HARDWARE/SOFTWARE CO-DESIGN FOR EFFICIENT MICROKERNEL EXECUTION

Martin Děcký
martin.decky@huawei.com

February 2019

Who Am I

- **Passionate programmer and operating systems enthusiast**
  - With a specific inclination towards multiserver microkernels
- **HelenOS developer since 2004**
- **Research Scientist from 2006 to 2018**
  - Charles University (Prague), Distributed Systems Research Group
- **Senior Research Engineer since 2017**
  - Huawei Technologies (Munich), German Research Center, Central Software Institute, OS Kernel Lab

MICROKERNEL MULTISERVER SYSTEMS ARE BETTER THAN MONOLITHIC SYSTEMS

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."*

HUAWEI

PROBLEM STATEMENT

Problem Statement

- **Microkernel design ideas go as back as 1969**
  - RC 4000 Multiprogramming System nucleus (Per Brinch Hansen)
    - Isolation of unprivileged processes, inter-process communication, hierarchical control
  - Even after 50 years they are not fully accepted as mainstream
- **Hardware and software used to be designed independently**
  - Designing CPUs used to be an extremely complicated and costly process
  - Operating systems used to be written after the CPUs were designed
  - Hardware designs used to be rather conservative

HUAWEI

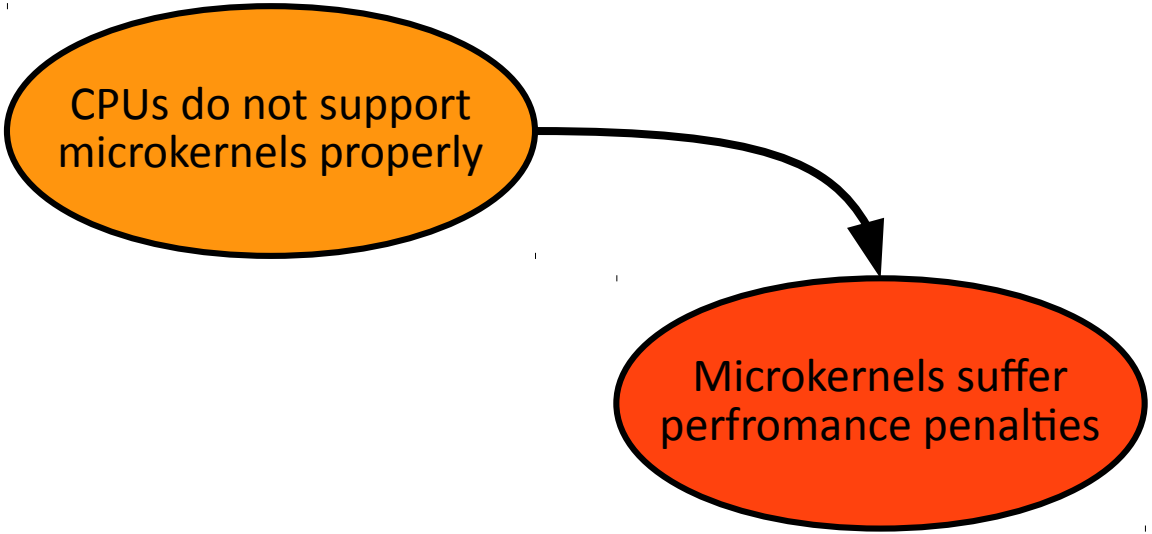
Problem Statement (2)

- **Mainstream ISAs used to be designed in a rather conservative way**
  - Can you name some really revolutionary ISA features since *IBM System/370 Advanced Function*?
  - Requirements on the new ISAs usually follow the needs of the mainstream operating systems running on the past ISAs
- **No wonder microkernels suffer performance penalties compared to monolithic systems**
  - The more fine-grained the architecture, the more penalties it suffers
  - **Let us design the hardware with microkernels in mind!**

