EUMEL / L2*
    - Jochen Liedtke (University of Bielefeld)
    - Microkernel running bitcode virtual machines
- **1982**
  - *QNX*
    - Gordon Bell, Dan Dodge (University of Waterloo, later Quantum Software Systems)
    - Earliest commercially successful microkernel-based OS (still in active development and use today, owned by BlackBerry)

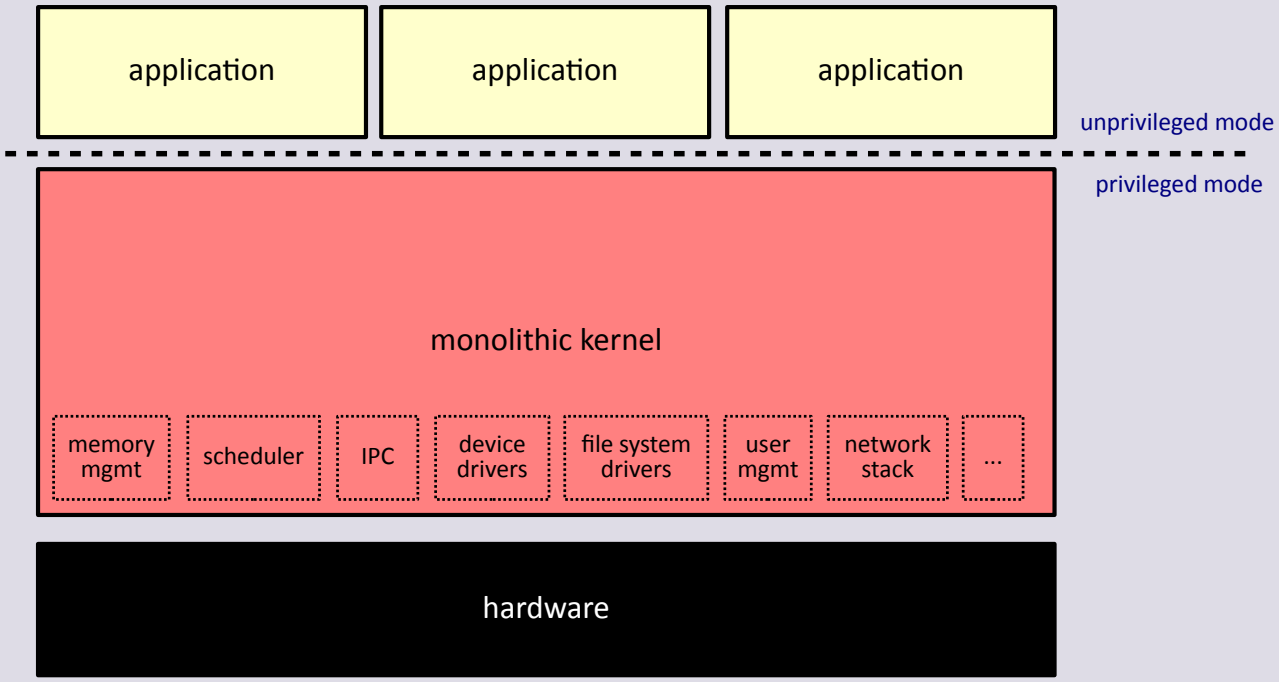
Very Abridged Operating Systems History

- **1985**
  - *CMU Mach*
    - Richard Rashid, Avie Tevanian (Carnegie Mellon University)
    - Arguably the most widespread microkernel code base
      - Core part of the operating systems by Apple (no longer following the original design principles) and GNU/Hurd
    - Highly influential
      - Affected the design of Windows NT
      - Establishing the usual terminology and conventions
    - Well-known shortcomings

