The M³ Hardware/Software Platform

Nils Asmussen

07/02/2024, Huawei Summit 2024

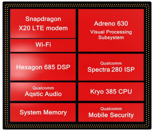
Motivation

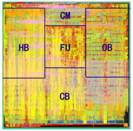
Software complexity

- Current operating systems are huge
- Used on various devices in daily life:

Hardware complexity

- Heterogeneity through specialization





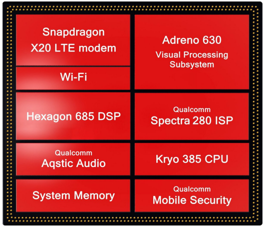
- Untrusted hardware components

Software Complexity

- Todays operating systems are huge and monolithic
- Microkernel-based systems as one solution: split OS up into isolated components
- Study: could have reduced severity of 96% of Linux' critical CVEs and eliminated 40% [1]

[1] Simon Biggs, Damon Lee, Gernot Heiser; The Jury Is In: Monolithic OS Design Is Flawed, APSys'18

Hardware Complexity: Heterogeneous Systems



Snapdragon X20 LTE modem

Wi-Fi

Adreno 630 Visual Processing Subsystem

Hexagon 685 DSP

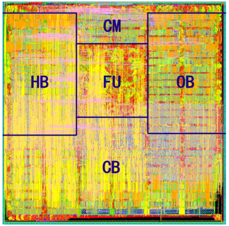
Qualcomm Spectra 280 ISP

Qualcomm Aqstic Audio

Kryo 385 CPU

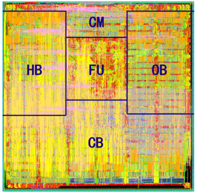
System Memory

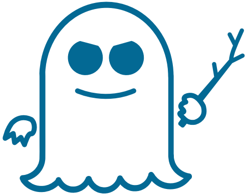
Qualcomm Mobile Security



- Demanded by performance and energy requirements
- Big challenge for OSes: single shared kernel on all cores does no longer work
- OSes need to be prepared for processing elements with different feature sets

Hardware Complexity: Untrusted Components



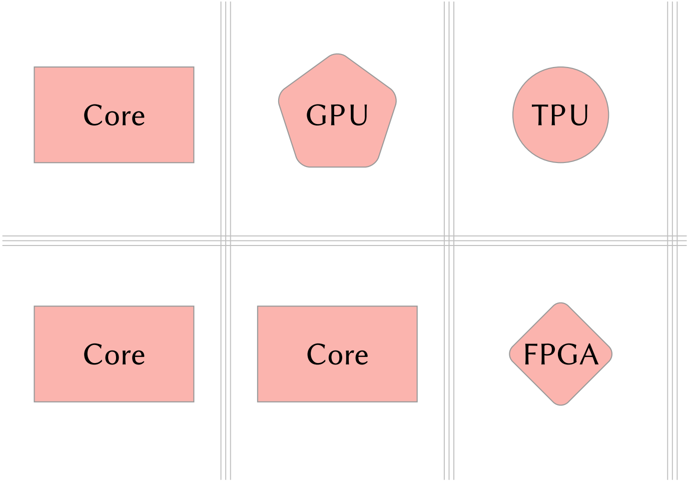


- Provided by third-party vendors
- Bug in such a component can compromise whole system (see Broadcom incident)
- Side channels in modern cores allow attackers to leak private data; some bypass all security measures of the core (address spaces, virtualization, . . .)
- Have been lurking in CPUs for many years, also due to complexity

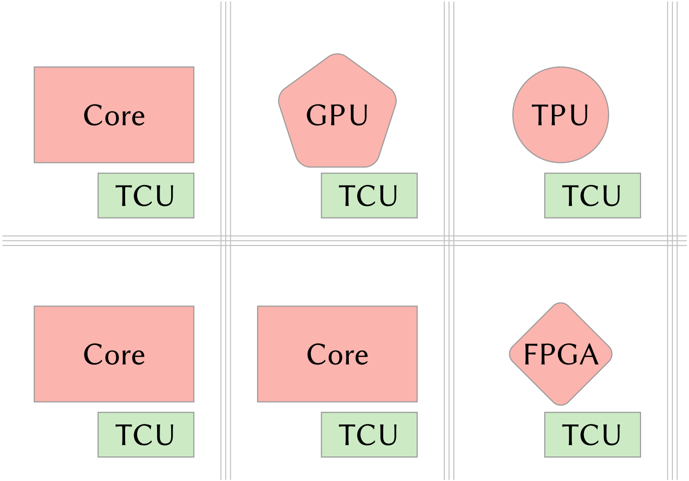
Our approach: split hardware and software into isolated components

Hardware/Operating System Co-Design

Hardware/Operating System Co-Design



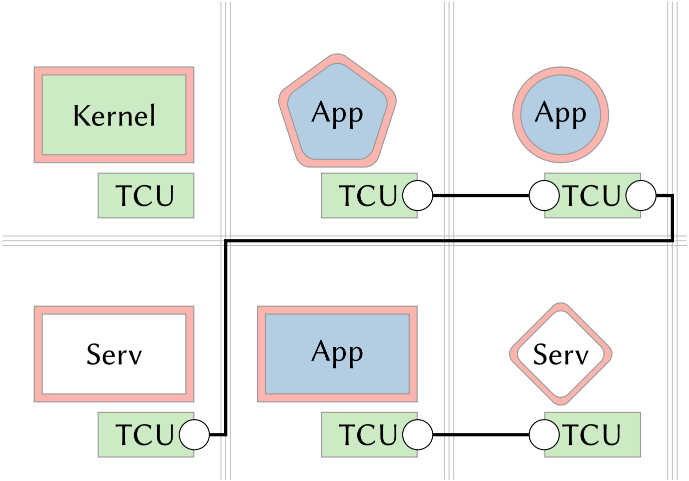
Hardware/Operating System Co-Design



Key ideas:

- TCU as new hardware component

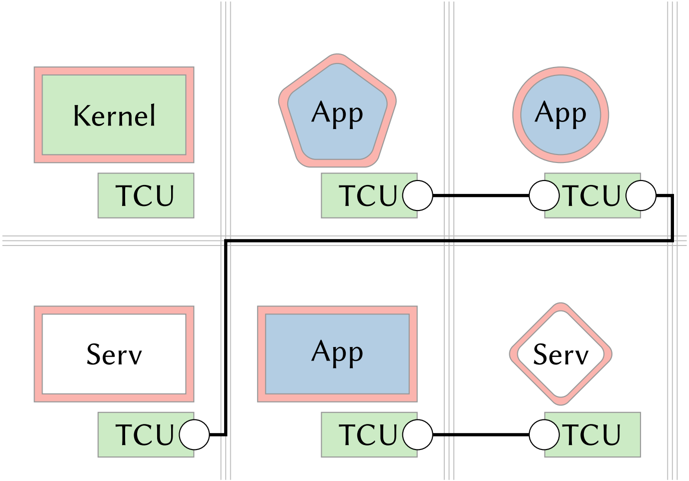
Hardware/Operating System Co-Design



Key ideas:

- TCU as new hardware component
- Direct communication between tiles

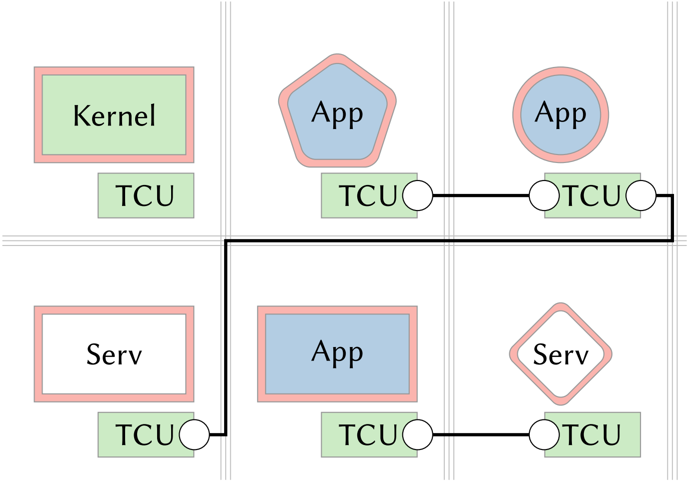
Hardware/Operating System Co-Design



Key ideas:

- TCU as new hardware component
- Direct communication between tiles
- Kernel on dedicated tile

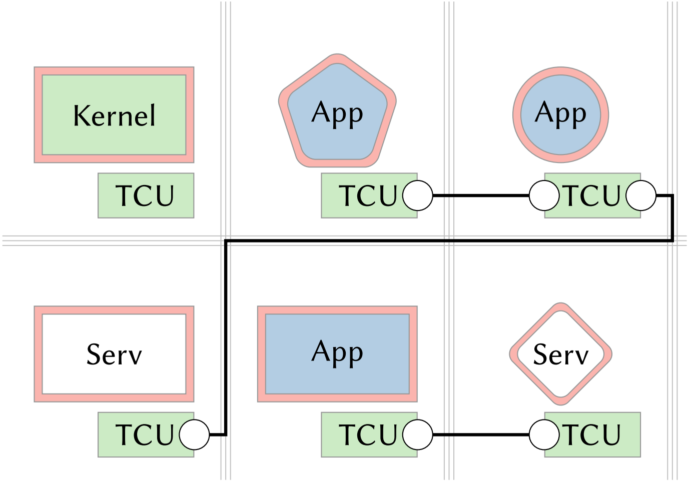
Hardware/Operating System Co-Design



Reduces complexity:

- Microkernel-based system

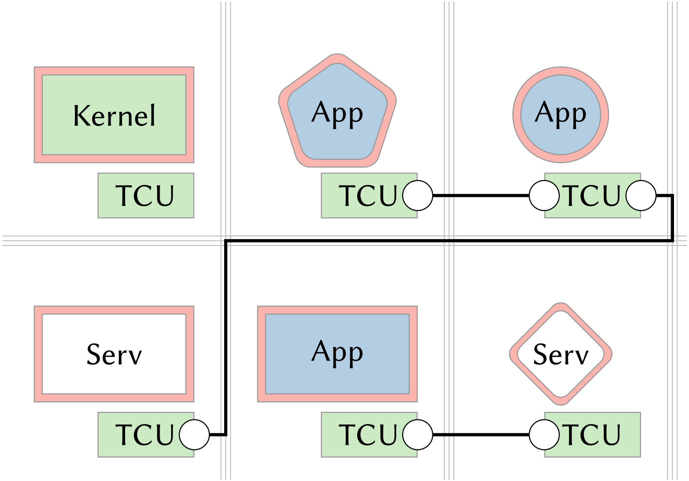
Hardware/Operating System Co-Design



Reduces complexity:

