ETH zürich

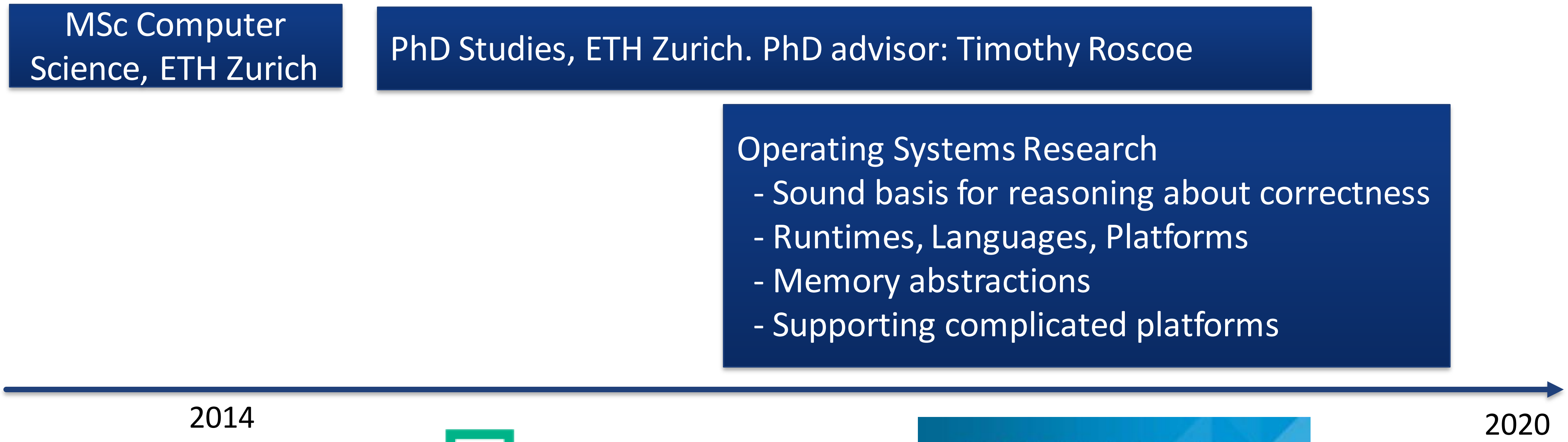
Reto Achermann

Realistic Hardware Abstractions and Least-Privilege Memory Management in Operating Systems

PhD Student, Systems Group, Department of Computer Science, ETH Zurich

Systems@ETH zürich

$ whoami



MSc Computer Science, ETH Zurich

PhD Studies, ETH Zurich. PhD advisor: Timothy Roscoe

Operating Systems Research
- Sound basis for reasoning about correctness
- Runtimes, Languages, Platforms
- Memory abstractions
- Supporting complicated platforms

2014

2020

Industry Internships

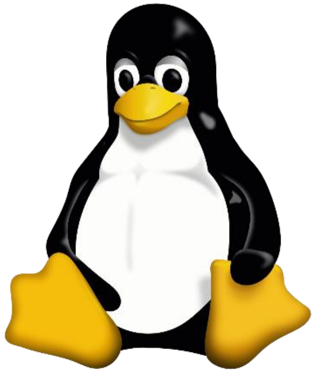
Hewlett Packard Labs

vmware RESEARCH

My Operating Systems Coding Experiences

- Platform support: Cavium ThunderX / Xeon Phi co-processor / ARM FastModels
- Drivers: USB Stack / DMA Engines / Xeon Phi / CPU / IOMMU
- Runtimes: OpenMP / libnuma / message-passing
- Extensions to the capability system
- Domain Specific Languages

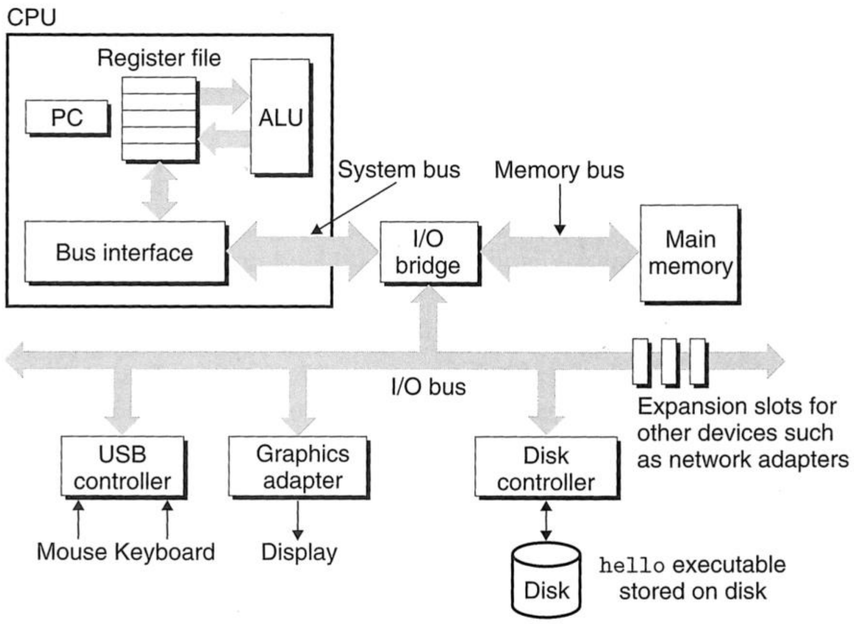
- Memory management
- Message-passing system
- Memory allocation policies
- Page-table replication
- Kernel modules

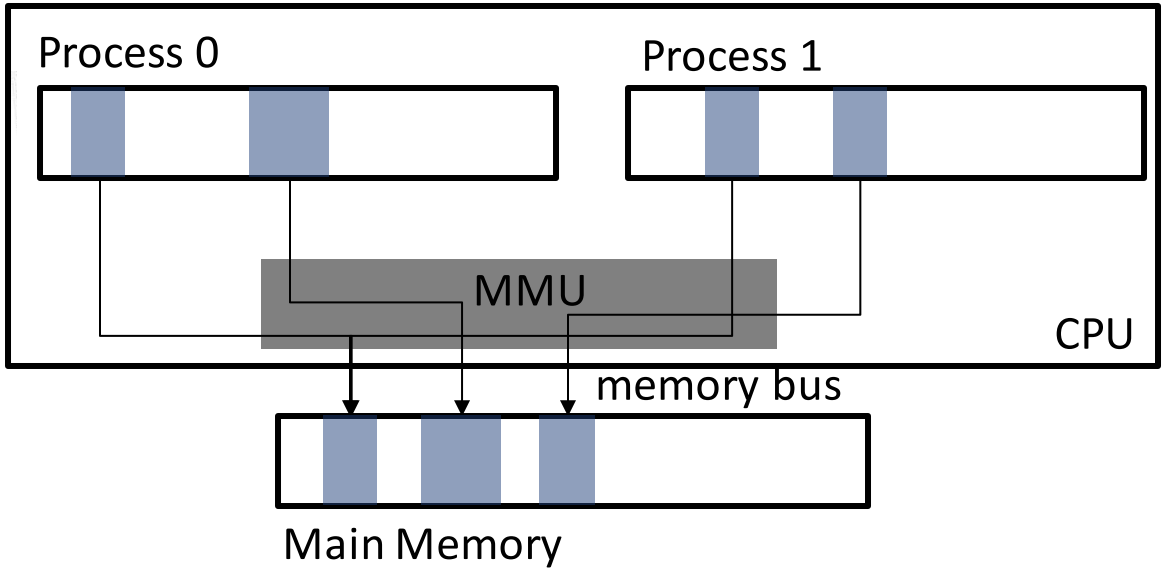


A deep **dissatisfaction** about the way operating systems abstract and represent hardware.

Computer Architecture 101

Figure 1.4
Hardware organization of a typical system. CPU: Central Processing Unit, ALU: Arithmetic/Logic Unit, PC: Program counter, USB: Universal Serial Bus.





systems, but all systems have a similar look and feel. Don’t worry about the complexity of this figure just now. We will get to its various details in stages throughout the course of the book.

The OS just runs on a new platform

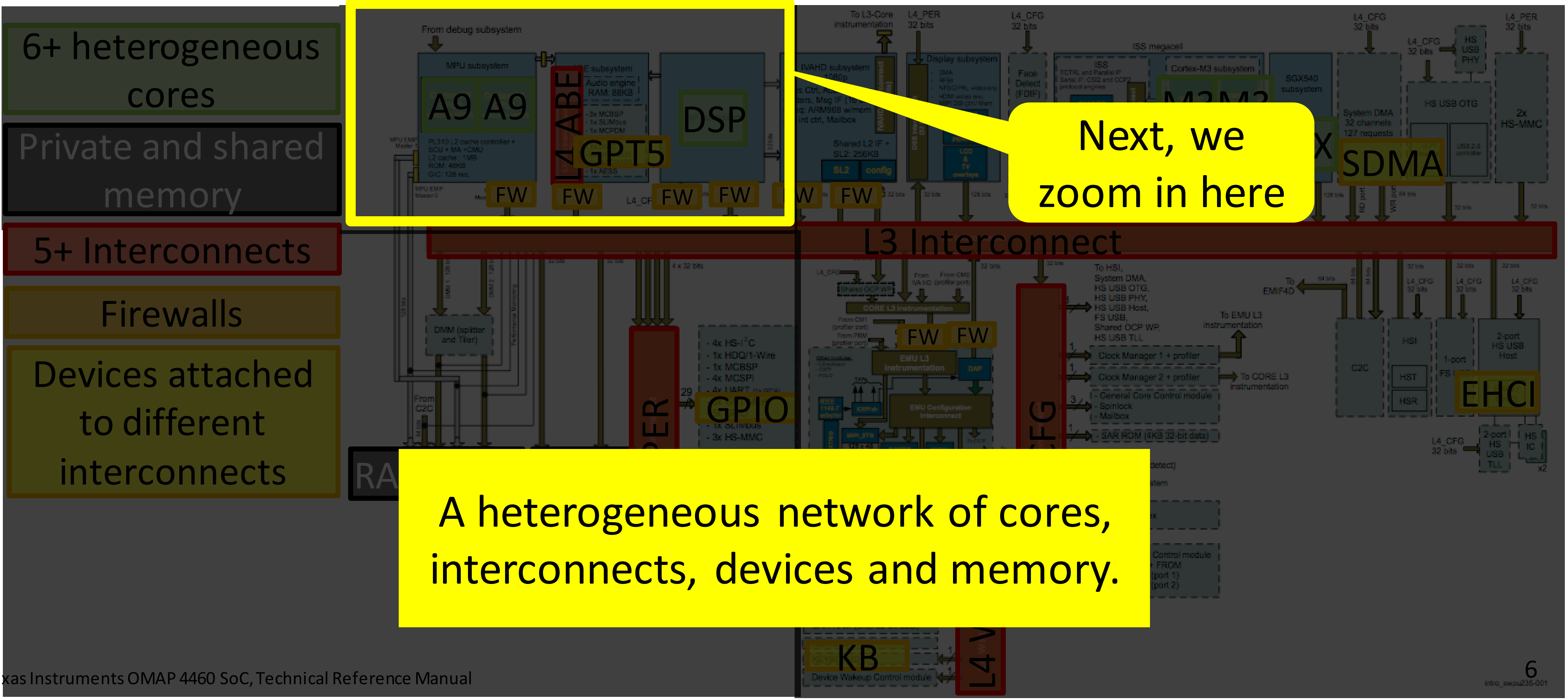
Operating System Assumptions:

- A single, flat physical address space
- physical address as a unique identifier
- homogeneous views from all CPUs and devices

Computer Systems, A Programmer’s Perspective by Randal E. Bryant and David R. O’Hallaron

Reality: Hardware Violates the Assumptions Made by Operating Systems

TexasInstruments OMAP4460, Q4 2011



6+ heterogeneous cores

Private and shared memory

5+ Interconnects

Firewalls

Devices attached to different interconnects

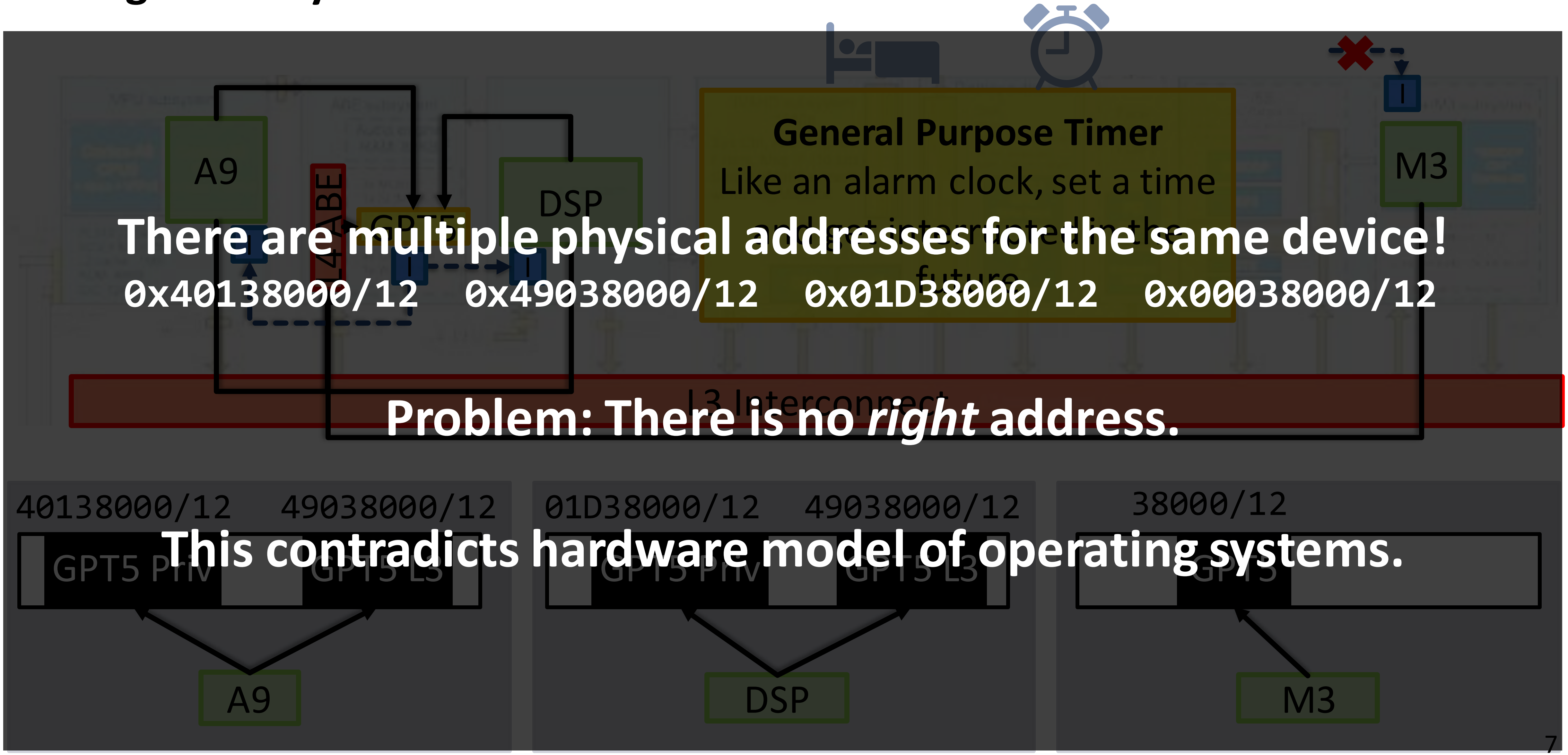
Next, we zoom in here

A heterogeneous network of cores, interconnects, devices and memory.

Texas Instruments OMAP 4460 SoC, Technical Reference Manual

Ambiguous Physical Addresses and Non-Uniform Views

OMAP 4460 SoC



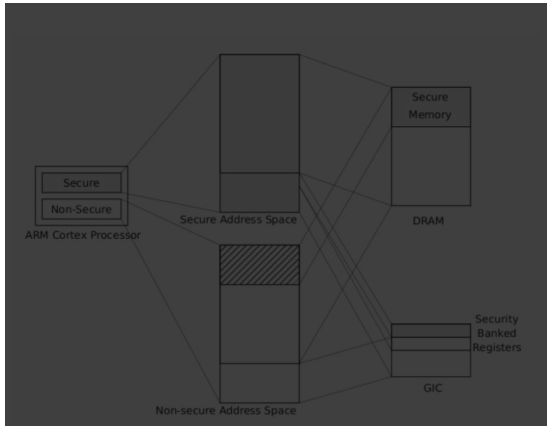
There are multiple physical addresses for the same device!

`0x40138000/12   0x49038000/12   0x01D38000/12   0x00038000/12`

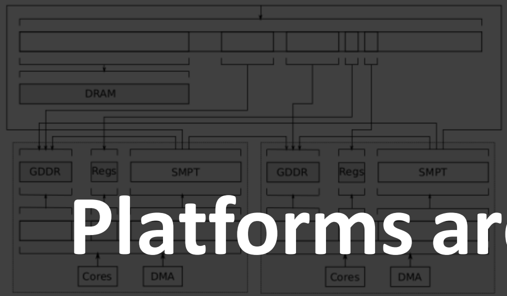
Problem: There is no *right* address.

This contradicts hardware model of operating systems.

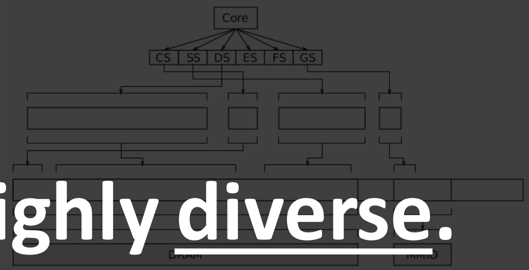
Complicated Memory Topologies are a Universal Problem



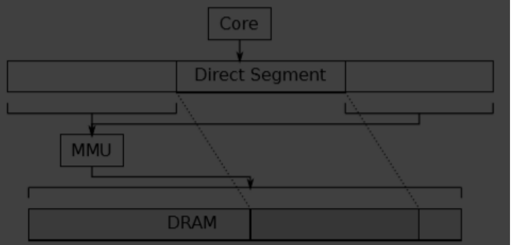
Secure and non-secure (ARM TrustZone / Intel SGX)



Co-Processors (Intel Xeon Phi)



Segmentation (x86, Pentium)

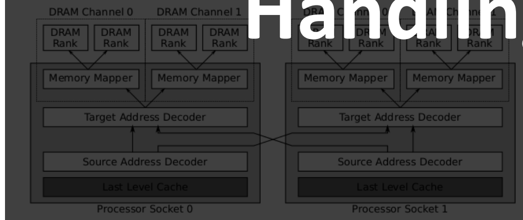


Direct Segments

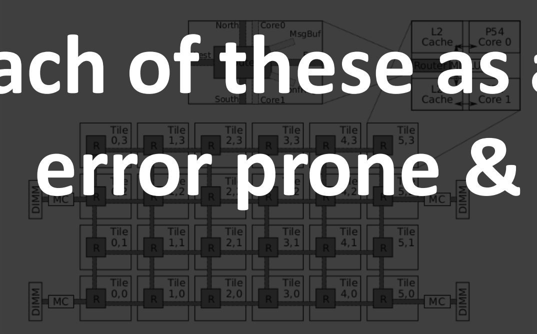
Platforms are highly diverse.

Address translations are configurable.

Handling each of these as a special case does not scale: error prone & time consuming.



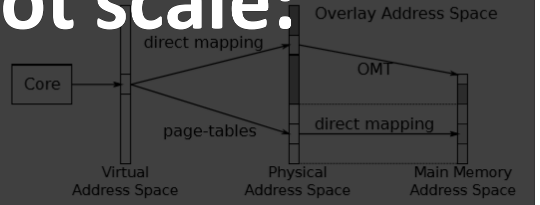
Memory Controller remappings



Intel Single Chip Cloud Computer

DRAM
Controller
DRAM
Controller
DRAM
Controller
Controller
Controller
Controller
Controller
NVM
NVM
NVM
NVM

Fabric Attached Memory (GenZ / The Machine)



Page Overlays

Mismatch: Hardware Abstraction in Operating System⇔ Real Hardware

This mismatch is a **problem** – the OS does not seem to get it right

Bugs and vulnerabilities in systems software:

- `CVE-2014-3601:` Miscalculation of affected pages
- `CVE-2016-5349:` Not enough memory address information provided
- `CVE-2017-8061:` Wrong DMA addresses
- `CVE-2017-16994:` Ignoring holes in huge-pages
- `CVE-2014-9888:` wrong access rights for data pages
- `CVE-2019-2250:` authorization bug allows writing to memory locations
- `CVE-2018-11994:` SMMU misconfiguration allows access to memory
- 30% of patches to Linux memory manager are bugfixes

My Research Focus

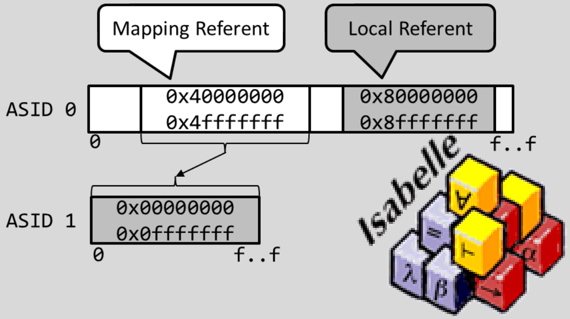
My goal:
A sound basis for reasoning about reliable operating system on any hardware platform.