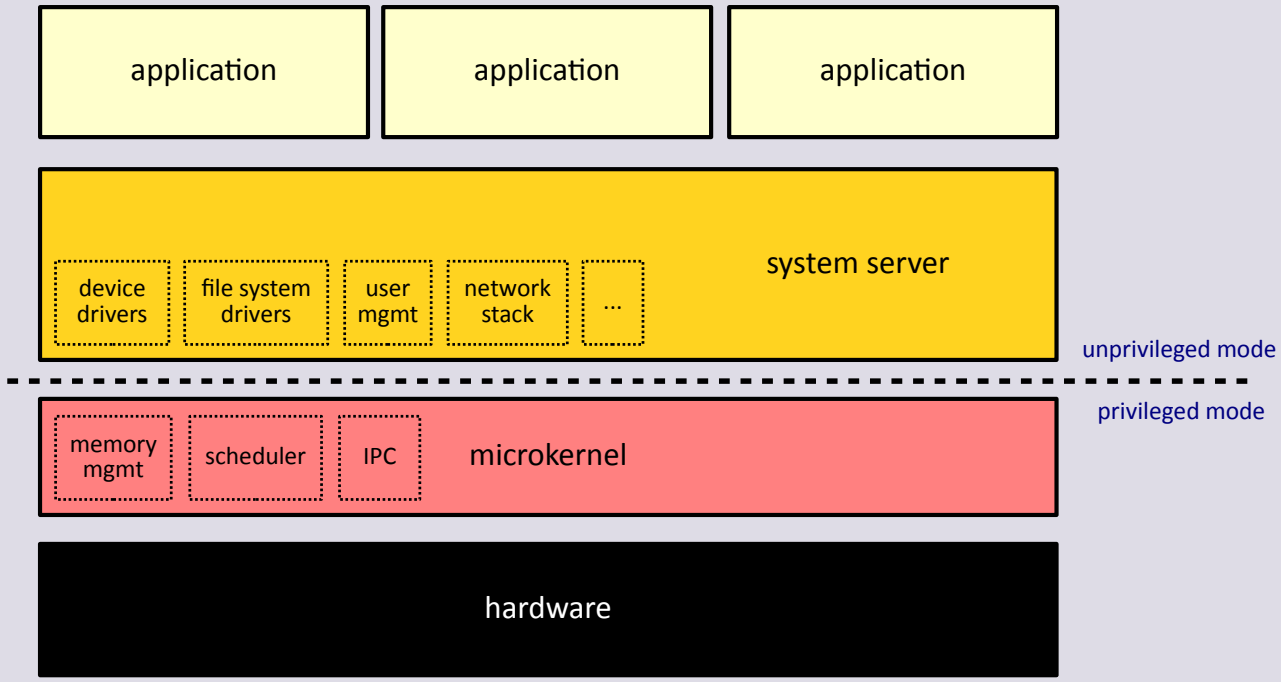
Very Abridged Operating Systems History

- **1988**
  - *L3*
    - Jochen Liedtke (Gesellschaft für Mathematik und Datenverarbeitung, later known as Fraunhofer)
    - Addressing the main performance issues of CMU Mach
      - Synchronous rendezvous-style remote calls instead of asynchronous in-kernel buffered message passing
- **1993**
  - *L4*
    - Order of magnitude performance improvement compared to CMU Mach
      - Small and cache-friendly kernel working set, fast-path IPC without complex processing (access rights, data interpretation, etc.)
    - User mode pagers and recursive address spaces
    - Non-portable hand-written assembly implementation (for 486 and Pentium)