- Microkernel-based system
- TCU adds uniform interface

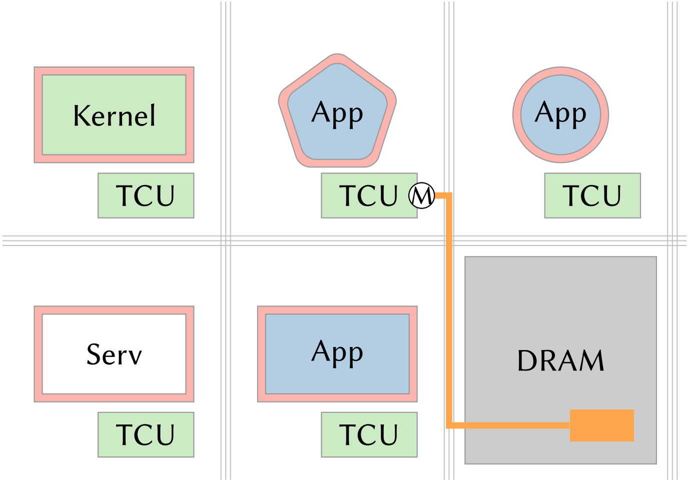
Hardware/Operating System Co-Design



Reduces complexity:

- Microkernel-based system
- TCU adds uniform interface
- TCU adds isolation

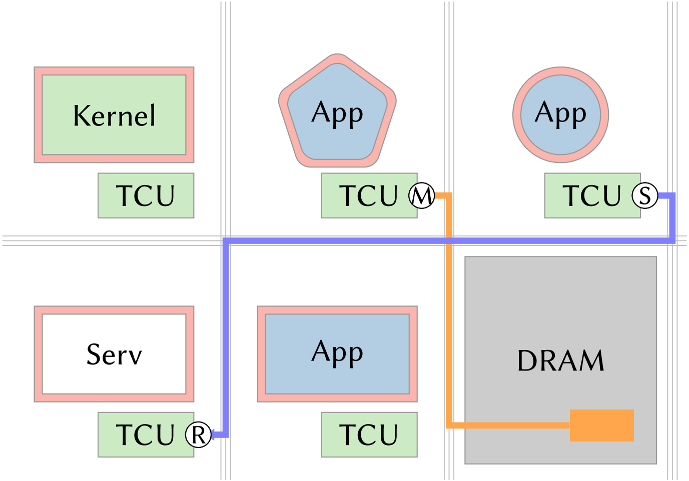
Communication



TCU provides *endpoints* to:

- Access memory (contiguous range, byte granular)

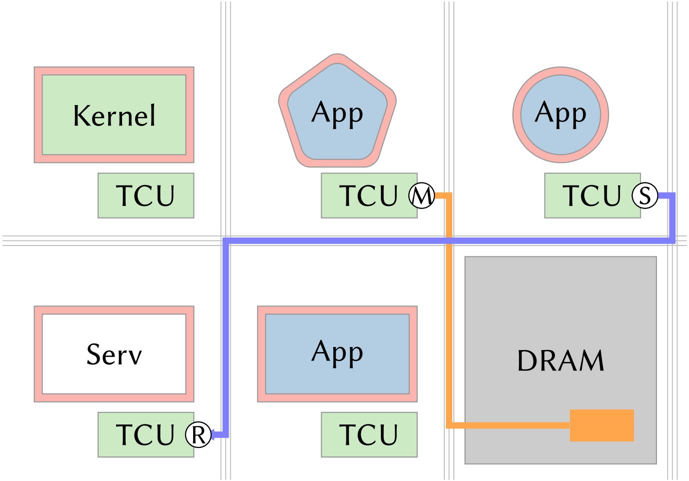
Communication



TCU provides *endpoints* to:

- Access memory (contiguous range, byte granular)
- Receive messages into a receive buffer
- Send messages to a receiving endpoint

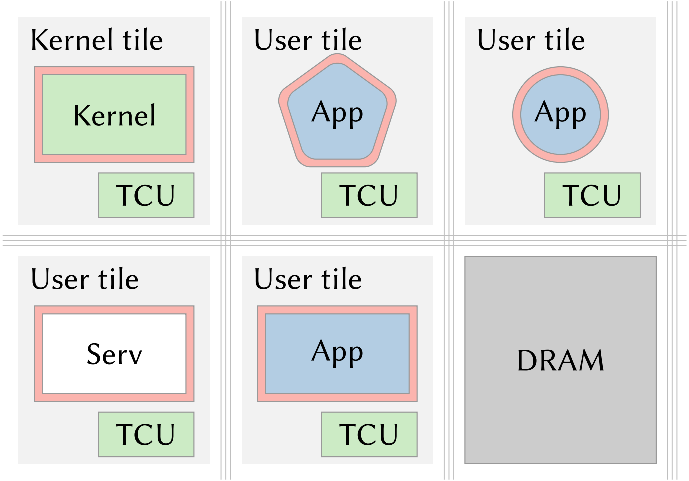
Communication



TCU provides *endpoints* to:

- Access memory (contiguous range, byte granular)
- Receive messages into a receive buffer
- Send messages to a receiving endpoint
- Replies for RPC

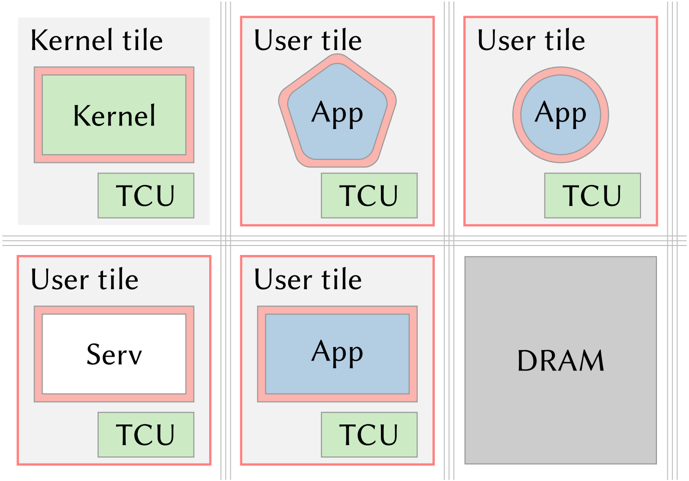
Isolation



TCU-based isolation:

- Additional protection layer

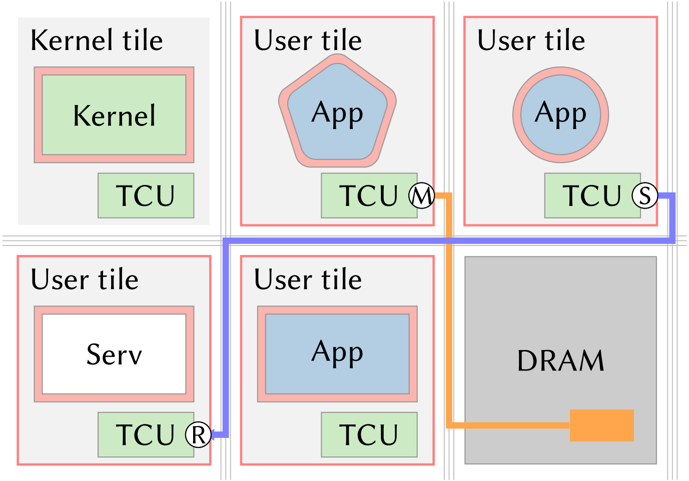
Isolation



TCU-based isolation:

- Additional protection layer

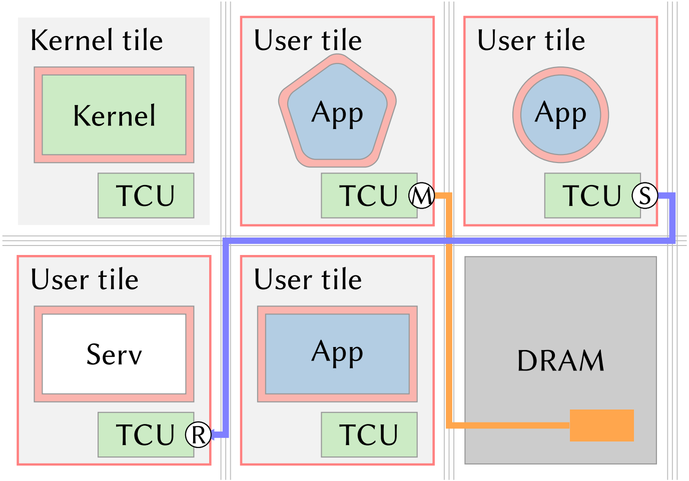
Isolation



TCU-based isolation:

- Additional protection layer
- Only kernel tile can establish communication channels

Isolation

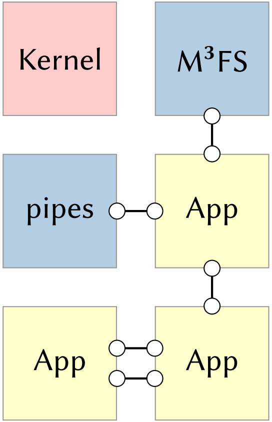


TCU-based isolation:

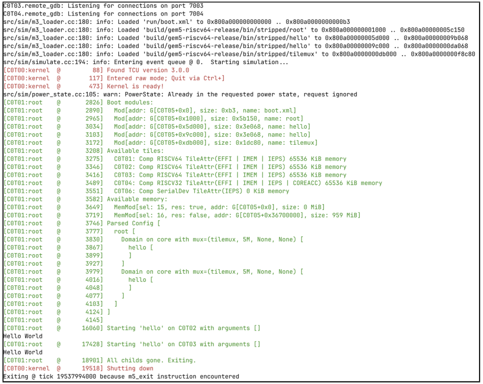
- Additional protection layer
- Only kernel tile can establish communication channels
- User tiles can only use established channels

OS Design

- M3: **M**icrokernel-based syste**m** for het. **m**anycores (or L4 ± 1)
- Implemented from scratch in Rust and C++
- Drivers, filesystems, etc. implemented on user tiles
- Kernel manages permissions, using capabilities
- TCU enforces permissions (communication, memory access)
- Kernel is independent of other tiles