Design and Implementation of Operating Systems

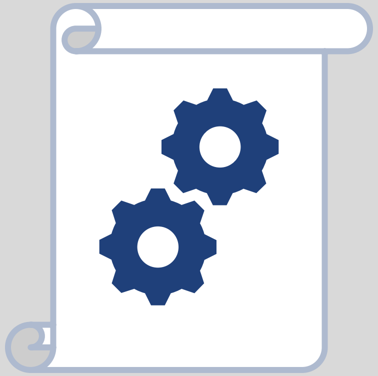
Domain Specific Languages for Systems Engineering

Applying Formal Methods to Operating Systems and Hardware Specification

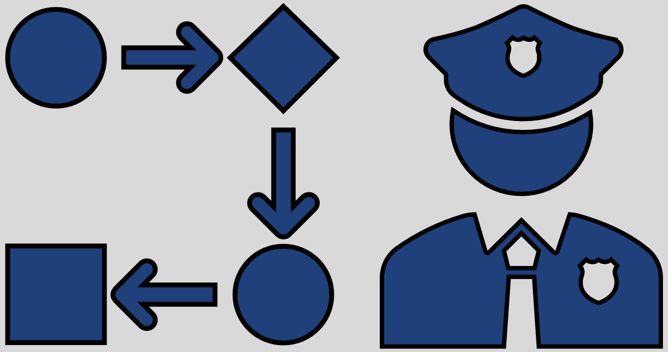
Realistic Hardware Abstractions and Least-Privilege Memory Management in Operating Systems



A new model to express memory addressing on modern machines



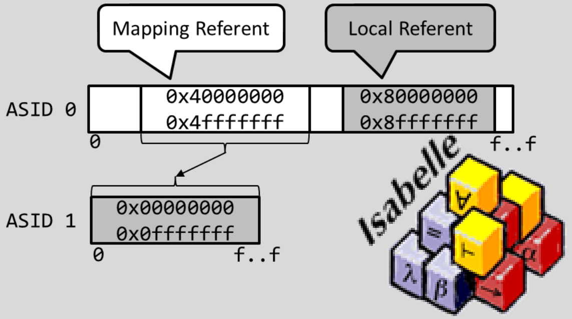
Generate OS code and hardware configuration



Express changes of the configuration and the required authority

Efficient Implementation in an Operating System

Realistic Hardware Abstractions and Least-Privilege Memory Management in Operating Systems



A new model to express memory addressing on modern machines

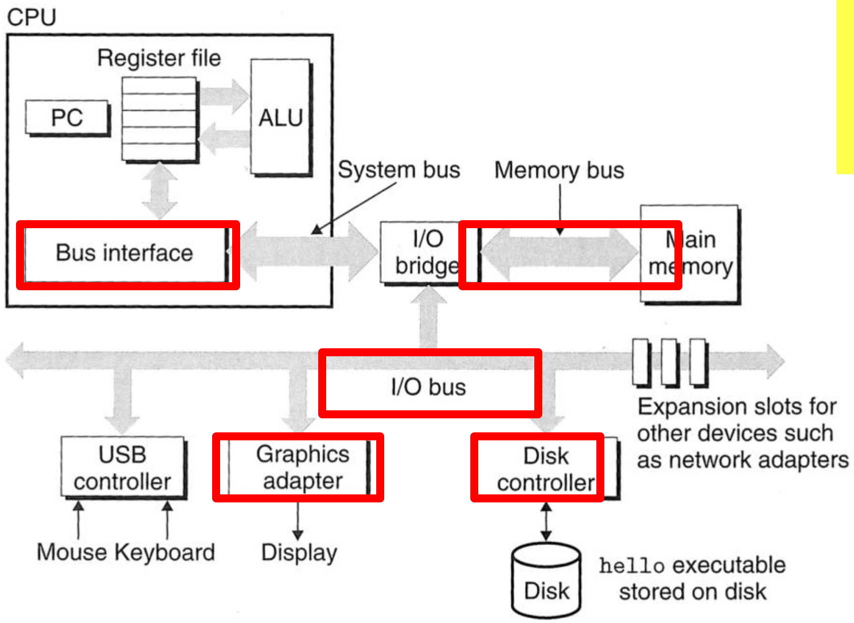
Generate OS code and hardware configuration

Express changes of the configuration and the required authority

Efficient Implementation in an Operating System

A More Realistic View of the Platform

Figure 1.4
Hardware organization of a typical system. CPU: Central Processing Unit, ALU: Arithmetic/Logic Unit, PC: Program counter, USB: Universal Serial Bus.



Configurable I/O bus:
- PCI bridge programming
- PCI hot-plug
- I/O MMUs / System MMUs
- Virtualization

Interconnects:
- Configurable, multi-stage translations
- Configurable Firewalls
- Private access ports

Processors
- Core-Local resources
- Cache Hierarchy, operation modes
- Configurable Multi-Stage Translation
- Virtualization

Devices
- Memory access restrictions
- Configurable Translations
- Self-virtualization

The OS must **correctly** configure these

The OS needs to be ported:
- Requires engineering effort
- Manual process: source of bugs (more engineering effort)

Computer Systems, A Programmer's Perspective by Randal E. Bryant and David R. O'Hallaron

Specification and Modeling Hardware

- Industry Standards: DeviceTrees / UEFI / ACPI / USB / PCI Express
  - Limited topology information, not available everywhere
  - Assumptions: a uniform view & unique physical addresses

→ Insufficient for accurate hardware representation

- Memory & Processor Models: ARM's ASL and Sewell et al.
  - Model the behavior of instructions and memory requests
  - Stop at processor boundaries

→ Complimentary problem orthogonal to address translation

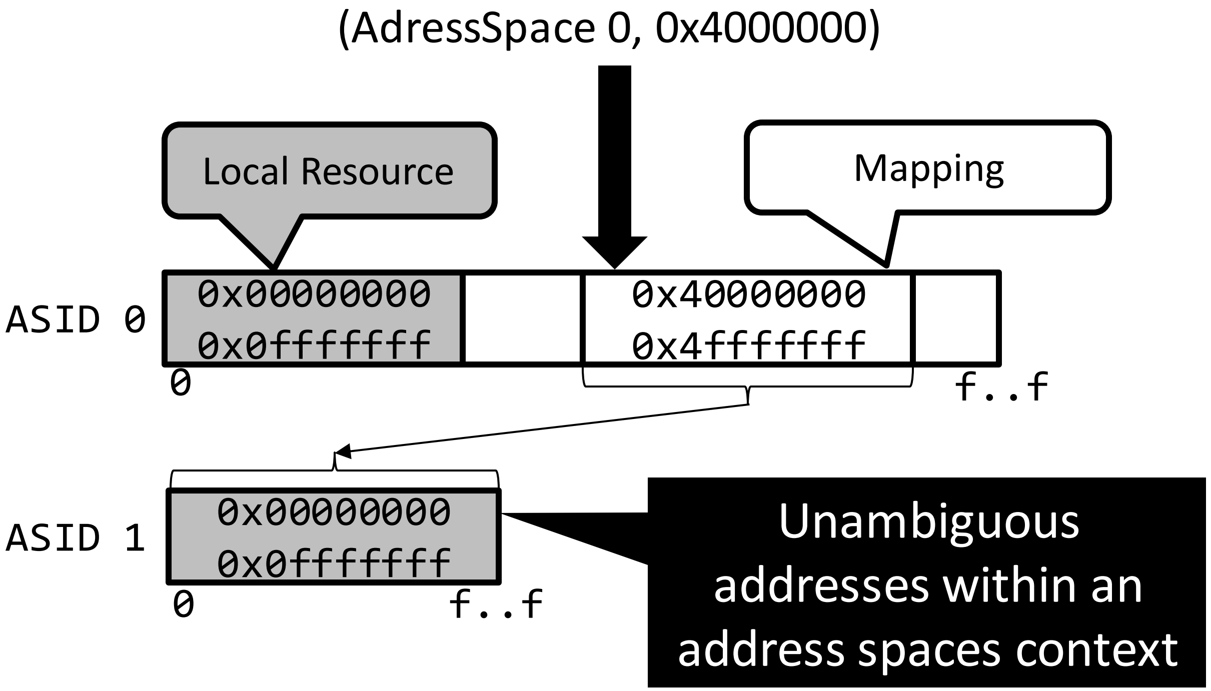
- Verified Operating Systems: seL4, CertiKOS
  - Proofs based on a linear flat array to physical memory

→ Proofs need an accurate hardware representation

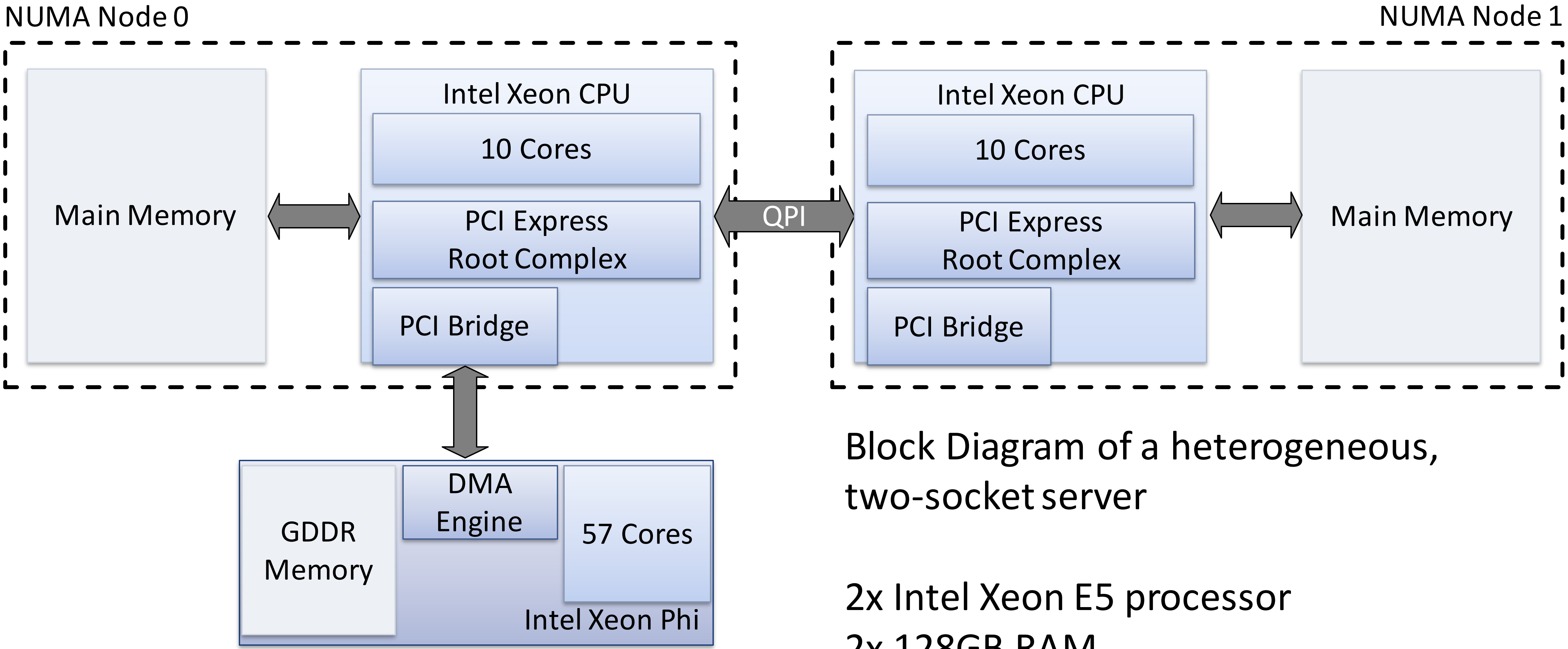
The Address Space Abstraction – A more faithful view of Memory Hardware