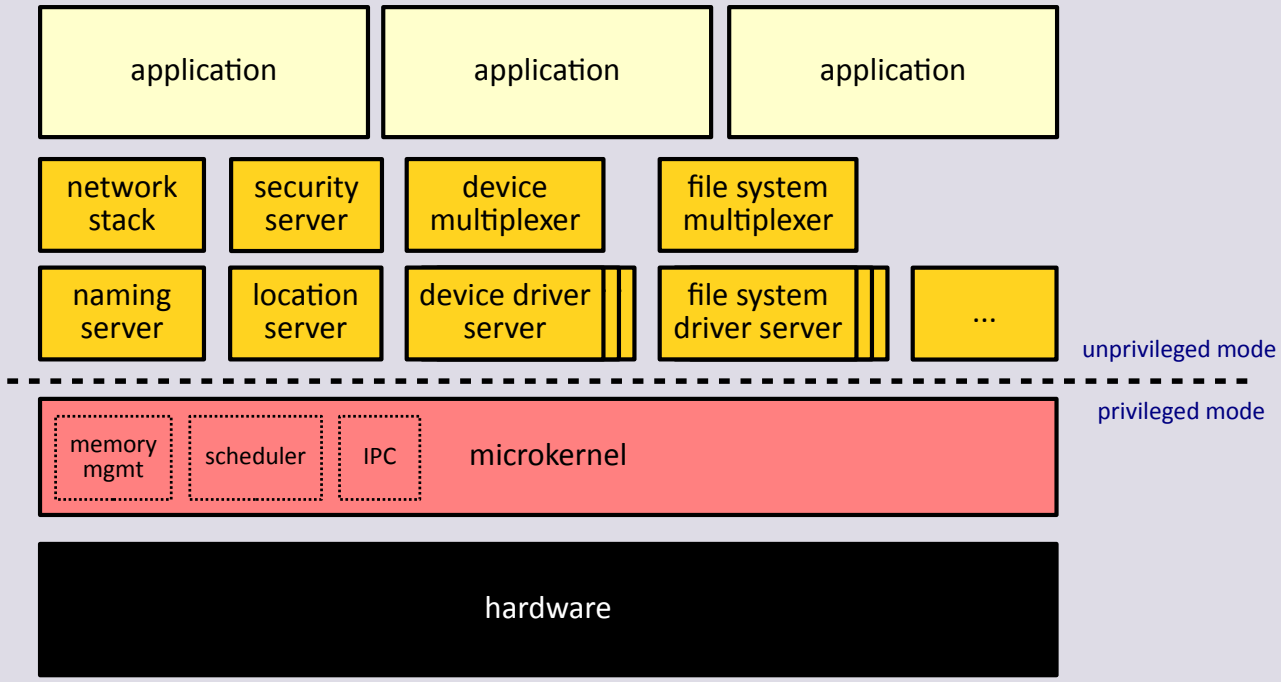
Monolithic OS Architecture



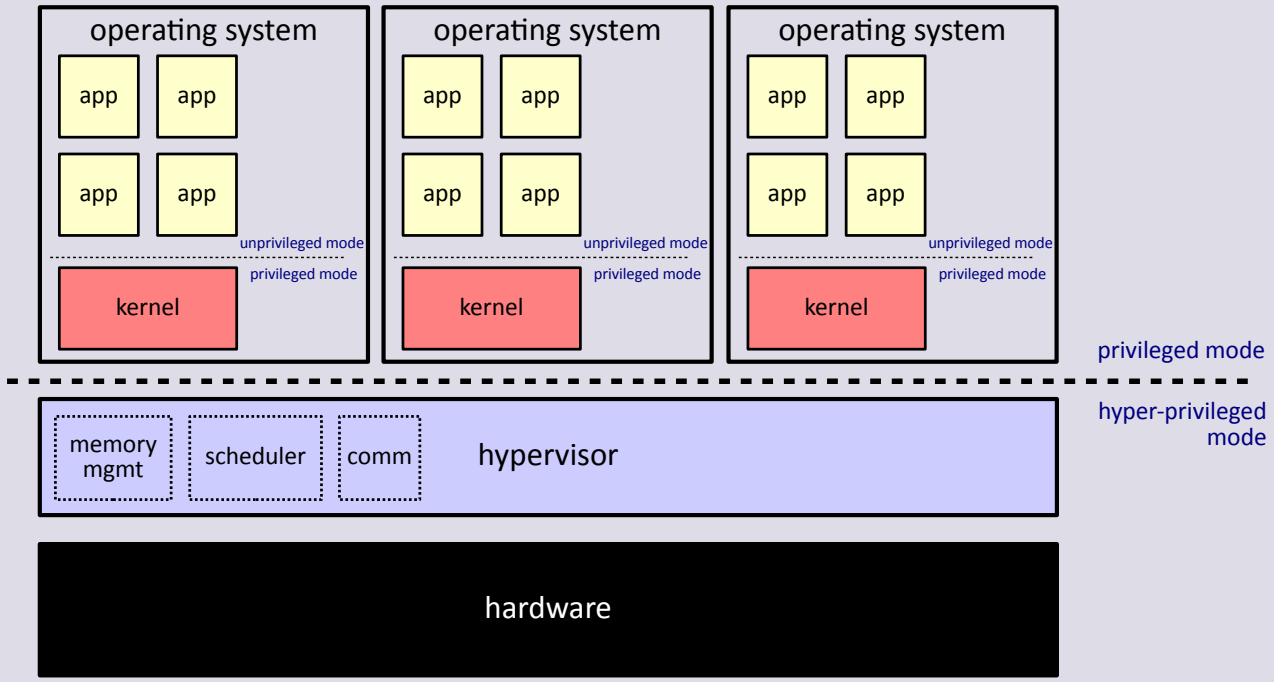
Single-server Microkernel OS Architecture