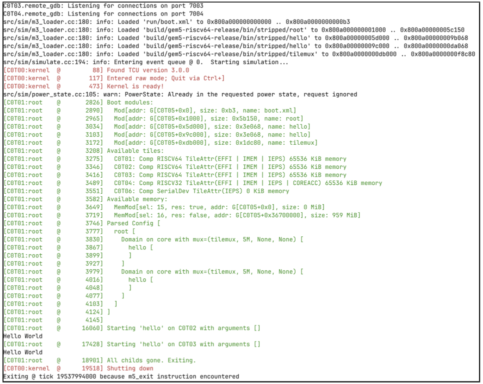
Prototype Platforms



```
C0T03.remote_gdb: Listening for connections on port 7003
C0T04.remote_gdb: Listening for connections on port 7004
src/sim/m3_loader.cc:180: info: Loaded 'run/boot.xml' to 0x800a000000000000 .. 0x800a0000000000b3
src/sim/m3_loader.cc:180: info: Loaded 'build/gem5-riscv64-release/bin/stripped/root' to 0x800a000000001000 .. 0x800a00000005c150
src/sim/m3_loader.cc:180: info: Loaded 'build/gem5-riscv64-release/bin/stripped/hello' to 0x800a00000005d000 .. 0x800a00000009b068
src/sim/m3_loader.cc:180: info: Loaded 'build/gem5-riscv64-release/bin/stripped/hello' to 0x800a00000009c000 .. 0x800a0000000da068
src/sim/m3_loader.cc:180: info: Loaded 'build/gem5-riscv64-release/bin/stripped/tilemux' to 0x800a0000000db000 .. 0x800a0000000f8c80
src/sim/simulate.cc:194: info: Entering event queue @ 0.  Starting simulation...
[C0T00:kernel  @         88] Found TCU version 3.0.0
[C0T00:kernel  @        117] Entered raw mode; Quit via Ctrl+]
[C0T00:kernel  @        473] Kernel is ready!
src/sim/power_state.cc:105: warn: PowerState: Already in the requested power state, request ignored
[C0T01:root    @       2826] Boot modules:
[C0T01:root    @       2890]   Mod[addr: G[C0T05+0x0], size: 0xb3, name: boot.xml]
[C0T01:root    @       2965]   Mod[addr: G[C0T05+0x1000], size: 0x5b150, name: root]
[C0T01:root    @       3034]   Mod[addr: G[C0T05+0x5d000], size: 0x3e068, name: hello]
[C0T01:root    @       3103]   Mod[addr: G[C0T05+0x9c000], size: 0x3e068, name: hello]
[C0T01:root    @       3172]   Mod[addr: G[C0T05+0xdb000], size: 0x1dc80, name: tilemux]
[C0T01:root    @       3208] Available tiles:
[C0T01:root    @       3275]   C0T01: Comp RISCV64 TileAttr(EFFI | IMEM | IEPS) 65536 KiB memory
[C0T01:root    @       3346]   C0T02: Comp RISCV64 TileAttr(EFFI | IMEM | IEPS) 65536 KiB memory
[C0T01:root    @       3416]   C0T03: Comp RISCV64 TileAttr(EFFI | IMEM | IEPS) 65536 KiB memory
[C0T01:root    @       3489]   C0T04: Comp RISCV32 TileAttr(EFFI | IMEM | IEPS | COREACC) 65536 KiB memory
[C0T01:root    @       3551]   C0T06: Comp SerialDev TileAttr(IEPS) 0 KiB memory
[C0T01:root    @       3582] Available memory:
[C0T01:root    @       3649]   MemMod[sel: 15, res: true, addr: G[C0T05+0x0], size: 0 MiB]
[C0T01:root    @       3719]   MemMod[sel: 16, res: false, addr: G[C0T05+0x36700000], size: 959 MiB]
[C0T01:root    @       3746] Parsed Config [
[C0T01:root    @       3777]   root [
[C0T01:root    @       3830]     Domain on core with mux=(tilemux, 5M, None, None) [
[C0T01:root    @       3867]       hello [
[C0T01:root    @       3899]       ]
[C0T01:root    @       3927]     ]
[C0T01:root    @       3979]     Domain on core with mux=(tilemux, 5M, None, None) [
[C0T01:root    @       4016]       hello [
[C0T01:root    @       4048]       ]
[C0T01:root    @       4077]     ]
[C0T01:root    @       4103]   ]
[C0T01:root    @       4124] ]
[C0T01:root    @       4145]
[C0T01:root    @      16060] Starting 'hello' on C0T02 with arguments []
Hello World
[C0T01:root    @      17428] Starting 'hello' on C0T03 with arguments []
Hello World
[C0T01:root    @      18901] All childs gone. Exiting.
[C0T00:kernel  @      19518] Shutting down
Exiting @ tick 19537994000 because m5_exit instruction encountered
```

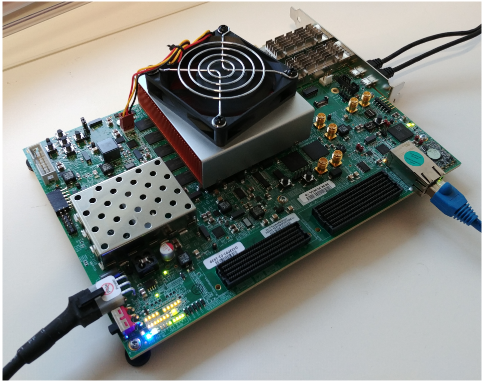
gem5 simulator

Prototype Platforms



```
C0T03.remote_gdb: Listening for connections on port 7003
C0T04.remote_gdb: Listening for connections on port 7004
src/sim/m3_loader.cc:180: info: Loaded 'run/boot.xml' to 0x800a000000000000 .. 0x800a0000000000b3
src/sim/m3_loader.cc:180: info: Loaded 'build/gem5-riscv64-release/bin/stripped/root' to 0x800a000000001000 .. 0x800a00000005c150
src/sim/m3_loader.cc:180: info: Loaded 'build/gem5-riscv64-release/bin/stripped/hello' to 0x800a00000005d000 .. 0x800a00000009b068
src/sim/m3_loader.cc:180: info: Loaded 'build/gem5-riscv64-release/bin/stripped/hello' to 0x800a00000009c000 .. 0x800a0000000da068
src/sim/m3_loader.cc:180: info: Loaded 'build/gem5-riscv64-release/bin/stripped/tilemux' to 0x800a0000000db000 .. 0x800a0000000f8c80
src/sim/simulate.cc:194: info: Entering event queue @ 0.  Starting simulation...
[C0T00:kernel  @         88] Found TCU version 3.0.0
[C0T00:kernel  @        117] Entered raw mode; Quit via Ctrl+]
[C0T00:kernel  @        473] Kernel is ready!
src/sim/power_state.cc:105: warn: PowerState: Already in the requested power state, request ignored
[C0T01:root    @       2826] Boot modules:
[C0T01:root    @       2890]   Mod[addr: G[C0T05+0x0], size: 0xb3, name: boot.xml]
[C0T01:root    @       2965]   Mod[addr: G[C0T05+0x1000], size: 0x5b150, name: root]
[C0T01:root    @       3034]   Mod[addr: G[C0T05+0x5d000], size: 0x3e068, name: hello]
[C0T01:root    @       3103]   Mod[addr: G[C0T05+0x9c000], size: 0x3e068, name: hello]
[C0T01:root    @       3172]   Mod[addr: G[C0T05+0xdb000], size: 0x1dc80, name: tilemux]
[C0T01:root    @       3208] Available tiles:
[C0T01:root    @       3275]   C0T01: Comp RISCV64 TileAttr(EFFI | IMEM | IEPS) 65536 KiB memory
[C0T01:root    @       3346]   C0T02: Comp RISCV64 TileAttr(EFFI | IMEM | IEPS) 65536 KiB memory
[C0T01:root    @       3416]   C0T03: Comp RISCV64 TileAttr(EFFI | IMEM | IEPS) 65536 KiB memory
[C0T01:root    @       3489]   C0T04: Comp RISCV32 TileAttr(EFFI | IMEM | IEPS | COREACC) 65536 KiB memory
[C0T01:root    @       3551]   C0T06: Comp SerialDev TileAttr(IEPS) 0 KiB memory
[C0T01:root    @       3582] Available memory:
[C0T01:root    @       3649]   MemMod[sel: 15, res: true, addr: G[C0T05+0x0], size: 0 MiB]
[C0T01:root    @       3719]   MemMod[sel: 16, res: false, addr: G[C0T05+0x36700000], size: 959 MiB]
[C0T01:root    @       3746] Parsed Config [
[C0T01:root    @       3777]   root [
[C0T01:root    @       3830]     Domain on core with mux=(tilemux, 5M, None, None) [
[C0T01:root    @       3867]       hello [
[C0T01:root    @       3899]       ]
[C0T01:root    @       3927]     ]
[C0T01:root    @       3979]     Domain on core with mux=(tilemux, 5M, None, None) [
[C0T01:root    @       4016]       hello [
[C0T01:root    @       4048]       ]
[C0T01:root    @       4077]     ]
[C0T01:root    @       4103]   ]
[C0T01:root    @       4124] ]
[C0T01:root    @       4145]
[C0T01:root    @      16060] Starting 'hello' on C0T02 with arguments []
Hello World
[C0T01:root    @      17428] Starting 'hello' on C0T03 with arguments []
Hello World
[C0T01:root    @      18901] All childs gone. Exiting.
[C0T00:kernel  @      19518] Shutting down
Exiting @ tick 19537994000 because m5_exit instruction encountered
```

gem5 simulator



FPGA

Research based on M^3

- **M^3: A Hardware/Operating-System Co-Design to Tame Heterogeneous Manycores**
  Nils Asmussen, Marcus Völp, Benedikt Nöthen, Hermann Härtig, Gerhard Fettweis, **ASPLOS'16**
- **M^{3}x: Autonomous Accelerators via Context-Enabled Fast-Path Communication**
  Nils Asmussen, Michael Roitzsch, Hermann Härtig, **UATC'19**
- **SemperOS: A Distributed Capability System**
  Matthias Hille, Nils Asmussen, Pramod Bhatotia, Hermann Härtig, **UATC'19**
- **Efficient and Scalable Core Multiplexing with M^{3}v**
  Nils Asmussen, Sebastian Haas, Carsten Weinhold, Till Miemietz, Michael Roitzsch, **ASPLOS'22**
- **Towards Disaggregation-Native Data Streaming between Devices**
  Nils Asmussen, Michael Roitzsch, **HCDS'24**
- **Core-Local Reasoning and Predictable Cross-Core Communication with M^3**
  Nils Asmussen, Sebastian Haas, Adam Lackorzyński, Michael Roitzsch, **RTAS'24**