[HotOS'15]

- Memory management needs an unambiguous reference to physical resources.
- Address Space (**Naming Problem**)
  - Context for resolving addresses
  - Range of address values e.g. [0, 2^b)
  - Regions of address mappings
  - Regions of local resources



New Address Space:
- Ambiguous addresses
- different interpretation of addresses



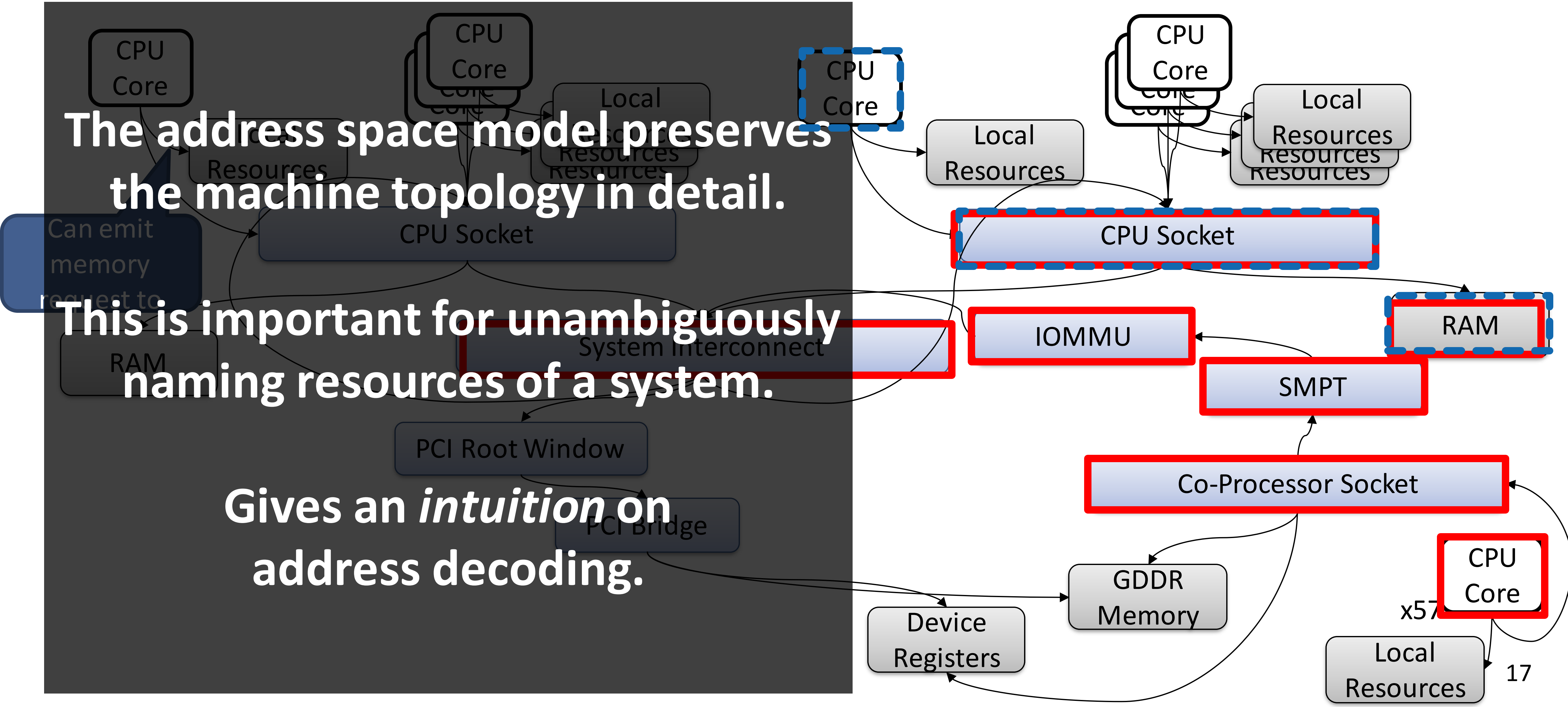
Block Diagram of a heterogeneous, two-socket server

2x Intel Xeon E5 processor
2x 128GB RAM

Intel Xeon Phi Co-Processor attached to PCI Express

ETH zürich

Decomposing a Heterogeneous, Two-Socket Server into Address Spaces



The address space model preserves the machine topology in detail.

This is important for unambiguously naming resources of a system.

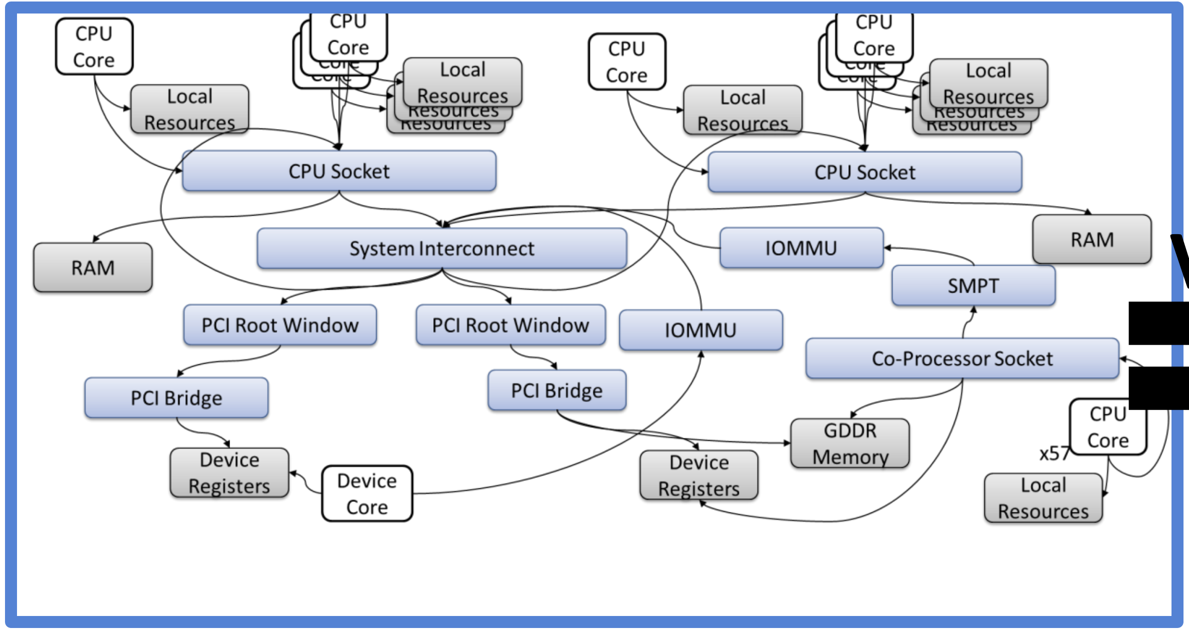
Gives an *intuition* on address decoding.

Applying Formal Methods to the Address Space Model

- Formalization of the model in Isabelle/HOL
  [Decoding Net, MARS’17 / ITP’18]
- Well-defined semantics of address resolution termination proofs, …
- Verification of algorithms on top of the model
- Capture the **current, static** state of the system

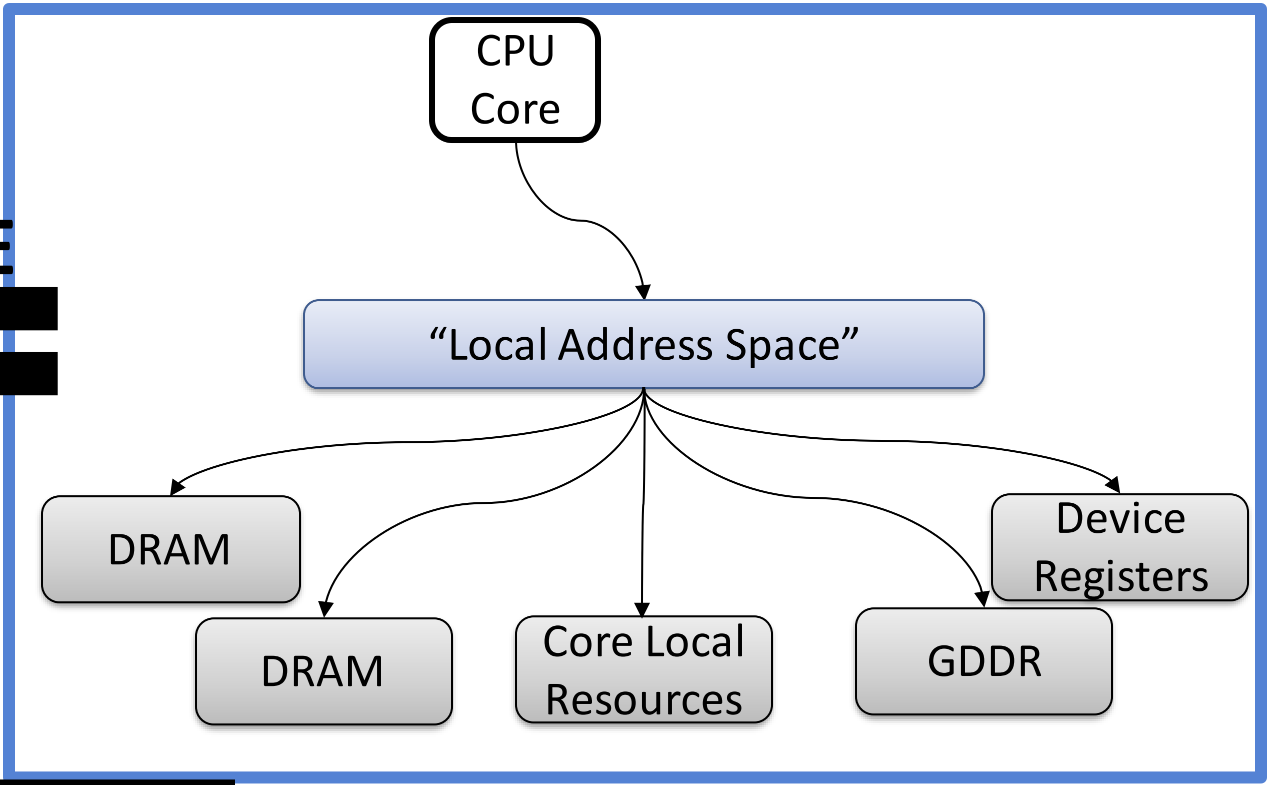
Sound foundation to express address translation of real hardware (e.g. TLB models)

“What is reachable from a core and at which address?”