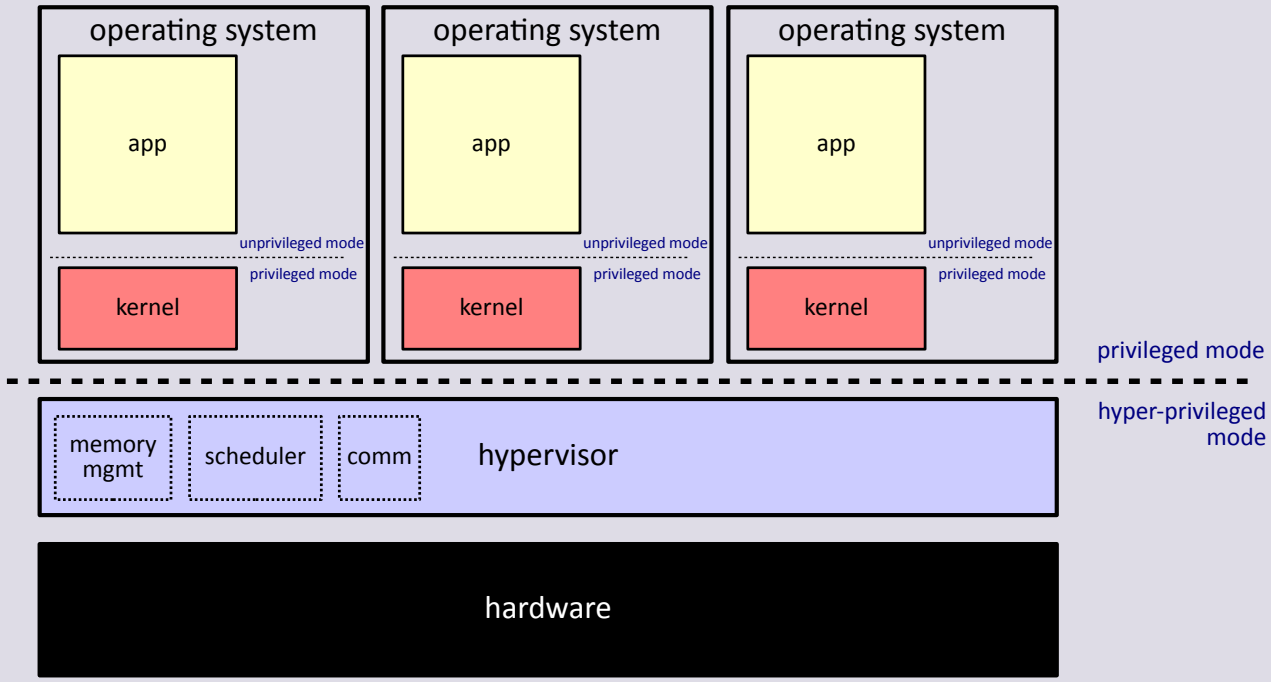
Multiserver Microkernel OS Architecture



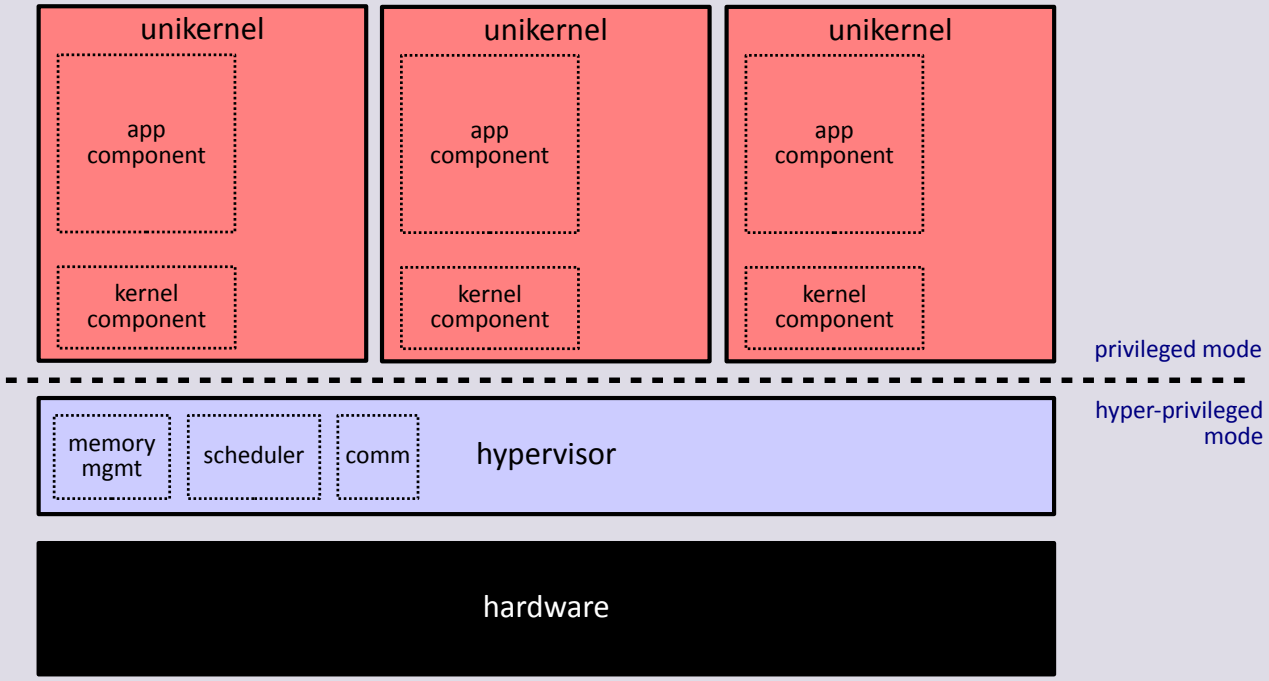
Type-1 Hypervisor Architecture



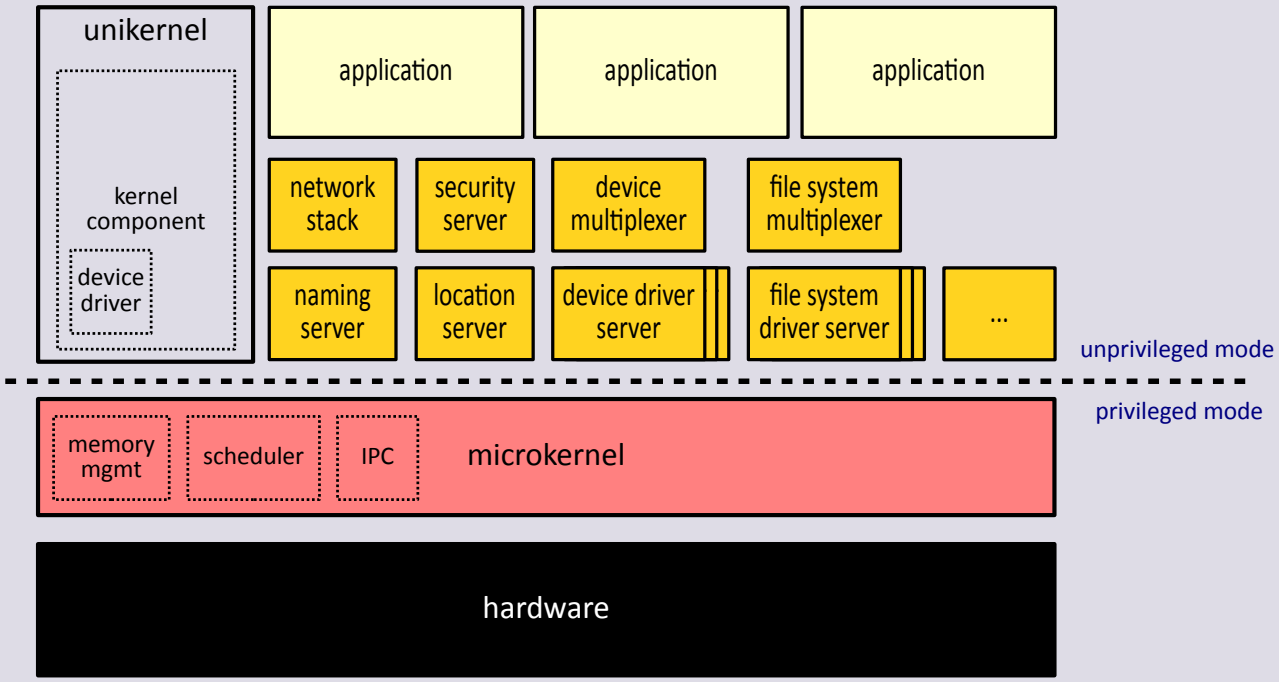
Type-1 Hypervisor Architecture



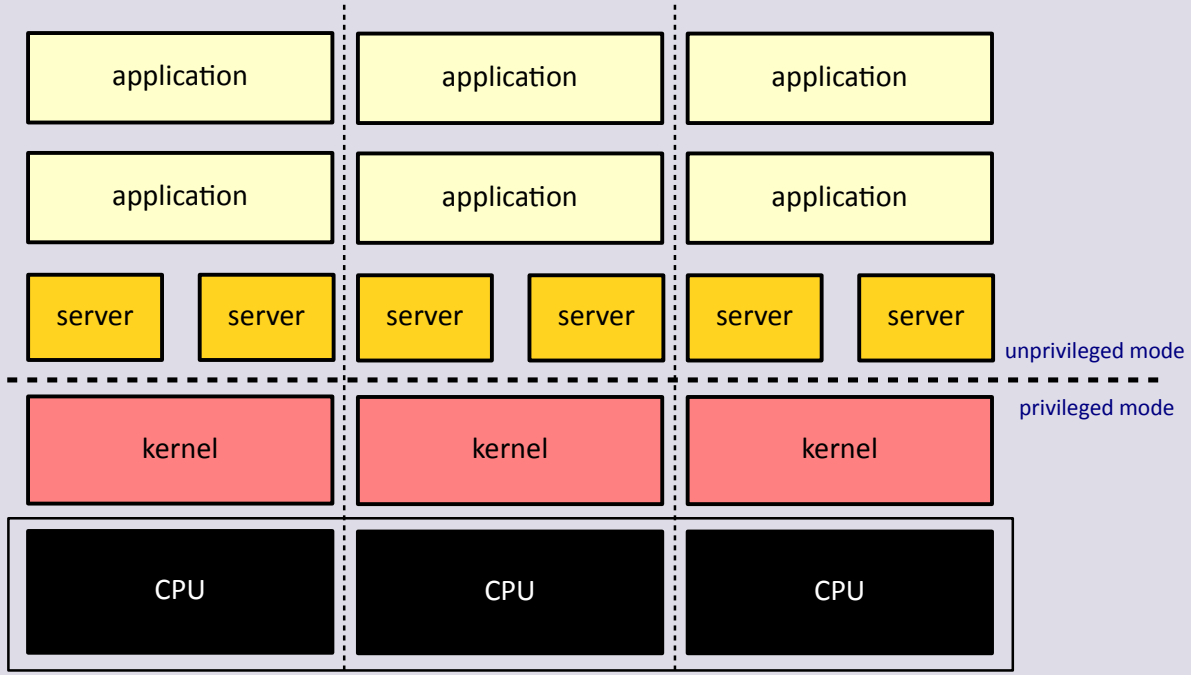
Type-1 Hypervisor Architecture



Multiserver Microkernel OS Architecture



Multikernel OS Architecture



Critical Systems

- **Mission-critical systems**
  - Essential to business/organization survival
    - E.g. on-line banking, state secrets, transport operation, electric grid
  - Usually associated with *security* properties (protecting computers against humans)
    - Fail-safe design
- **Safety-critical systems**
  - Essential to human well-being and survival
    - E.g. medical devices, transport control, nuclear power plant control
  - Usually associated with *safety* properties (protecting humans against computers)
    - Fail-operational design

Critical Systems

- **Avoiding fundamentally unreliable software architecture**
  - *"To me, writing a monolithic system in 1991 is a truly poor idea."*
    [Tanenbaum 1991]
  - *"There are no demonstrated examples of highly secure or highly robust unstructured (monolithic) systems in the history of computing."*