Research based on M^3

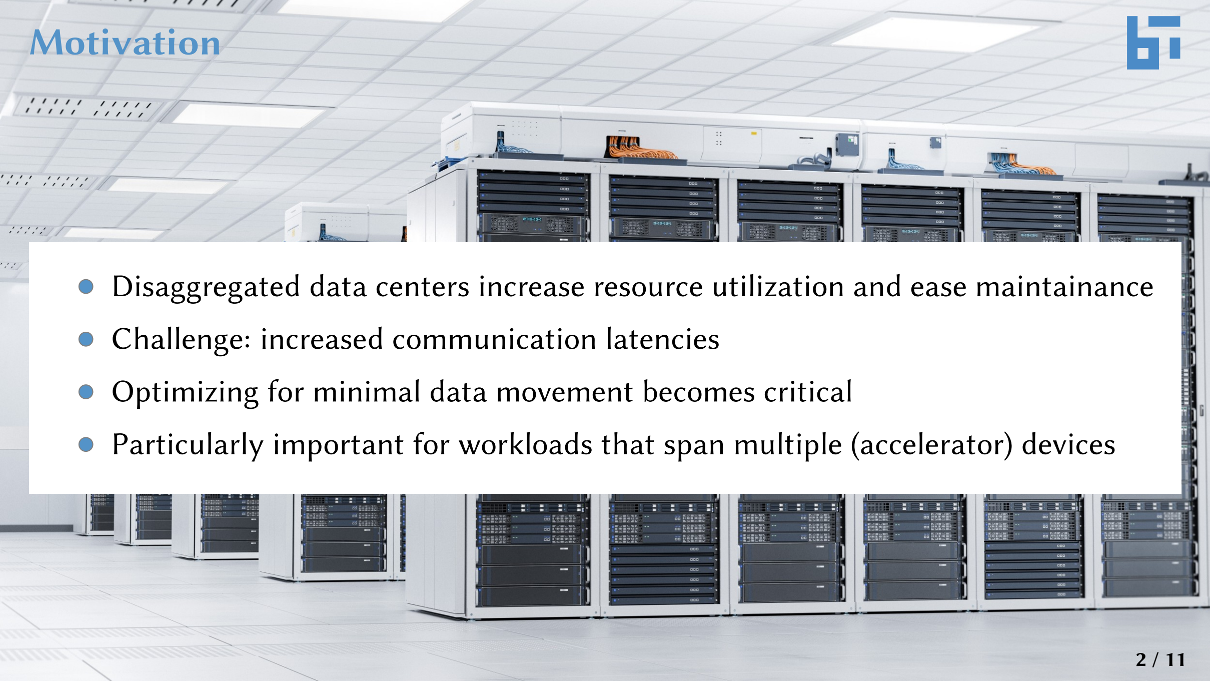
- **M^3: A Hardware/Operating-System Co-Design to Tame Heterogeneous Manycores**
  Nils Asmussen, Marcus Völp, Benedikt Nöthen, Hermann Härtig, Gerhard Fettweis, **ASPLOS'16**
- **M^{3}x: Autonomous Accelerators via Context-Enabled Fast-Path Communication**
  Nils Asmussen, Michael Roitzsch, Hermann Härtig, **UATC'19**
- **SemperOS: A Distributed Capability System**
  Matthias Hille, Nils Asmussen, Pramod Bhatotia, Hermann Härtig, **UATC'19**
- **Efficient and Scalable Core Multiplexing with M^{3}v**
  Nils Asmussen, Sebastian Haas, Carsten Weinhold, Till Miemietz, Michael Roitzsch, **ASPLOS'22**
- **Towards Disaggregation-Native Data Streaming between Devices**
  Nils Asmussen, Michael Roitzsch, **HCDS'24**
- **Core-Local Reasoning and Predictable Cross-Core Communication with M^3**
  Nils Asmussen, Sebastian Haas, Adam Lackorzyński, Michael Roitzsch, **RTAS'24**

Towards Disaggregation-Native Data Streaming between Devices

Nils Asmussen, Michael Roitzsch

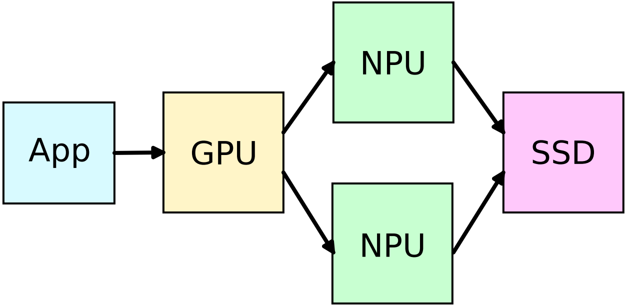
HCDS'24 – April 28, 2024

Motivation

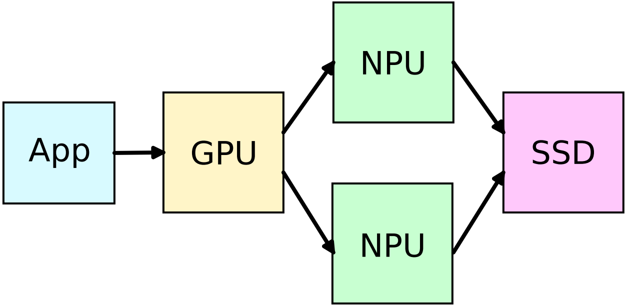


- Disaggregated data centers increase resource utilization and ease maintainance
- Challenge: increased communication latencies
- Optimizing for minimal data movement becomes critical
- Particularly important for workloads that span multiple (accelerator) devices

Example Workload on Future CXL-based Systems

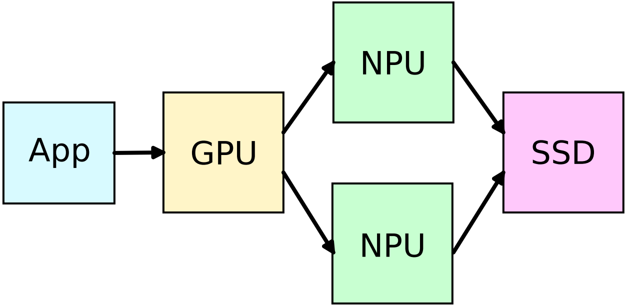


Example Workload on Future CXL-based Systems



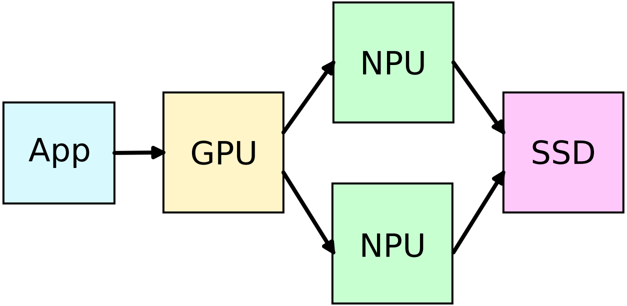
- CXL allows device-device interaction

Example Workload on Future CXL-based Systems



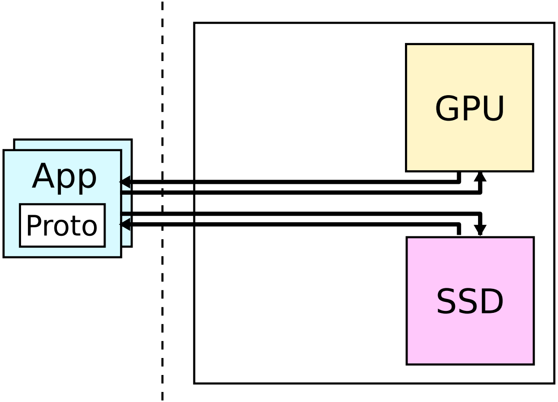
- CXL allows device-device interaction
- Driver for each device on each device infeasible

Example Workload on Future CXL-based Systems



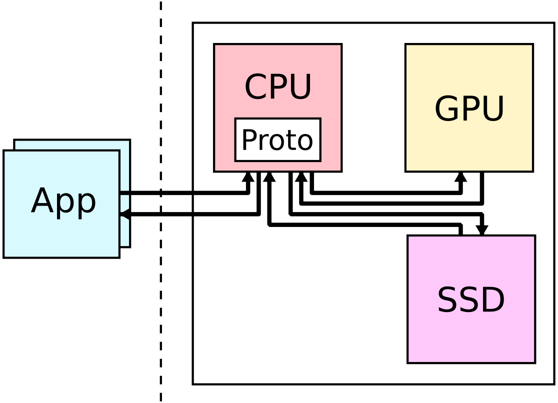
- CXL allows device-device interaction
- Driver for each device on each device infeasible
- Open questions: Protocol design and **placement**

Protocol Placement: Application-Side



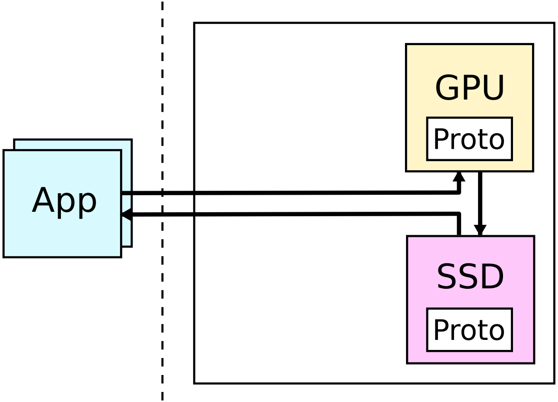
- No common protocol required
- Extra (cross-machine) communication hops

Protocol Placement: Central Resource-Side



- No common protocol
- Extra communication hops

Protocol Placement: Distributed Resource-Side



- Accelerators or co-processors execute common protocol
- No extra communication hops

Disaggregation-Native Devices

Requirements:

- **Direct communication:** avoid CPUs as intermediaries

Disaggregation-Native Devices

Requirements:

- **Direct communication:** avoid CPUs as intermediaries
- **Access restrictions:** enforce application-specific permissions at accelerators

Disaggregation-Native Devices

Requirements:

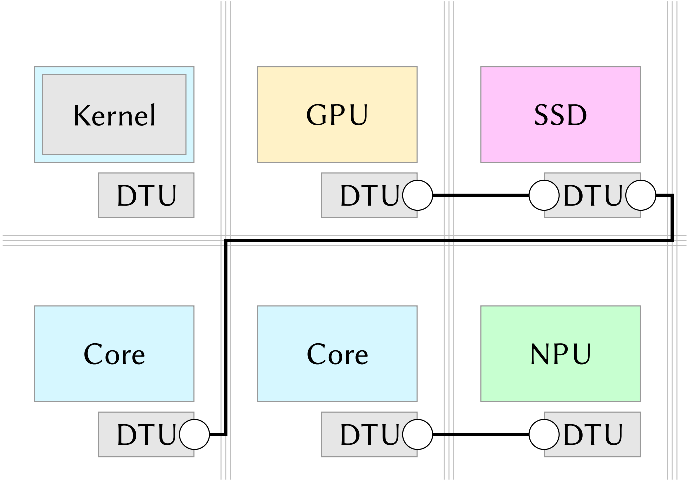
- **Direct communication:** avoid CPUs as intermediaries
- **Access restrictions:** enforce application-specific permissions at accelerators
- **Common protocol:** device-specific protocols replaced by common protocol