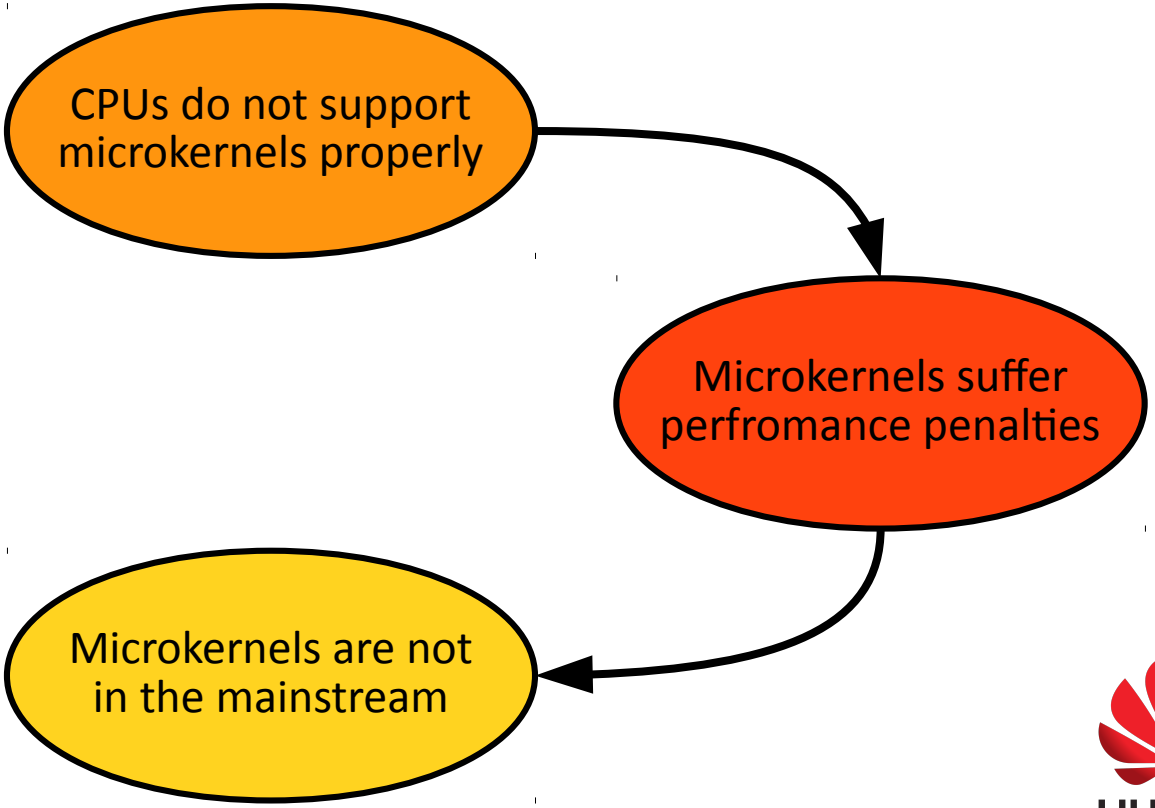
HUAWEI

The Vicious Cycle

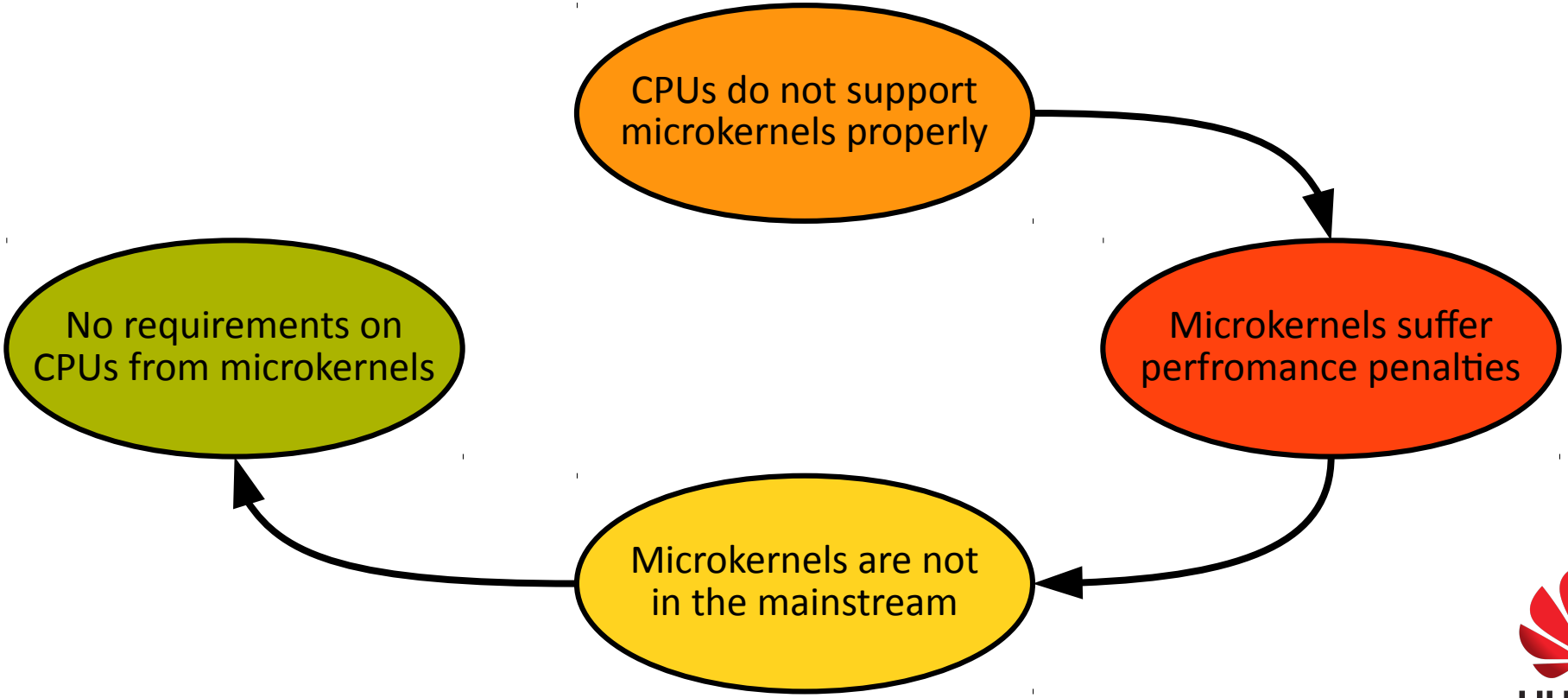
The Vicious Cycle



The Vicious Cycle

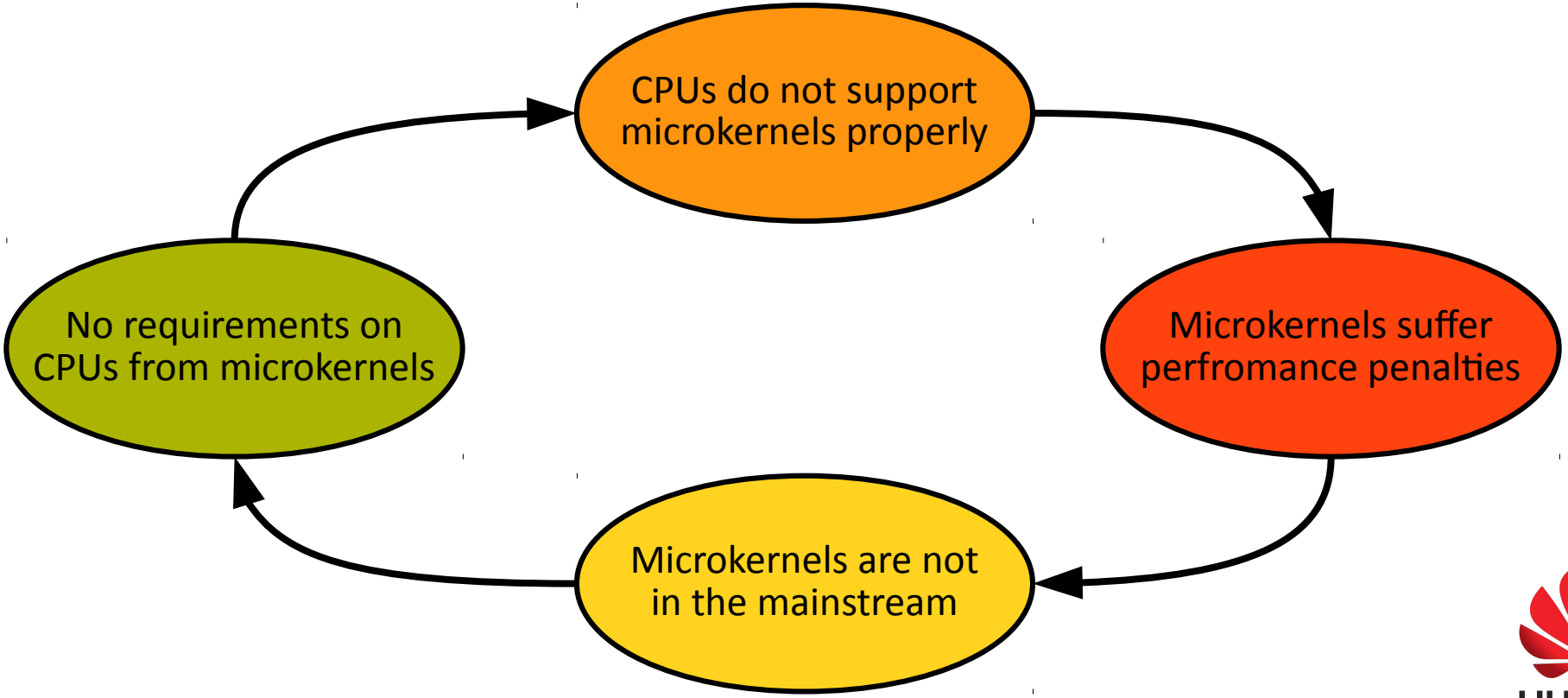


The Vicious Cycle



HUAWEI

The Vicious Cycle



HUAWEI

ANY IDEAS?

Communication between Address Spaces

- **Control and data flow between subsystems**
  - Monolithic kernel
    - Function calls
      - Passing arguments in registers and on the stack
      - Passing direct pointers to memory structures
  - Multiserver microkernel
    - IPC via microkernel syscalls
      - Passing arguments in a subset of registers
      - Privilege level switch, address space switch
      - Scheduling (in case of asynchronous IPC)
      - Data copying or memory sharing with page granularity

HUAWEI

Communication between Address Spaces (2)

- **Is the kernel round-trip of the IPC necessary?**
  - Suggestion for synchronous IPC: Extended *Jump*/*Call* and *Return* instructions that also switch the address space
    - Communicating parties identified by a "call gate" (capability) containing the target address space and the PC of the IPC handler (implicit for return)
      - Call gates stored in a TLB-like hardware cache (CLB)
      - CLB populated by the microkernel similarly to TLB-only memory management architecture