    [Shapiro 2006]
  - *"An operating system is said to be reliable when a typical user has never experienced even a single failure in his or her lifetime and does not know anybody who has ever experienced a failure."*
    [Tanenbaum 2014]
  - *"If one in million car tires randomly exploding is not acceptable, why is this still acceptable in software?"*
    [Tanenbaum 2021]

Mixed-criticality Systems

- **Accommodating two types of workloads side-by-side**
  - High criticality
    - Strict requirements on safety, security, real-time behavior, etc.
      - Explicit requirements
      - Formalized development process
  - Low criticality
    - Focus on sequential performance and vendor/user customization
    - Cost-prohibitive to apply same level of scrutiny as for high criticality
      - Lack of explicit and a priori defined requirements and criteria of correctness
      - Agile development
  - Hardware vs. software separation

Monolithic OS Design Is Flawed

- **Biggs S., Lee D., Heiser G.: *The Jury Is In: Monolithic OS Design Is Flawed: Microkernel-based Designs Improve Security*, ACM 9th Asia-Pacific Workshop on Systems (APSys), 2018**
  - *"While intuitive, the benefits of the small TCB have not been quantified to date. We address this by a study of critical Linux CVEs, where we examine whether they would be prevented or mitigated by a microkernel-based design. We find that almost all exploits are at least mitigated to less than critical severity, and 40 % completely eliminated by an OS design based on a verified microkernel, such as seL4."*
    - https://dl.acm.org/doi/10.1145/3265723.3265733

Microkernel Design Principles

- **Component-based architecture**
  - System composed of isolated components that communicate via well-defined interfaces
- **Separation of concerns**
  - Each component takes care of a specific well-defined functionality and implements it well
- **Split of mechanism and policy**
  - Components implement generic mechanisms without implicitly imposing a specific policy on the client components
- **Least privilege**
  - Components have a minimal set of privileges required to do their job

Microkernel Emerging Properties

- **Fine-grained components**
  - As opposed to monolithic components
- **Minimality of the kernel & trusted computing base**
  - Most mechanisms do not require the privileged CPU mode
  - File systems, most device drivers, security policies, etc., run as user mode components
- **Modularity**
  - Replacing component implementation while keeping the interface
- **Seamless virtualization**
  - VMs and tasks are essentially similar entities

Microkernel Emerging Properties

- **Loose module coupling**
  - Configurability via different composition of modules
  - Policies in user space and distributed
- **Architectural safety, security, reliability and dependability guarantees**
  - Limiting the “blast radius” of faults at run time
- **Architectural enabler for advanced reasoning about correctness**
  - Certification
  - Real-time guarantees
  - Formal verification

Practical Differences

- **Monolithic kernel**
  - Configurability via compile-time options and parametrization
  - Modularity via run-time dynamic linking
  - Tight module coupling, weak module cohesion
  - Structure is implicit and not enforced (especially at run time)
- **Microkernel**
  - Configurability via different use (policy in user space)
  - Modularity via extension in user space
  - Loose module coupling, strong module cohesion
  - Structure is explicit and enforced (even at run time)