Disaggregation-Native Devices

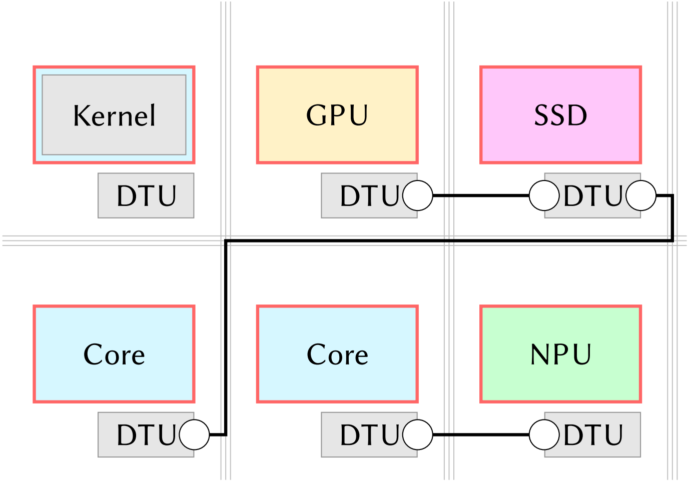
Requirements:

- **Direct communication:** avoid CPUs as intermediaries
- **Access restrictions:** enforce application-specific permissions at accelerators
- **Common protocol:** device-specific protocols replaced by common protocol
- **Protocol deployment:** implemented on accelerator or co-processor

M3 as Foundation for Disaggregation-Native Devices

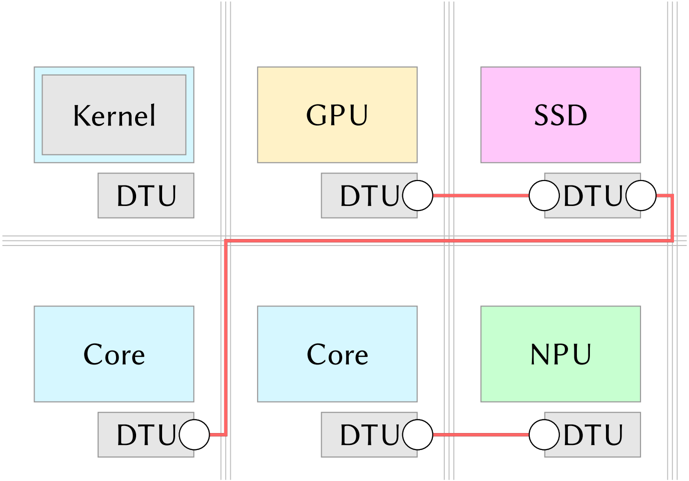


M³ as Foundation for Disaggregation-Native Devices



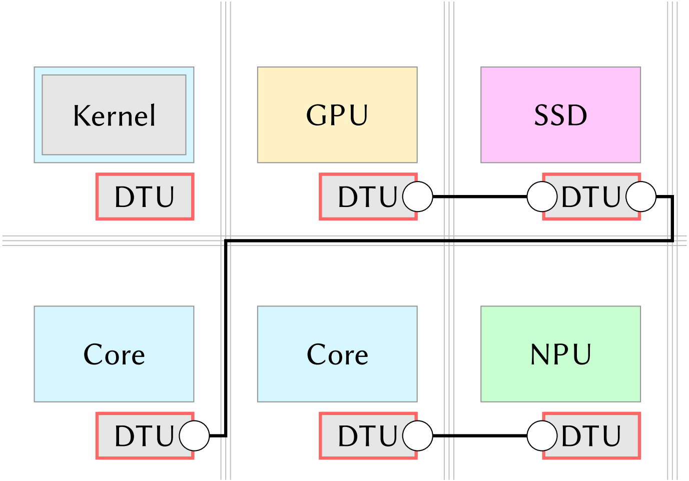
- Supports heterogeneous devices by design

M3 as Foundation for Disaggregation-Native Devices



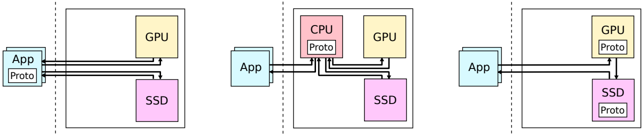
- Supports heterogeneous devices by design
- Direct communication between devices

M^3 as Foundation for Disaggregation-Native Devices



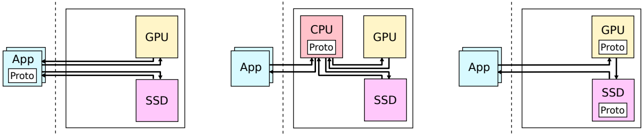
- Supports heterogeneous devices by design
- Direct communication between devices
- Access restrictions via DTU

Potential of Disaggregation-Native Devices



- M^3 on gem5 simulator

Potential of Disaggregation-Native Devices



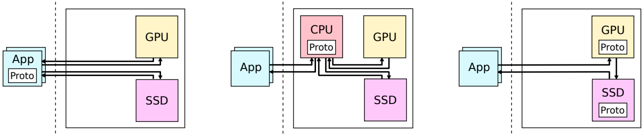
- M^3 on gem5 simulator
- Conservative settings:

  1μs inter machine latency

  0.5μs intra machine latency

  4 GHz CPUs, 1 GHz Co-Proc.

Potential of Disaggregation-Native Devices



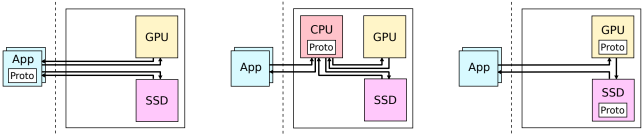
- M^3 on gem5 simulator
- Conservative settings:

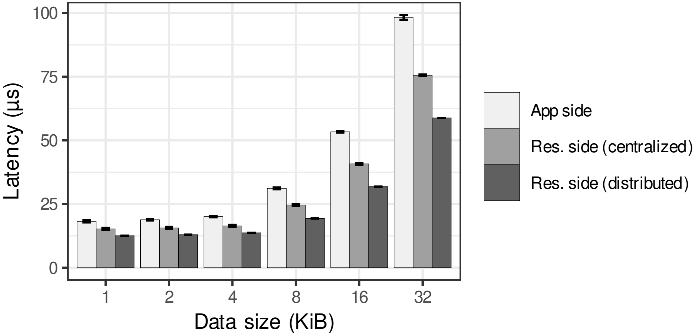
  1μs inter machine latency

  0.5μs intra machine latency

  4 GHz CPUs, 1 GHz Co-Proc.
- No compute, only protocol execution on CPUs/Co-Proc.

Potential of Disaggregation-Native Devices





- M^3 on gem5 simulator
- Conservative settings:
  $1\mu s$ inter machine latency
  $0.5\mu s$ intra machine latency
  4 GHz CPUs, 1 GHz Co-Proc.
- No compute, only protocol execution on CPUs/Co-Proc.

Challenges and Open Questions

- M^3 is currently designed for SoCs
- External management of CPU-less servers with M^3 kernel?
- How to design the protocol to be simple, efficient, and flexible?
- How do the DTU primitives map to CXL fabrics?

Summary

- Disaggregation-native: direct communication between devices
- Offers latency benefits even with conservative measurements
- Common data-streaming protocol could also ease programming
- We believe M^3 provides a nice foundation

Core-Local Reasoning and Predictable Cross-Core Communication with M^3

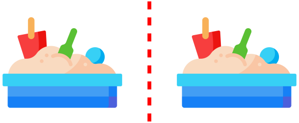
Nils Asmussen[1], Sebastian Haas[1], Adam Lackorzyński[2], Michael Roitzsch[1]

[1]Barkhausen Institut, [2]TU Dresden

RTAS’24, May 15th 2024

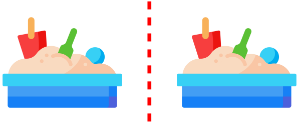
Requirements for Cyber-Physical Systems

1. Strong security guarantees

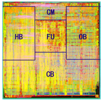


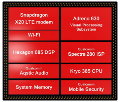
Requirements for Cyber-Physical Systems

1. Strong security guarantees



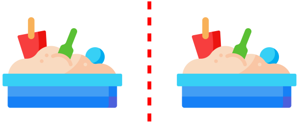
2. Hardware-level heterogeneity



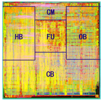


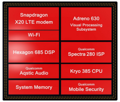
Requirements for Cyber-Physical Systems

1. Strong security guarantees



2. Hardware-level heterogeneity

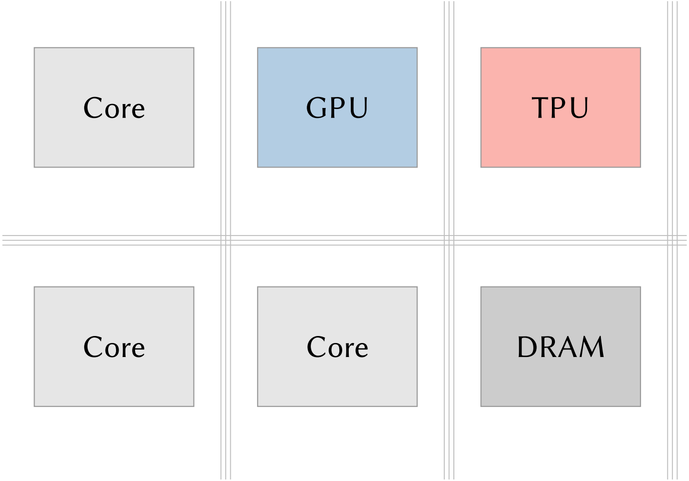




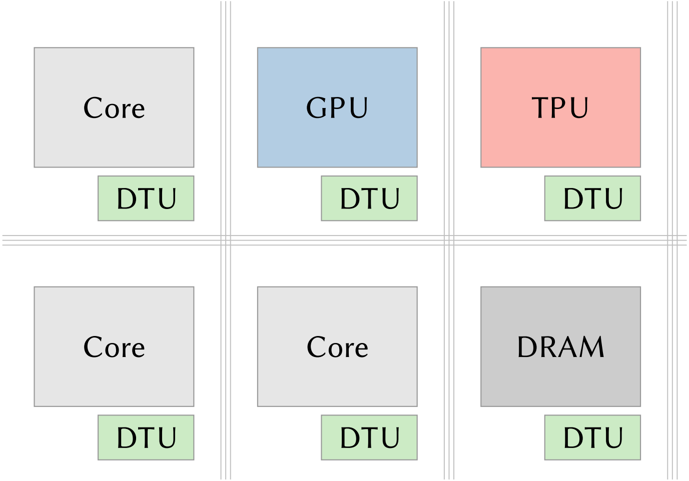
3. Real-time guarantees

M^3: Hardware/Operating System Co-Design

M^3: Hardware/Operating System Co-Design



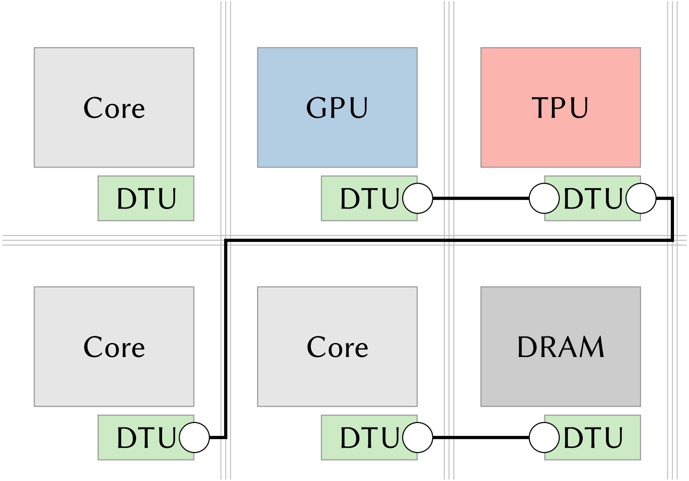
M^3: Hardware/Operating System Co-Design