  - Suggestion for asynchronous IPC: Using CPU cache lines as the buffers for the messages
    - *Async Jump*/*Call*, *Async Return* and *Async Receive* instructions
    - Using the CPU cache like an extended register stack engine

HUAWEI

Communication between Address Spaces (3)

- **Bulk data**
  - Observation: Memory sharing is actually quite efficient for large amounts of data (multiple pages)
    - Overhead is caused primarily by creating and tearing down the shared pages
    - Data needs to be page-aligned
  - Sub-page granularity and dynamic data structures
    - Suggestion: Using CPU cache lines as shared buffers
      - Much finer granularity than pages (typically 64 to 128 bytes)
      - A separate virtual-to-cache mapping mechanism before the standard virtual-to-physical mapping

Fast Context Switching

- **Current microsecond-scale latency hiding mechanisms**
  - Hardware multi-threading
    - Effective
    - Does not scale beyond a few threads
  - Operating system context switching
    - Scales for any thread count
    - Too slow (order of 10 µs)
- **Goal: Finding a sweet spot between the two mechanisms**

Fast Context Switching (2)

- **Suggestion: Hardware cache for contexts**
  - Again, similar mechanism to TLB-only memory management
  - Dedicated instructions for context store, context restore, context switch, context save, context load
    - Context data could be potentially ABI-optimized
  - Autonomous mechanism for event-triggered context switch (e.g. external interrupt)
  - Efficient hardware mechanism for latency hiding
    - The equivalent of fine/coarse-grained simultaneous multithreading
      - The software scheduler is in charge of setting the scheduler policy
      - The CPU is in charge of scheduling the contexts based on ALU, cache and other resource availability

HUAWEI

User Space Interrupt Processing

- **Extension of the fast context switching mechanism**
  - Efficient delivery of interrupt events to user space device drivers
    - Without the routine microkernel intervention
  - An interrupt could be directly handled by a preconfigured hardware context in user space
    - A clear path towards moving even the timer interrupt handler and the scheduler from kernel space to user space
    - Going back to interrupt-driven handling of peripherals with extreme low latency requirements (instead of polling)
  - The usual pain point: Level-triggered interrupts
    - Some coordination with the platform interrupt controller is probably needed to automatically mask the interrupt source