VE
=

Normal Form



Proof of view-equivalence

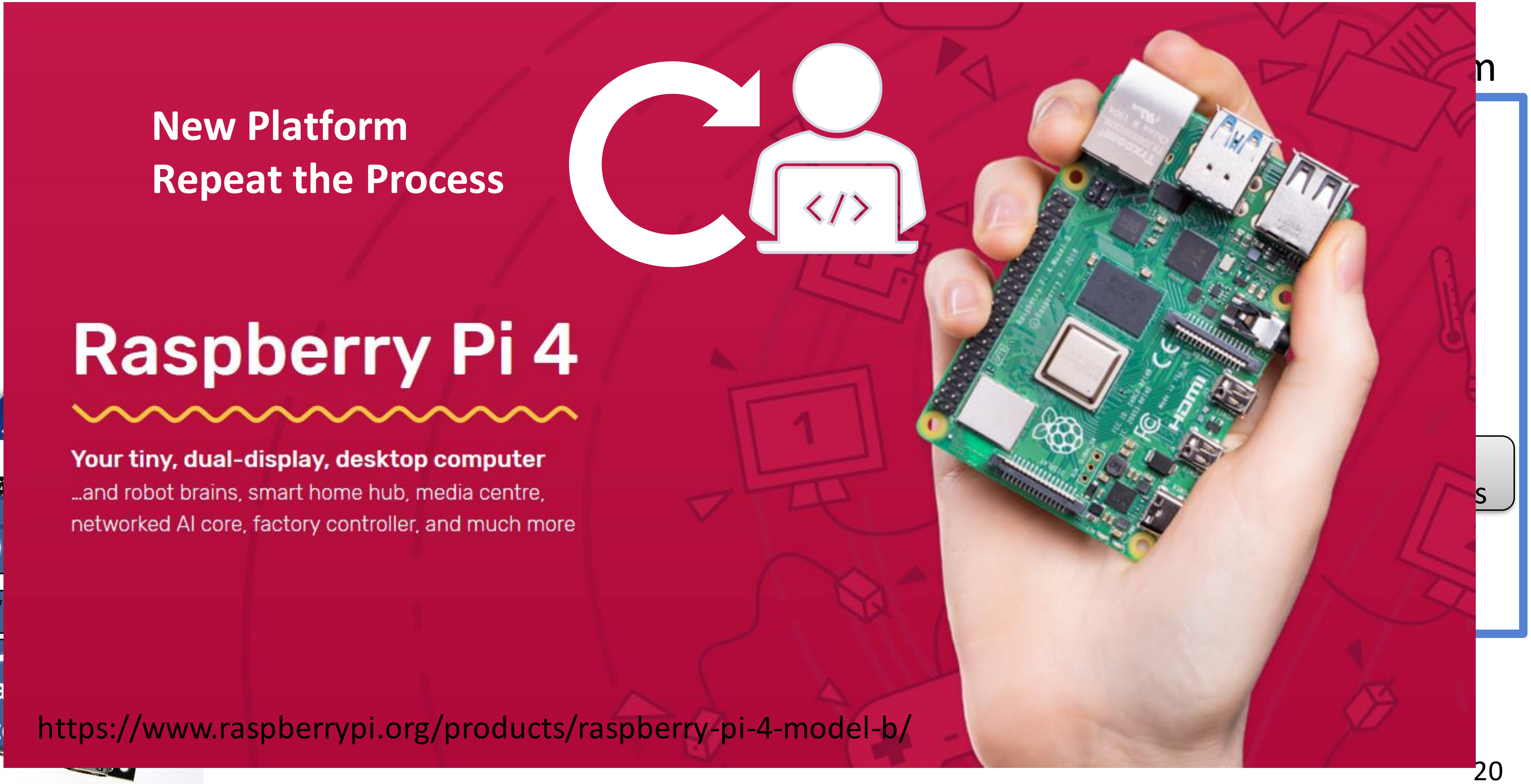
Algorithms produce provably to the correct values. Fewer bugs in the OS.

Unreachable Resources

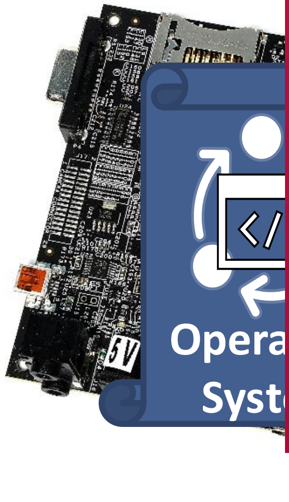
Using the Correct Output to write Platform Specific OS Code

New Platform

Repeat the Process


Raspberry Pi 4
Your tiny, dual-display, desktop computer
…and robot brains, smart home hub, media centre, networked AI core, factory controller, and much more

https://www.raspberrypi.org/products/raspberry-pi-4-model-b/



n

s

Many new SoC-Platforms are released every year

System-on-a-Chip Platform Vendors

Actions Semiconductor, Advanced Micro Devices (AMD), Advanced Semiconductor Engineering (ASE), Aeroflex Gaisler, Agate Logic, Alchip, Allwinner Technology, Altera, Amkor Technology, Amlogic, Analog Devices, Anyka, Apple Inc., Applied Micro Circuits Corporation (AMCC), ARM Holdings, ASIX Electronics, Atheros, Atmel, Axis Communications, Broadcom, Cambridge Silicon Radio, Cavium Networks, CEVA, Inc., Cirrus Logic, Conexant, Core Logic, Coronis (Wavenis Technology), Cortina Systems, CPU Tech, Cypress Semiconductor, FameG (Fulhua Microelectronics Corp.), Freescale Semiconductor, Frontier Silicon Ltd, Fujifilm, HiSilicon, Horizon Semiconductors, Imagination Technologies, Infineon Technologies, Innova Card, Intel Corporation, InvenSense, Lattice Semiconductor, Leadcore Technology, LSI Corporation, Marvell Technology Group, MediaTek, Maxim Integrated Products, Milkymist, MIPS Technologies, Mistletoe Technologies, MosChip Semiconductor Technology, MStar Semiconductor, Naksha Technologies, Nokia, NuCORE Technology, Nufront, NVIDIA, NXP Semiconductors (formerly Philips Semiconductors),
Corporation, PMC-Sier
Scorpion, Redpine Sign
Sequence Design, Shar
Silicon Motion, Skywor
SolidRun, Spreadtrum,
Sunplus Technology, Sy
Tensilica, Teridian Sem
TranSwitch, Vimicro,
RDA Microelectronics

SoC Released 2018

Apple

A12, S4, W3

Samsung

Exynos 9 Series, Exynos 7 Octa, Exynos 5 Hexa

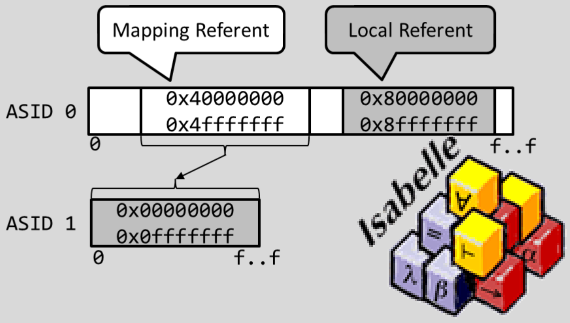
QualComm

SDM439, SDM429, SDM632, SDM670, SDM710, SDM845, QCC5120

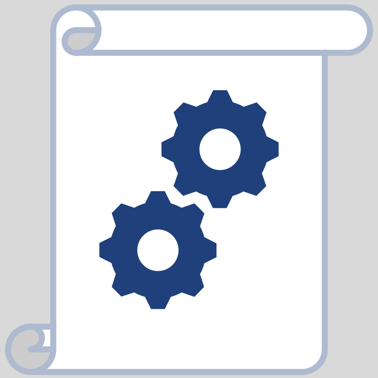
T6755S, Helio
Helio P70

Each platform is different.
Operating system needs to be adapted every time.
Manual porting:
- introduces more bugs…
- time effort

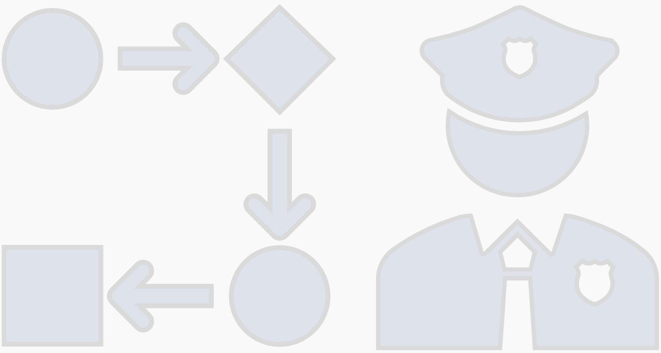
Realistic Hardware Abstractions and Least-Privilege Memory Management in Operating Systems



A new model to express memory addressing on modern machines



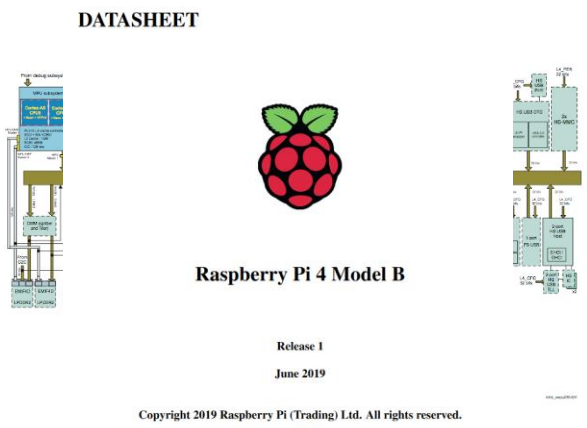
Generate OS code and hardware configuration



Express changes of the configuration and the required authority

Efficient Implementation in an Operating System

Automatic Generation of OS Code From the Model



DATASHEET

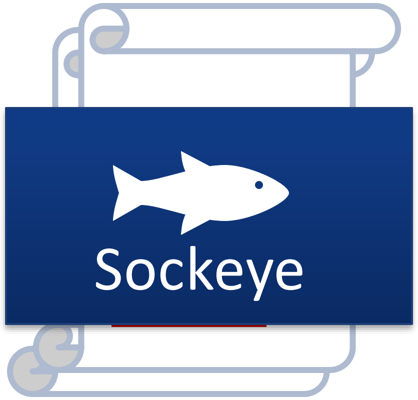
Raspberry Pi 4 Model B

Release 1

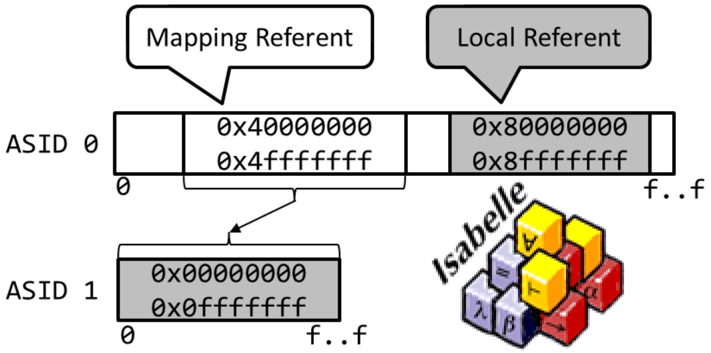
June 2019

Copyright 2019 Raspberry Pi (Trading) Ltd. All rights reserved.