Key ideas:

- DTU as new hardware component

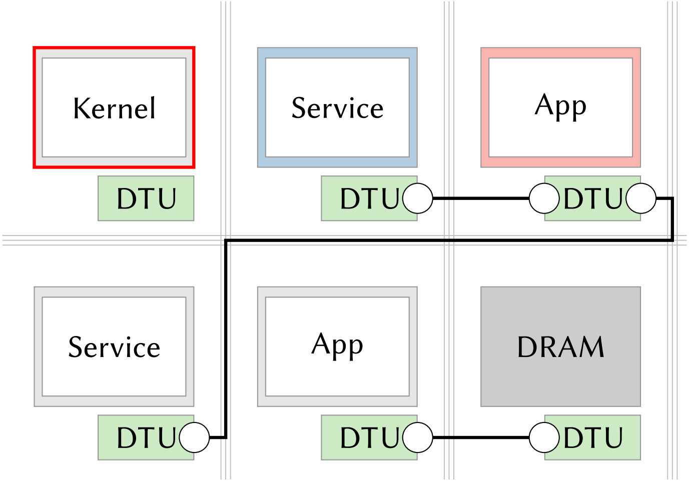
M3: Hardware/Operating System Co-Design



Key ideas:

- DTU as new hardware component
- Direct communication between tiles

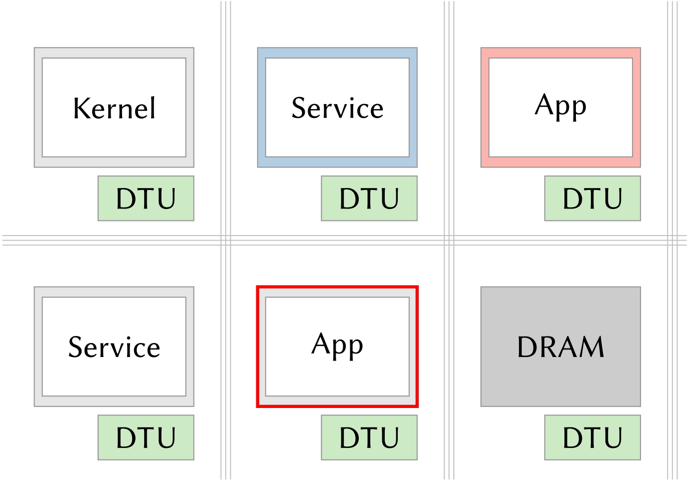
M3: Hardware/Operating System Co-Design



Key ideas:

- DTU as new hardware component
- Direct communication between tiles
- Kernel on dedicated tile

M3: Advantages for Security [1]

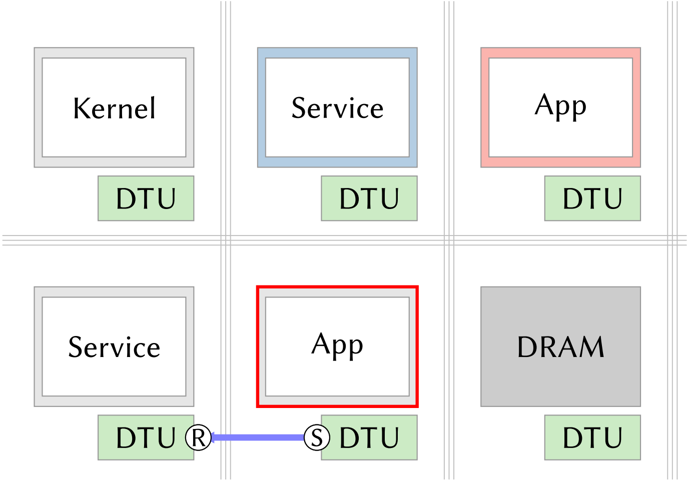


Allows integration of untrusted cores/accelerators:

- OS kernel needs to configure *endpoint* first

[1] Asmussen et al.; M3: A Hardware/Operating-System Co-Design to Tame Heterogeneous Manycores, ASPLOS 2016

M3: Advantages for Security [1]

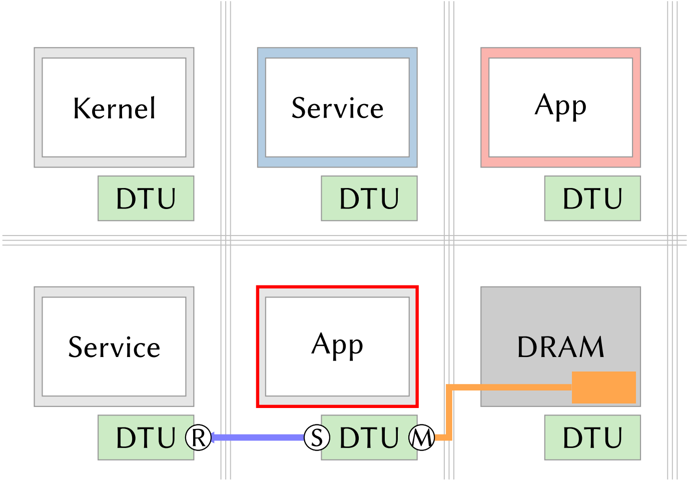


Allows integration of untrusted cores/accelerators:

- OS kernel needs to configure *endpoint* first
- Send/receive endpoint for message passing

[1] Asmussen et al.; M3: A Hardware/Operating-System Co-Design to Tame Heterogeneous Manycores, ASPLOS 2016

M³: Advantages for Security [1]

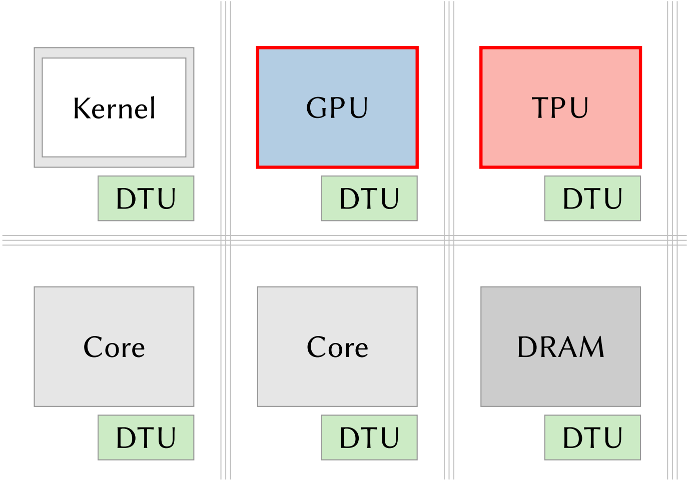


Allows integration of untrusted cores/accelerators:

- OS kernel needs to configure *endpoint* first
- Send/receive endpoint for message passing
- Memory endpoint to access tile-external memory

[1] Asmussen et al.; M³: A Hardware/Operating-System Co-Design to Tame Heterogeneous Manycores, ASPLOS 2016

M^3: Advantages for Heterogeneity [2]

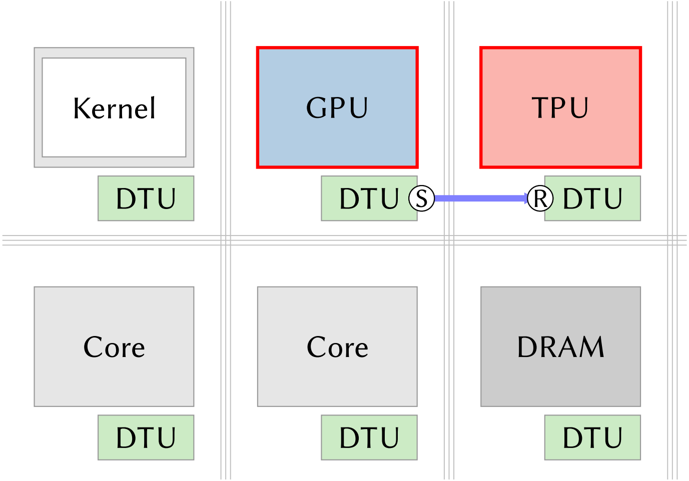


DTU provides uniform interface:

- Simplifies platform management for OS

[2] Asmussen et al.; M^{3}x: Autonomous Accelerators via Context-Enabled Fast-Path Communication, UATC 2019

M³: Advantages for Heterogeneity [2]



DTU provides uniform interface:

- Simplifies platform management for OS
- Simplifies collaboration of different tiles

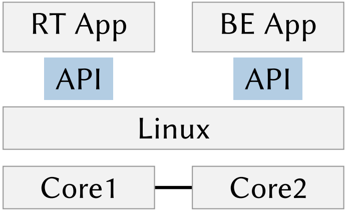
[2] Asmussen et al.; M³x: Autonomous Accelerators via Context-Enabled Fast-Path Communication, UATC 2019

Why is M^3 Interesting for Real-Time?