Capabilities as First-Class Entities

- **Capabilities as unforgeable object identifiers**
  - But eventually each access to an object needs to be bound-checked and translated into the (flat) virtual address space
  - Suggestion: Embedding the capability reference in pointers
    - RV128 (128-bit variant of RISC-V) would provide 64 bits for the capability reference and 64 bits for object offset
      - 128-bit flat pointers are probably useless anyway
  - Besides the (somewhat narrow) use in the microkernel, this could be useful for other purposes
    - Simplifying the implementation of managed languages' VMs
    - Working with multiple virtual address spaces at once

HUAWEI

Prior Art

- **Nordström S., Lindh L., Johansson L., Skoglund T.: *Application Specific Real-Time Microkernel in Hardware*, 14th IEEE-NPSS Real Time Conference, 2005**
  - Offloading basic microkernel operations (e.g. thread creation, context switching) to hardware shown to improve performance by 15 % on average and up to 73 %
    - This was a coarse-grained approach
- **Hardware message passing in Intel SCC and Tilera TILE-G64/TILE-Pro64**
  - Asynchronous message passing with tight software integration

Prior Art (2)

- **Hajj I. E,, Merritt A., Zellweger G., Milojicic D., Achermann R., Faraboschi P., Hwu W., Roscoe T., Schwan K.: *SpaceJMP: Programming with Multiple Virtual Address Spaces*, 21st ACM ASPLOS, 2016**
  - Practical programming model for using multiple virtual address spaces on commodity hardware (evaluated on DragonFly BSD and Barrelfish)
    - Useful for data-centric applications for sharing large amounts of memory between processes
- **Intel IA-32 Task State Segment (TSS)**
  - Hardware-based context switching
  - Historically, it has been used by Linux
    - The primary reason for removal was not performance, but portability

HUAWEI

Prior Art (3)

- **Intel VT-x VM Functions (VMFUNC)**
  - Efficient cross-VM function calls
    - Switching the EPT and passing register arguments
    - Current implementation limited to 512 entry points
    - Practically usable even for very fine-grained virtualization with the granularity of individual functions
      - Liu Y., Zhou T., Chen K., Chen H., Xia Y.: *Thwarting Memory Disclosure with Efficient Hypervisor-enforced Intra-domain Isolation*, 22nd ACM SIGSAC Conference on Computer and Communications Security, 2015
        - *"The cost of a VMFUNC is similar with a syscall"*
        - *"… hypervisor-level protection at the cost of system calls"*
  - **SkyBridge paper to appear at EuroSys 2019**

HUAWEI

Prior Art (4)

- **Woodruff J., Watson R. N. M., Chisnall D., Moore S., Anderson J., Davis B., Laurie B., Neumann P. G., Norton R., Roe. M.: *The CHERI capability model: Revisiting RISC in the an age of risk*, 41st ACM Annual International Symposium on Computer Architecture, 2014**
  - Hardware-based capability model for byte-granularity memory protection
  - Extension of the 64-bit MIPS ISA
    - Evaluated on an extended MIPS R4000 FPGA soft-core
    - 32 capability registers (256 bits)
  - Limitation: Inflexible design mostly due to the tight backward compatibility with a 64-bit ISA
- **Intel MPX**
  - Several design and implementation issues, deemed not production-ready

HUAWEI

Summary

- **Traditionally, hardware has not been designed to accommodate the requirements of microkernel multiserver operating systems**
  - Microkernels thus suffer performance penalties
    - This prevented them from replacing monolithic operating systems and closed the vicious cycle
- **Hardware design is hopefully becoming more accessible and democratic**
  - E.g. RISC-V
- **Co-designing the hardware and software might help us gain the benefits of the microkernel multiserver design with no performance penalties**
  - However, it requires some out-of-the-box thinking

HUAWEI

Acknowledgements

- **OS Kernel Lab at Huawei Technologies**
  - Javier Picorel
  - Haibo Chen

Huawei Dresden R&D Lab

- **Focusing on microkernel research, design and development**
  - Basic research
  - Applied research
  - Prototype development
  - Collaboration with academia and other technology companies
- **Looking for senior operating system researchers, designers, developers and experts**
  - Previous microkernel experience is a big plus
  - *"A startup within a large company"*
  - Shaping the future product portfolio of Huawei
    - **Including hardware/software co-design via HiSilicon**

Q&A


HUAWEI
THANK YOU!