Microkernel Overhead

- **A.k.a. the unfounded anxiety that refuses to die**
  - Liedtke has shown **30 years ago** that the overhead is **negligeable** (assuming proper microkernel design) [1]
  - Bershad has argued **33 years ago** that the IPC overhead is increasingly **irrelevant** (since the real-world performance of computer systems is dominated by other factors) [2]
  - The market share of monolithic operating systems is hardly caused by the lack of IPC overhead alone
    - The market share of Coca Cola is hardly caused by the taste alone
- **Real-world microkernel users simply “do not care about the overhead”**
  - The overall performance is satisfactory to them
  - Whatever measurable overhead is there, it is considered a reasonable price for the run-time component isolation and the safety/security guarantees that are fundamentally not available in monolithic operating systems

Microkernel Overhead

- **OSDI 2024 HongMeng paper [3]**
  - *“ In vehicles, HM achieves a 60% faster boot time and a 60% lower cross-domain latency. In smartphones, HM achieves 17% shorter app startup time and 10% less frame drops [than Linux].”*
  - *“HongMeng is a commercialized general-purpose microkernel that retains microkernel principles while providing structural supports to address compatibility and performance challenges in emerging scenarios. It also facilitates future exploration of microkernels’ benefits in production. For instance, its flexibility offers opportunities to accommodate the increasing hardware heterogeneity that Linux fails to address, and to achieve fault tolerance for improving availability.”*

Microkernel Overhead

- **Minimizing the overhead**
  - Little room for a major improvement purely on the software side
  - Improvements on the hardware side still possible
    - Mainstream CPUs are designed with monolithic operating systems in mind
      - A vicious cycle between CPU design and IPC overhead
    - **SkyBridge** [4]
    - **User Interrupts** on Intel's x86-64 [5]
    - Dedicated SW/HW co-design
      - **XPC** [6]
      - Asynchronous IPC and shared memory using cache lines
      - Unbound hardware multi-threading
      - Hardware capabilities (similar to CHERI, but targeting multiple protection domains)

Microkernel Overhead

- **Amortizing the overhead**
  - Highly parallel architectures
    - Asynchronous communication required anyway for optimal load balancing
  - Hererogeneous architectures
    - Distributed & disaggregated hardware
    - FPGA-based hardware
    - Non-cache coherent hardware
    - Different ISAs
    - Microkernels as universal abstractions for “CPU drivers”

References

[1] Bershad B. N.: *The Increasing Irrelevance of IPC Performance for Micro-Kernel-Based Operating Systems*, in Proceedings of the Workshop on Micro-Kernels and Other Kernel Architectures, USENIX, 1992, https://dl.acm.org/doi/10.5555/646405.692226

[2] Liedtke J.: *On Micro-Kernel Construction*, in Proceedings of the 15th ACM Symposium on Operating Systems Principles (SOSP), ACM, 1995, https://dl.acm.org/doi/10.1145/224056.224075

[3] Chen H., Xie M., Jia N., Wang N., Li Y., Liu N., Liu Y., Wang F., Huang Q., Li K., Yang H., Wang H., Yin J., Peng Y., Xu F.: *Microkernel Goes General: Performance and Compatibility in the HongMeng Production Microkernel*, in Proceedings of the 18th USENIX Conference on Operating Systems Design and Implementation (OSDI), 2024, https://dl.acm.org/doi/10.5555/3691938.3691963

[4] Mi Z., Li D., Yang Z., Wang X., Chen H.: *SkyBridge: Fast and Secure Inter-Process Communication for Microkernels*, in Proceedings of the 14th EuroSys Conference, 2019, https://dl.acm.org/doi/10.1145/3302424.3303946

[5] Intel: *Intel 64 and IA-32 Architectures Software Developer's Manual*, Volume 3: System Programming Guide, Chapter 8: User Interrupts, March 2025

[6] Du D., Hua Z., Xia Y., Zang B., Chen H.: *XPC: Architectural Support for Secure and Efficient Cross Process Call*, in Proceedings of the 46th International Symposium on Computer Architecture (ISCA), ACM, 2019, https://dl.acm.org/doi/10.1145/3307650.3322218

Thank you!

Questions?