1. Tile specialization and local reasoning
2. Low-latency and low-jitter cross-tile communication
3. Avoids issues with client priorities

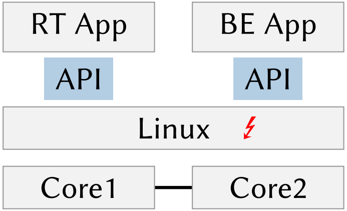
Real-Time with M^3: Tile Specialization and Local Reasoning

Shared Linux



Real-Time with M³: Tile Specialization and Local Reasoning

Shared Linux



RT App

BE App

API

API

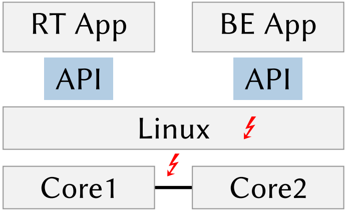
Linux ↯

Core1

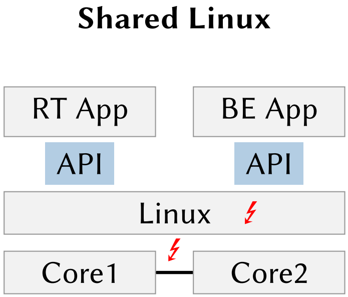
Core2

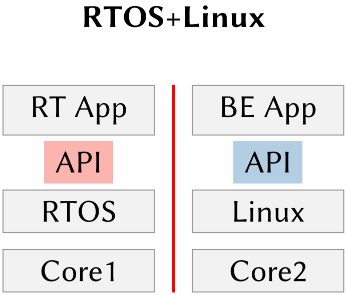
Real-Time with M3: Tile Specialization and Local Reasoning

Shared Linux

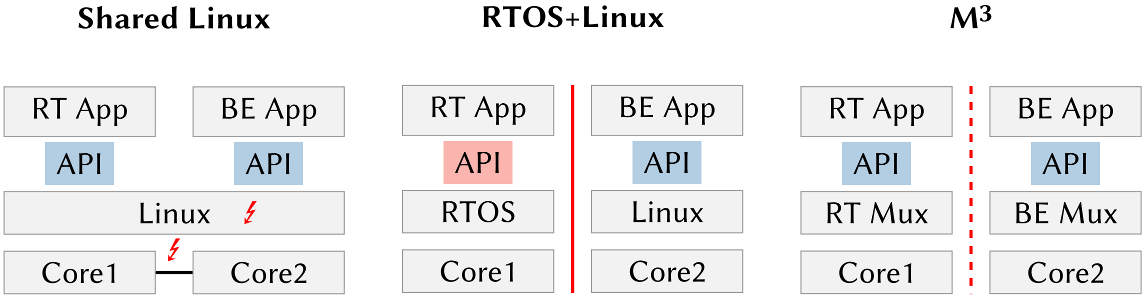


Real-Time with M^3: Tile Specialization and Local Reasoning

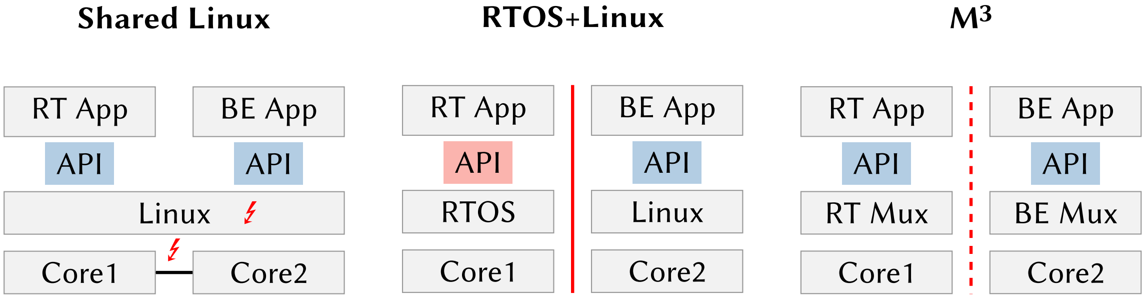




Real-Time with M^3: Tile Specialization and Local Reasoning



Real-Time with M³: Tile Specialization and Local Reasoning



M³ enables *local reasoning* without losing the shared-system experience

Real-Time with M³: Client Priorities

Interference from low-prio to high-prio clients [3]:

- Sorted IPC queue
- Scheduling budgets

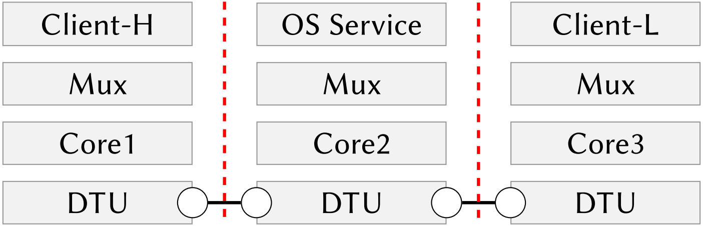
[3] Mergendahl et al.: The thundering herd: Amplifying kernel interference to attack response times, RTAS 2022

Real-Time with M³: Client Priorities

Interference from low-prio to high-prio clients [3]:

- Sorted IPC queue
- Scheduling budgets

M³ side steps these problems:



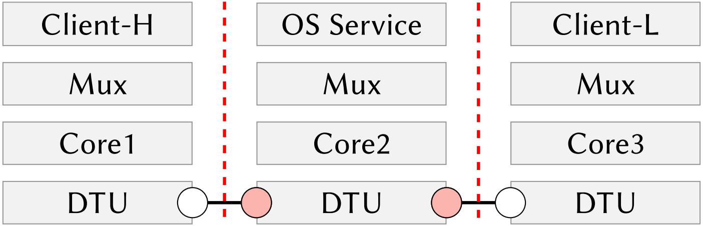
[3] Mergendahl et al.: The thundering herd: Amplifying kernel interference to attack response times, RTAS 2022

Real-Time with M3: Client Priorities

Interference from low-prio to high-prio clients [3]:

- Sorted IPC queue
- Scheduling budgets

M3 side steps these problems:



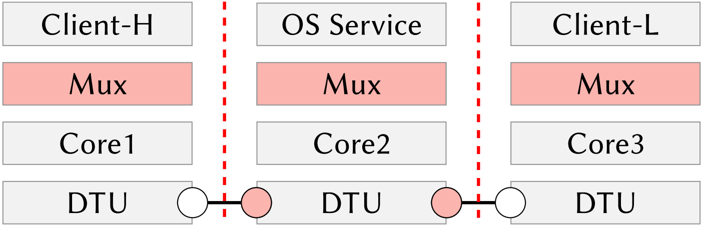
[3] Mergendahl et al.: The thundering herd: Amplifying kernel interference to attack response times, RTAS 2022

Real-Time with M³: Client Priorities

Interference from low-prio to high-prio clients [3]:

- Sorted IPC queue
- Scheduling budgets

M³ side steps these problems:



Client-H | OS Service | Client-L

Mux | Mux | Mux

Core1 | Core2 | Core3

DTU | DTU | DTU

[3] Mergendahl et al.: The thundering herd: Amplifying kernel interference to attack response times, RTAS 2022

Enhancements to M^3's Real-Time Guarantees

1. NoC-traffic regulation
2. End-to-end enforcement of resource limits
3. Fast and energy-efficient communication

Local Reasoning?

- Cross-tile interference study
- Single foreground workload disturbed max. possible background workloads
- Run on gem5-based simulation platform (FPGA measurements in paper)

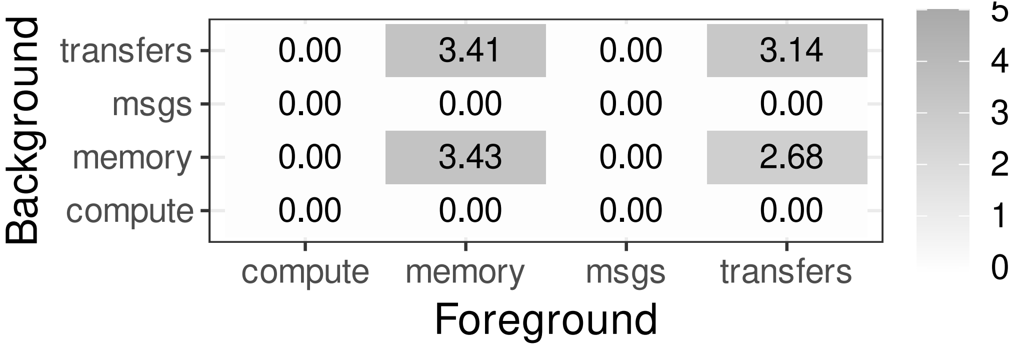
Local Reasoning?

- Cross-tile interference study
- Single foreground workload disturbed max. possible background workloads
- Run on gem5-based simulation platform (FPGA measurements in paper)

Workloads

- **Transfers**: DMA requests to memory
- **Memory**: memory accesses beyond cache capacity
- **Msgs**: message passing with max. message size
- **Compute**: number crunching without cache misses

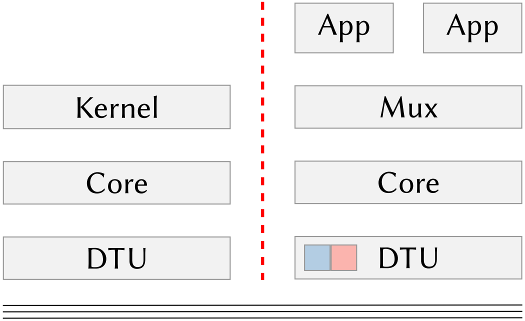
Local Reasoning?



Background \ Foreground	compute	memory	msgs	transfers
transfers	0.00	3.41	0.00	3.14
msgs	0.00	0.00	0.00	0.00
memory	0.00	3.43	0.00	2.68
compute	0.00	0.00	0.00	0.00

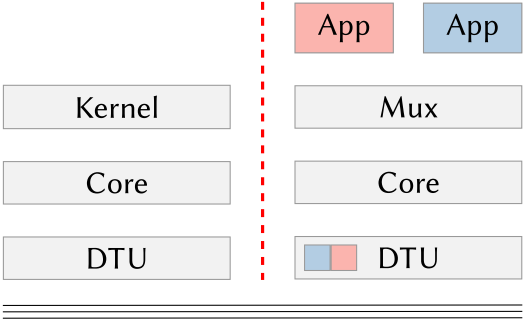
Network-on-Chip Regulation

- Added multiple *token-bucket registers* to DTU



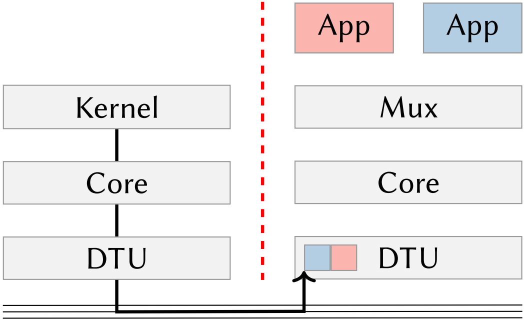
Network-on-Chip Regulation

- Added multiple *token-bucket registers* to DTU



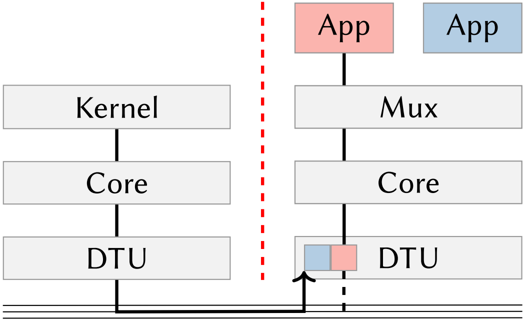
Network-on-Chip Regulation

- Added multiple *token-bucket registers* to DTU



Network-on-Chip Regulation

- Added multiple *token-bucket registers* to DTU
- Register contains:
  - `amount`: available bytes
  - `limit`: max. bytes
  - `rate`: bytes added per time