Vendor supplied data (e.g. Hardware Manual)



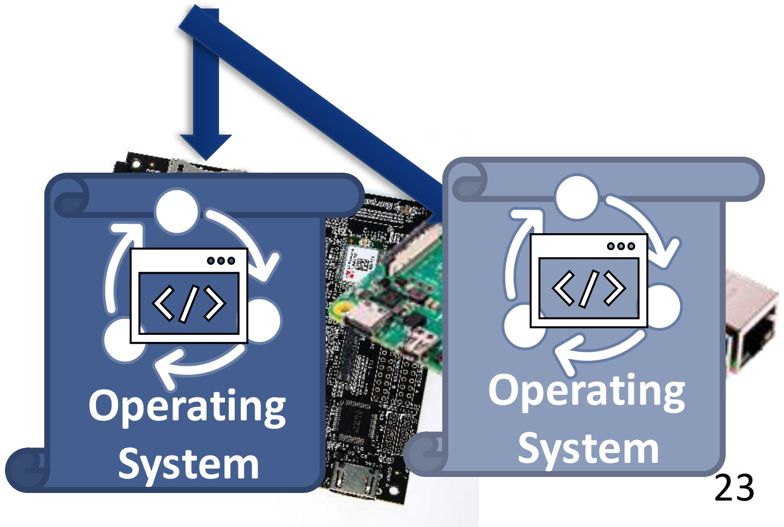
Machine readable description of the platform



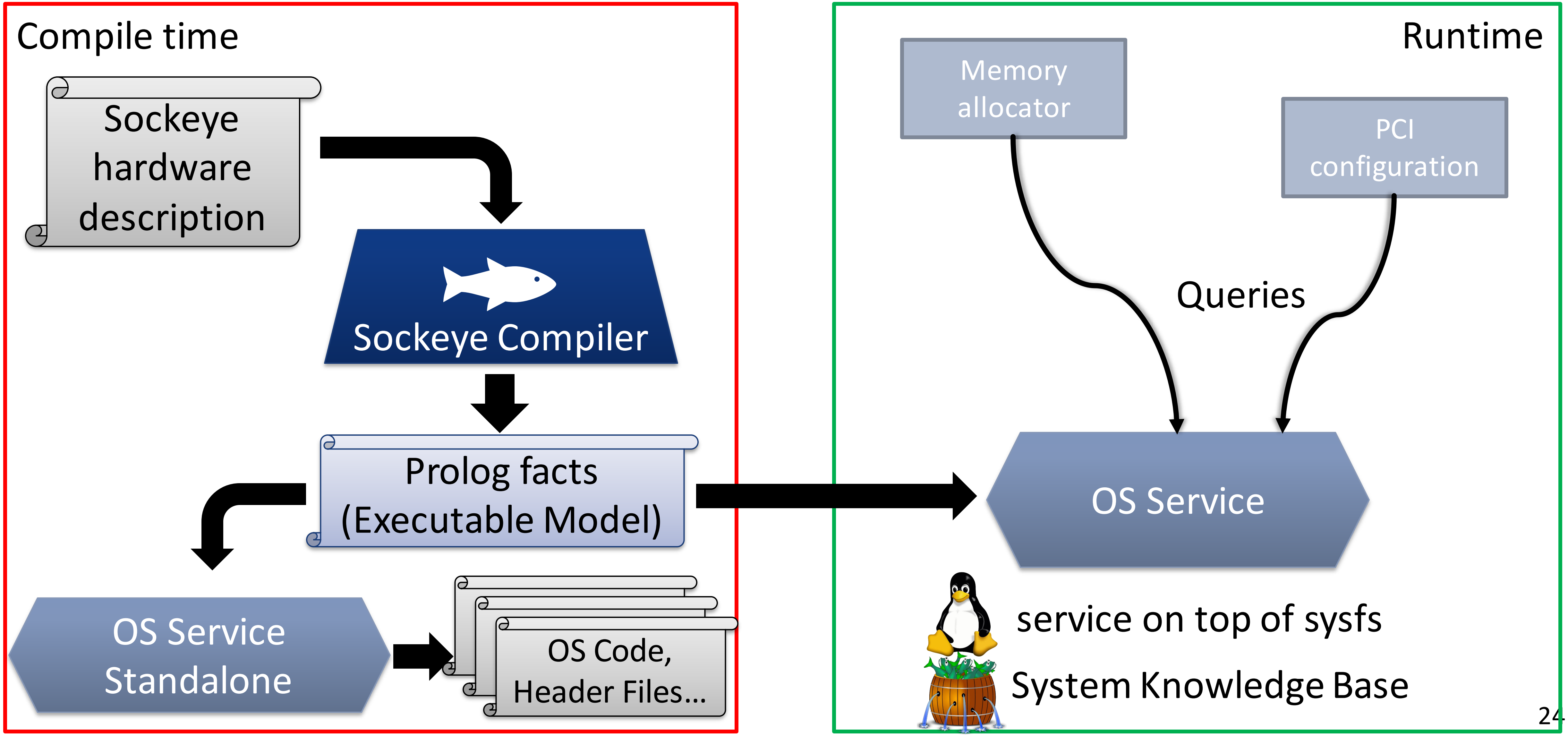
Executable representation of the model

Sockeye Domain Specific Language for Hardware Descriptions

Generation of correct low-level OS code



Toolchain



Compile time

Sockeye hardware description

Sockeye Compiler

Prolog facts (Executable Model)

OS Service Standalone

OS Code, Header Files…

Runtime

Memory allocator

PCI configuration

Queries

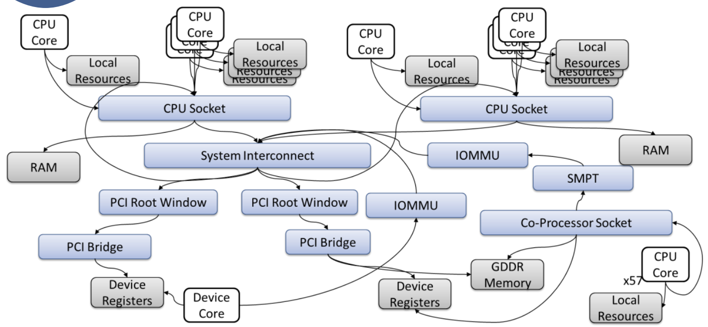
OS Service

service on top of sysfs

System Knowledge Base

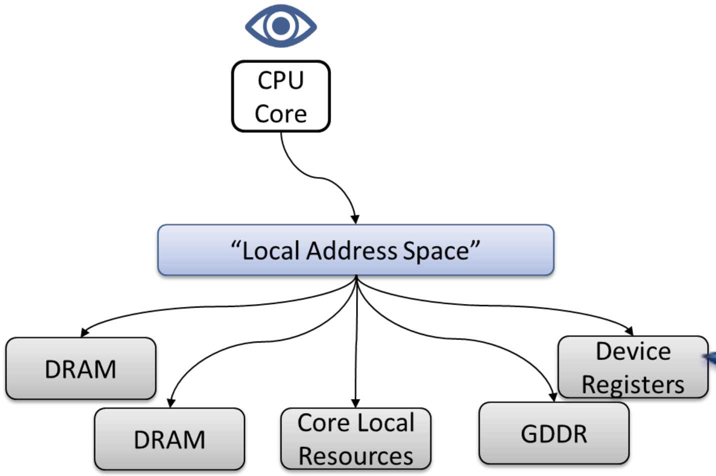
Use Case Example: Correct-by-Construction Page-Table Generation

Specify the observing core



Topology as generated by Sockeye

Flatten the decoding net using view-preserving operations



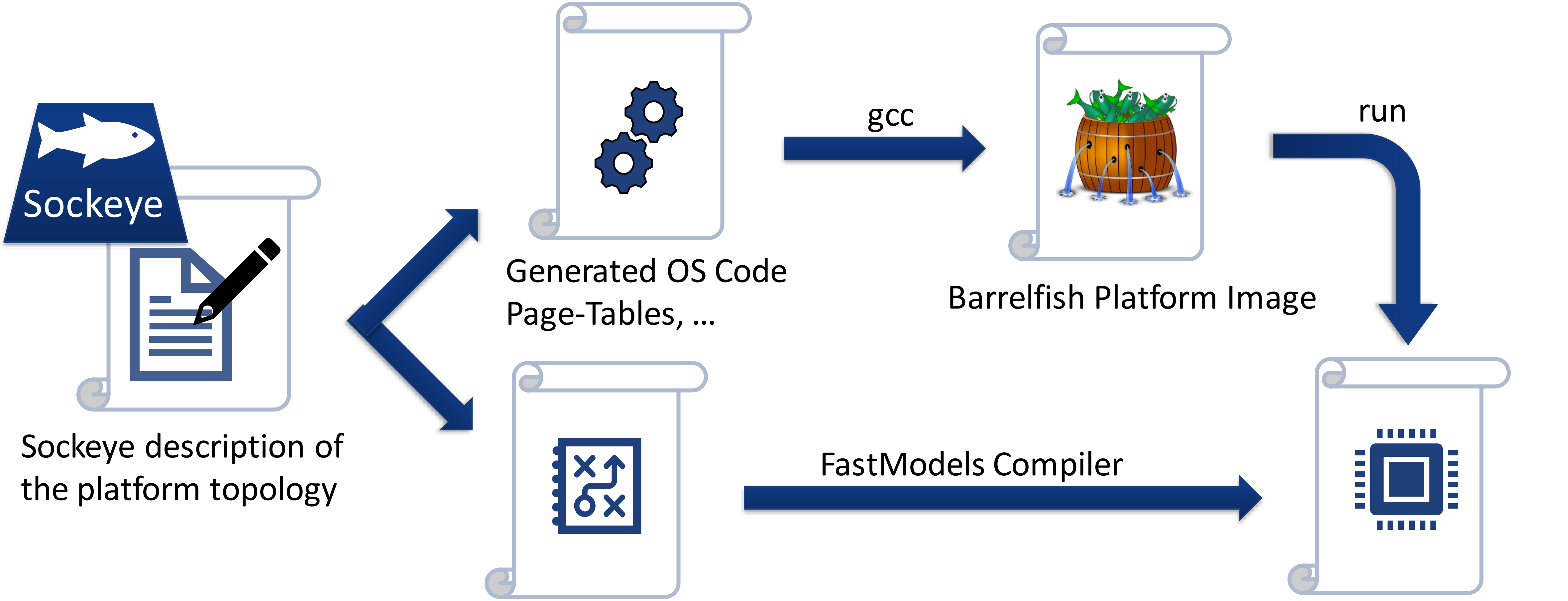
ARMv7,
ARMv8,
x86_64,
K1OM

Page-Table Binary

Page Table Receipt
Architecture = [ARMv8]
Mappings = [
  DRAM0 @ 0x8000000,
  DevRegs @ 0x10000000
]

Generate page-tables based on the flattened representation

Validation: Custom Simulation Platforms for ARM FastModels



Sockeye description of the platform topology

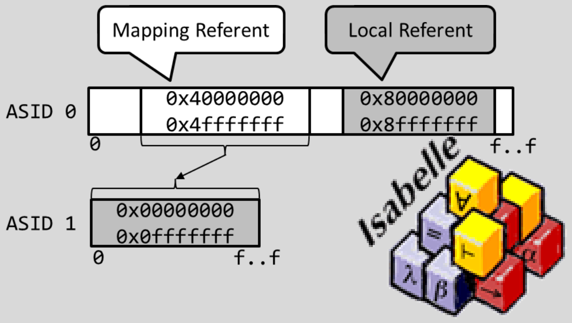
Generated OS Code
Page-Tables, …

Barrelfish Platform Image

Platform Description File (LISA)

FastModel Simulator Executable

Realistic Hardware Abstractions and Least-Privilege Memory Management in Operating Systems



A new model to express memory addressing on modern machines

Generate OS code and hardware configuration

Express changes of the configuration and the required authority

Efficient Implementation in an Operating System

Dynamic Configuration of the Decoding Net

Semantics of **dynamic** configuration of the **translate** function

Rights and authority required to **change** the translate function

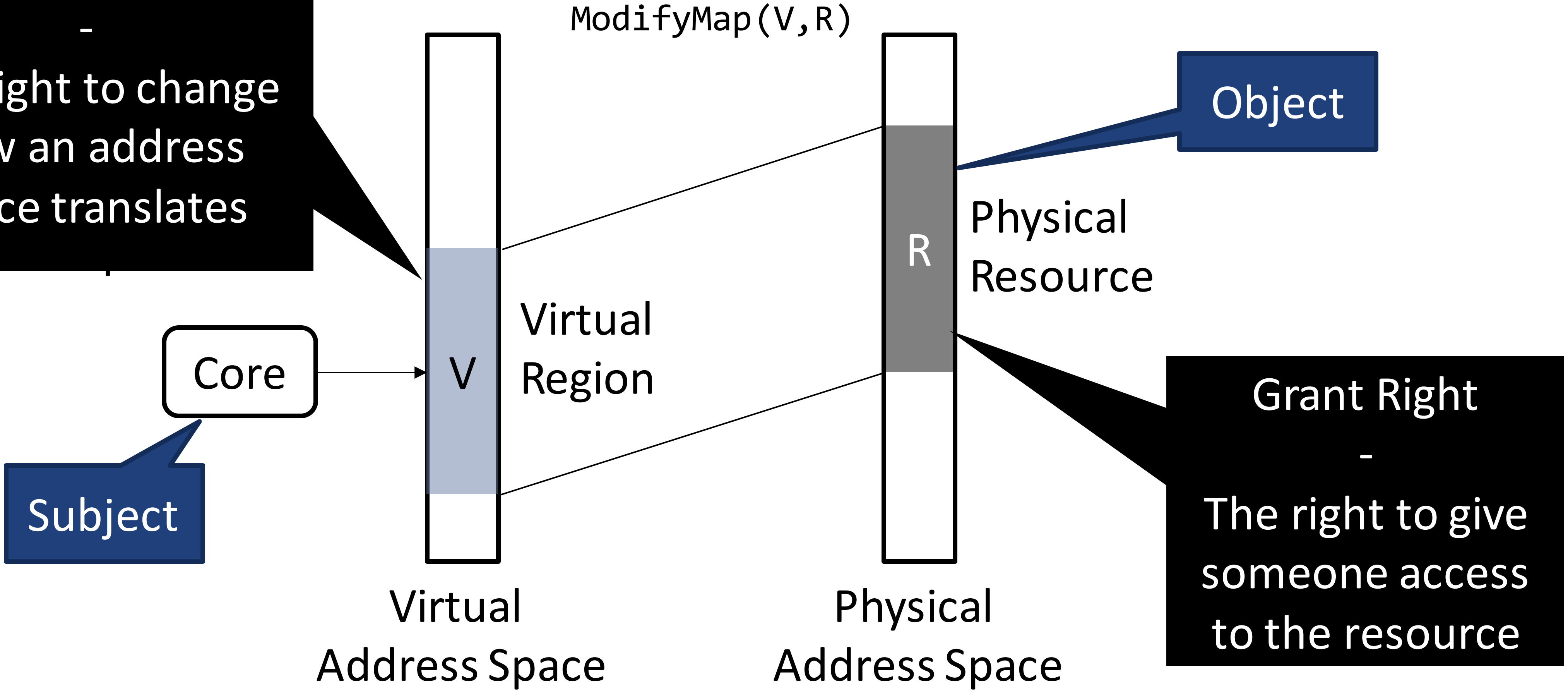
Principle of Least Privilege

Fine-Grained Decomposition of Rights and Operations

Linux:
mmap()
Kernel h

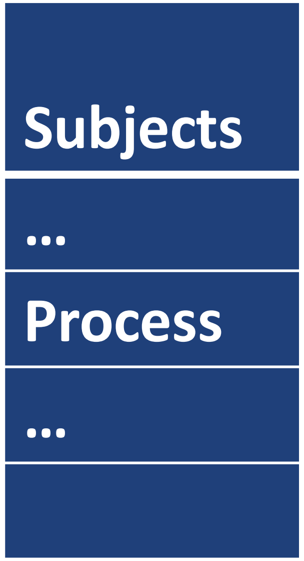
What ha
least pri

Map Right
-
The right to change how an address space translates



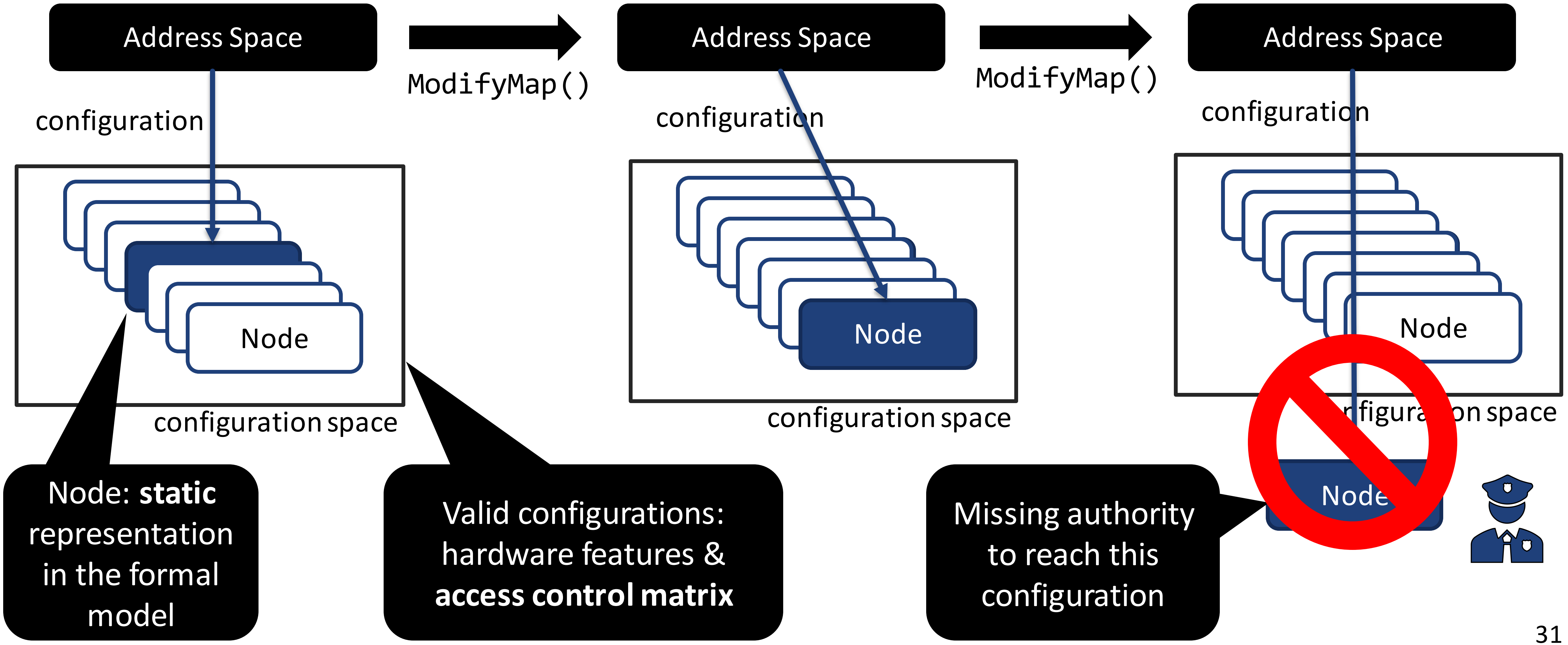
Grant Right
-
The right to give someone access to the resource

Expressing Fine-Grained Authority in a System



The access control matrix defines what
address space configurations and **transitions** are **valid**.

Dynamic Address Space Configuration



Node: **static** representation in the formal model

Valid configurations: hardware features & **access control matrix**

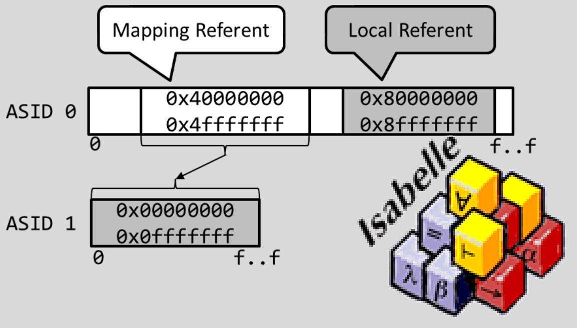
Missing authority to reach this configuration

From the Static Model to a Dynamic Implementation in Three Steps

[Submitted to ASPLOS20]

- ✓ Identification of all relevant **objects** and **subjects** and their relationship
- Development of an **executable specification** for rapid prototyping
- Executable specification as a guide for a **OS implementation**
  - Barrelfish with Multiple Address Spaces + Least-Privilege
  - Access control with Capabilities as a natural match for least privilege
  - Support for heterogeneous platforms

Realistic Hardware Abstractions and Least-Privilege Memory Management in Operating Systems



A new model to express memory addressing on modern machines

Generate OS code and hardware configuration

Express changes of the configuration and the required authority

Efficient Implementation in an Operating System

Evaluation

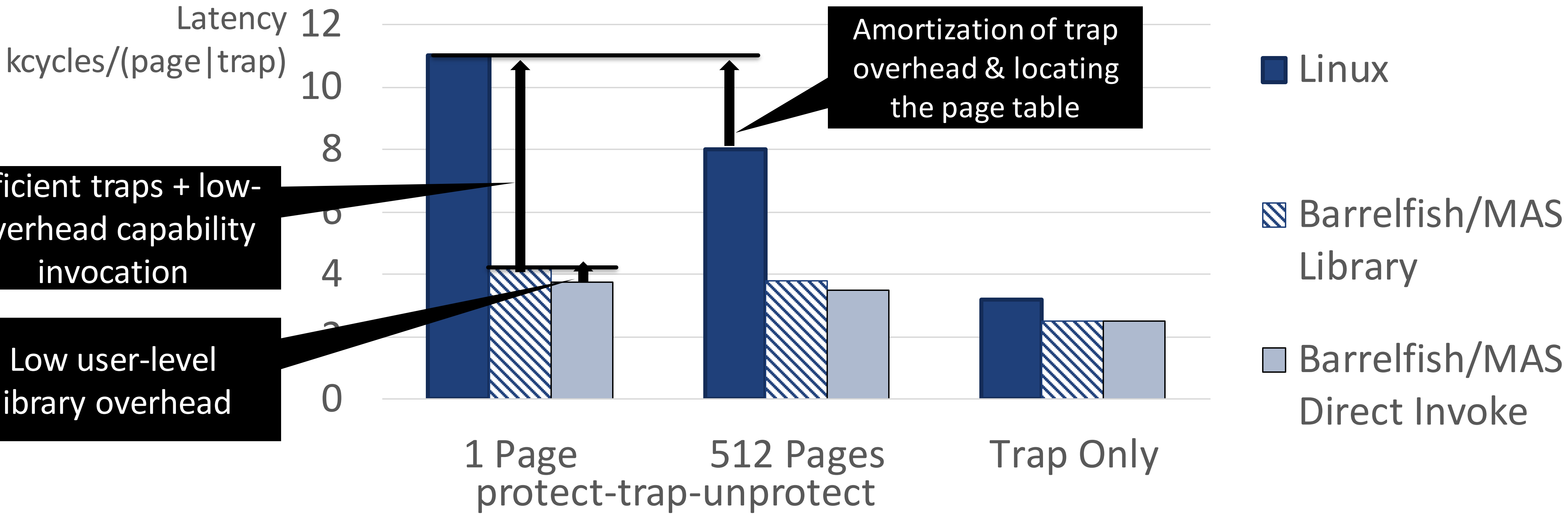
Multiple address spaces & least-privilege access contro.

What’s the cost of implementing this in an operating system?

1) What is the cost of memory management operations?

2) What is the overhead for dynamic address space configuration following the least-privilege principle?

Evaluation of virtual memory management operations
Appel/Li Benchmark



Amortization of trap overhead & locating the page table

Efficient traps + low-overhead capability invocation

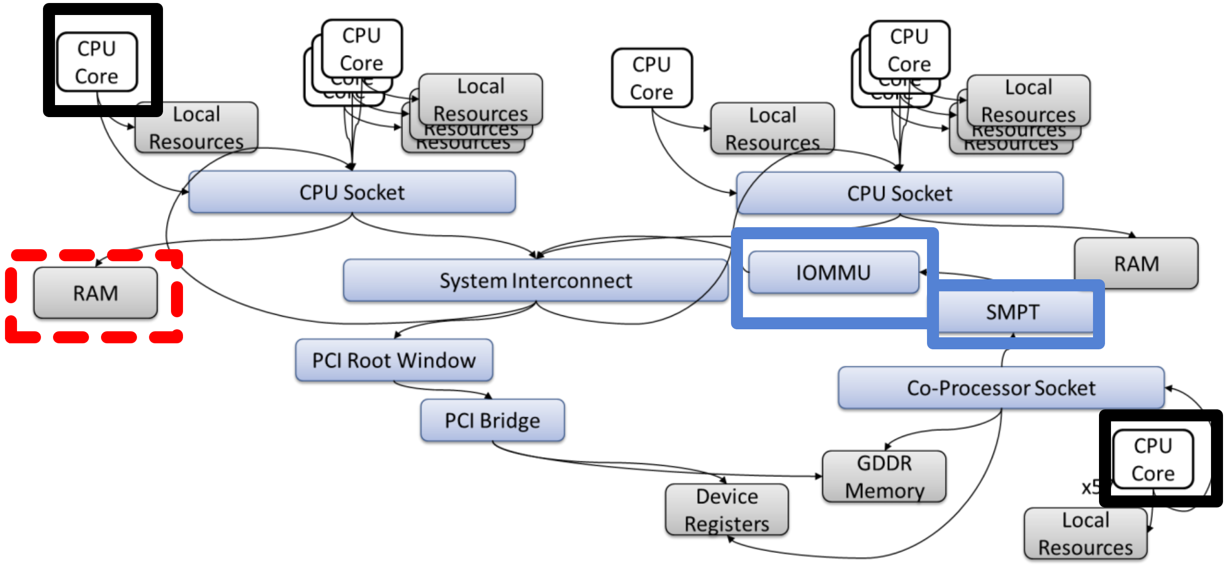
Low user-level library overhead

Match the performance of Linux despite implementation following least-privilege and explicit address spaces

The Cost of Dynamic Reconfiguration with Least-Privilege

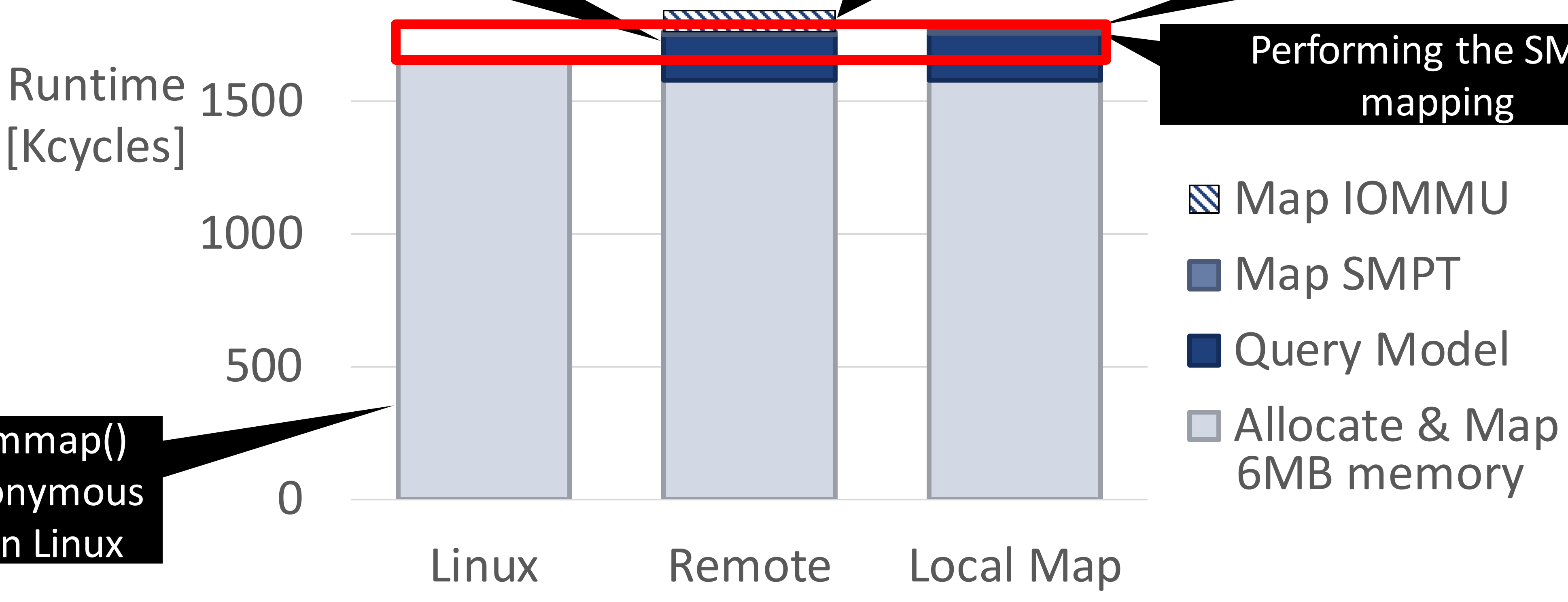
Task: Setup a shared buffer between the host CPU and the co-processor

1) Invoke model to obtain
   - memory resources to allocate
   - list of address spaces to configure
2) Allocation & mapping of memory
3) Configuration of address translation



Two-Socket server with Xeon Phi Co-Processor

The Cost of Dynamic Reconfiguration with Least-Privilege



Model evaluation for allocation & configuration

RPC to IOMMU service

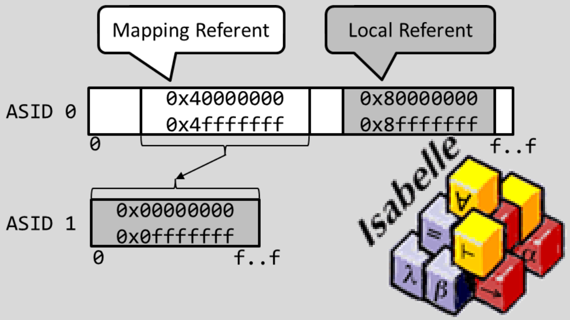
IOMMU mapping using direct capability invocations

Performing the SMPT mapping

Baseline: mmap() of 6MB anonymous memory on Linux

< 10% overhead for additional mappings and model queries

Realistic Hardware Abstractions and Least-Privilege Memory Management in Operating Systems



A new model to express memory addressing on modern machines

Generate OS code and hardware configuration

Express changes of the configuration and the required authority

Efficient Implementation in an Operating System

Future Directions

Performance Characteristics

Memory Access & Type Properties

Dynamic Caches

Vision

Apply the same approach to other areas of the operating system to obtain correct and reliable system software running on any platform.

Combining OS design & implementation with programming languages, code synthesis and Formal Methods.

Thanks to my collaborators

Timothy Roscoe
Simon Gerber
Gerd Zellweger
Izzat El Hajj

Lukas Humbel
Kornilios Kourtis
Roni Haecki
Alexander Merritt

Nora Hossle
Dejan Milojicic
Moritz Hoffmann
Contributors to the Barrelfish Operating System

David Cock
Stefan Kaestle
Sabela Ramos

Daniel Schwyn
Tim Harris
Jayneel Gandhi

List of Publications

- **R. Achermann**, N. Hossle, L. Humbel, D. Schwyn, D. Cock, T. Roscoe. *A Least-Privilege Memory Protection Model for Modern Hardware.* (ArXiv / in submission to ASPLOS'20)
- L. Azriel, L. Humbel, **R. Achermann**, A. Richardson, M. Hoffmann, A. Mendelson, T. Roscoe, RN. Watson, P. Faraboschi, D. Milojicic D. *Memory-side protection with a capability enforcement co-processor*. (TACO).
- **R. Achermann**, L. Humbel, D. Cock, T. Roscoe. *Physical addressing on real hardware in Isabelle/HOL*. (ITP'18).
- L. Humbel, **R. Achermann**, D. Cock, T. Roscoe. *Towards Correct-by-Construction Interrupt Routing on Real Hardware.* (PLOS'17).
- **R. Achermann**, C. Dalton, P. Faraboschi, M. Hoffmann, D. Milojicic, G. Ndu, A. Richardson, T. Roscoe, A. L. Shaw; R. N. M. Watson. *Separating Translation from Protection in Address Spaces with Dynamic Remapping*. (HOTOS'XVI).
- **R. Achermann**, L. Humbel, D. Cock and T. Roscoe. *Formalizing Memory Accesses and Interrupts*. (MARS 2017).
- S. Kaestle, **R. Achermann**, R. Haecki, M. Hoffmann, S. Ramos, and T. Roscoe. *Machine-Aware Atomic Broadcast Trees for Multicores*. (OSDI'16).
- I. El Hajj, A. Merritt, G. Zellweger, D. Milojicic, **R. Achermann**, W. Hwu, K. Schwan, T. Roscoe, P. Faraboschi. *SpaceJMP: Programming with Multiple Virtual Address Spaces*. (ASPLOS XXI).
- S. Kaestle, **R. Achermann,** T. Roscoe, T. Harris. *Shoal: Smart Allocation and Replication of Memory For Parallel Programs* (ATC'15)
- S. Gerber, G. Zellweger, **R. Achermann**, K. Kourtis, T. Roscoe, D. Milojicic. Not Your Parents' Physical Address Space. (HotOS XV).