End-to-end Resource Limits

- M^3 is a microkernel-based OS
- Uses capabilities, similar to Composite [4]
- However: original M^3 did not enforce limits and had no policy

[4] Parmer et al.: Predictable interrupt management and scheduling in the Composite component-based system, RTSS 2008

End-to-end Resource Limits

1. Policy: distribution of resources

```
<app args="example" time="10ms" noc-bw="1GB/s"/>
```

End-to-end Resource Limits

1. Policy: distribution of resources

```
<app args="example" time="10ms" noc-bw="1GB/s"/>
```

2. Capabilities: fine-grained division/exchange of resources
   - *Resource manager* turns XML properties into capabilities
   - Starting application on tile requires a *tile capability*
   - Tile capability has quotas attached (CPU time, NoC bandwidth, ...)
   - *Derive* creates new capability with subset of quota

End-to-end Resource Limits

1. Policy: distribution of resources

   ```
   <app args="example" time="10ms" noc-bw="1GB/s"/>
   ```

2. Capabilities: fine-grained division/exchange of resources
   - *Resource manager* turns XML properties into capabilities
   - Starting application on tile requires a *tile capability*
   - Tile capability has quotas attached (CPU time, NoC bandwidth, …)
   - *Derive* creates new capability with subset of quota
3. Enforcement
   - Multiplexer enforces CPU time (with timer)
   - DTU enforces NoC bandwidth (with token-bucket register)

Evaluation

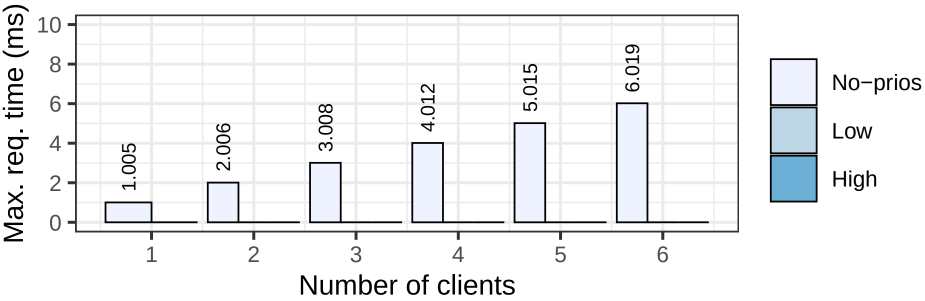
1. Cross-tile communication latency and jitter
2. Communication latency with per-priority endpoints
3. Local reasoning with NoC regulation
4. Fast and energy-efficient communication

Communication Latency with Per-Priority Endpoints

- Service on dedicated tile, 1-6 clients on other tiles
- Two receive endpoints: high (1 client) and low priority (rest of clients)
- Each request takes fixed amount of time: 1ms

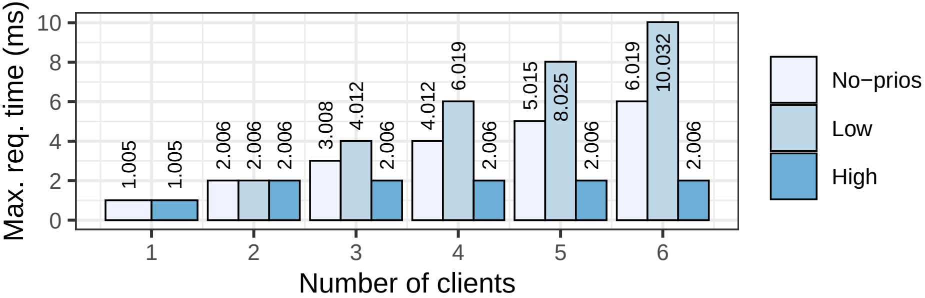
Communication Latency with Per-Priority Endpoints

- Service on dedicated tile, 1-6 clients on other tiles
- Two receive endpoints: high (1 client) and low priority (rest of clients)
- Each request takes fixed amount of time: 1ms

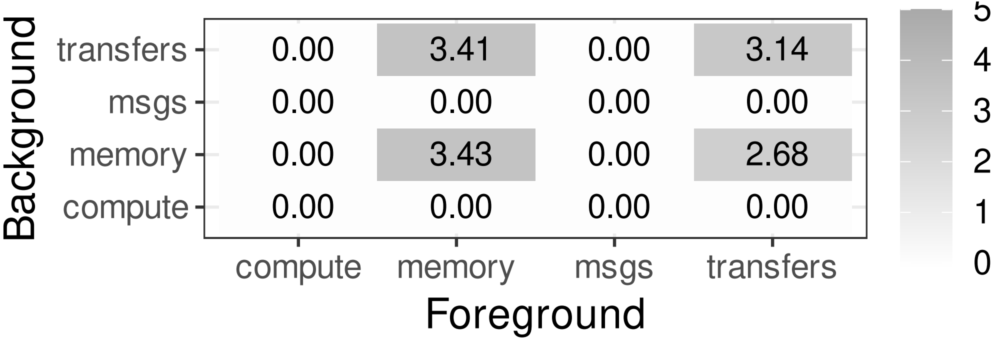


Communication Latency with Per-Priority Endpoints

- Service on dedicated tile, 1-6 clients on other tiles
- Two receive endpoints: high (1 client) and low priority (rest of clients)
- Each request takes fixed amount of time: 1ms



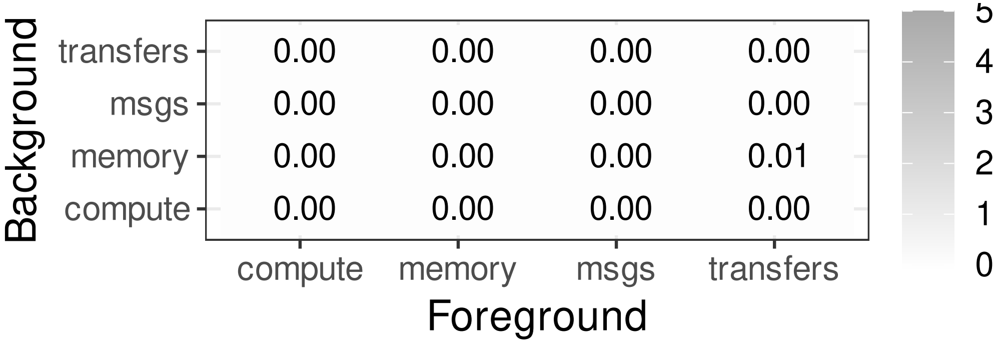
Local Reasoning without/with NoC Regulation



Background \ Foreground	compute	memory	msgs	transfers
transfers	0.00	3.41	0.00	3.14
msgs	0.00	0.00	0.00	0.00
memory	0.00	3.43	0.00	2.68
compute	0.00	0.00	0.00	0.00

Without NoC regulation

Local Reasoning without/with NoC Regulation



Background \ Foreground	compute	memory	msgs	transfers
transfers	0.00	0.00	0.00	0.00
msgs	0.00	0.00	0.00	0.00
memory	0.00	0.00	0.00	0.01
compute	0.00	0.00	0.00	0.00

With NoC regulation (rate = 8 MiB/s)

Future Work

- Real-time version of multiplexer with real-time scheduler
- Bounded request-handling times in kernel and services
- Hardware implementation of NoC regulation

Conclusion

- M^3 is a promising platform for cyber-physical systems
  - Designed for heterogeneous systems
  - Strong isolation between tiles
- This work demonstrates local reasoning as another benefit:
  - No shared hardware resources between tiles
  - Tiles can be specialized for real-time / best-effort without losing shared-system experience
  - NoC regulation to limit interference
- Source code of hardware and software is available:
  `https://github.com/Barkhausen-Institut/M3`

Artifact * AEC * Evaluated * RTAS *
Consistent * Well Documented * Easy to Reuse *

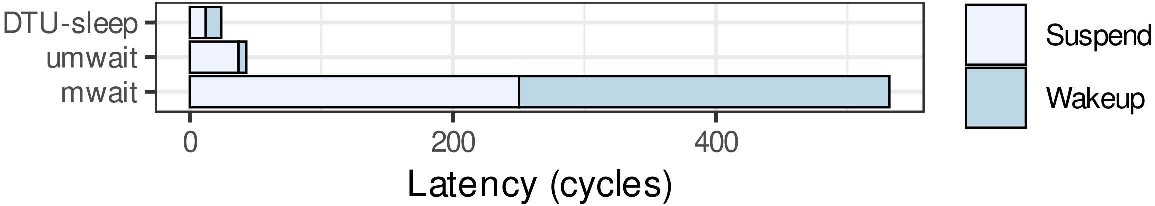
Backup Slides

Communication Latency and Jitter

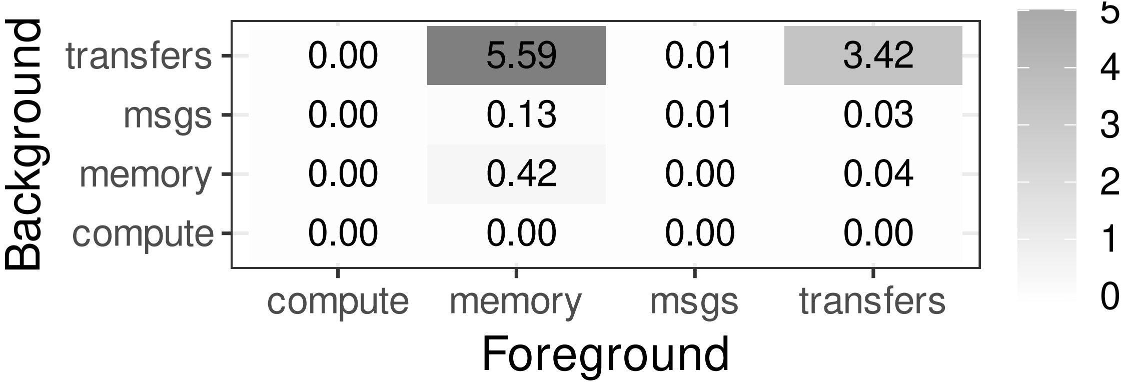
OS	Platform	avg	P99	min	max	σ
M^3	FPGA	537	675	484	3571	107
M^3	S-RISCV	319	316	316	3460	99
Linux	S-RISCV	16234	24824	11152	36578	3042
Linux	S-x86	15317	22773	10529	35112	1335
NOVA	S-x86	7058	7017	6919	130181	3899
M^3	S-x86	405	416	377	3347	93
L4Re	H-Arm	2605	2639	1622	22739	644
NOVA	H-x86	10261	10442	9958	55724	1408

Table: Round-trip latency for cross-core messaging on different hardware platforms and OSes (including outliers).

Application-Controlled Core Sleep

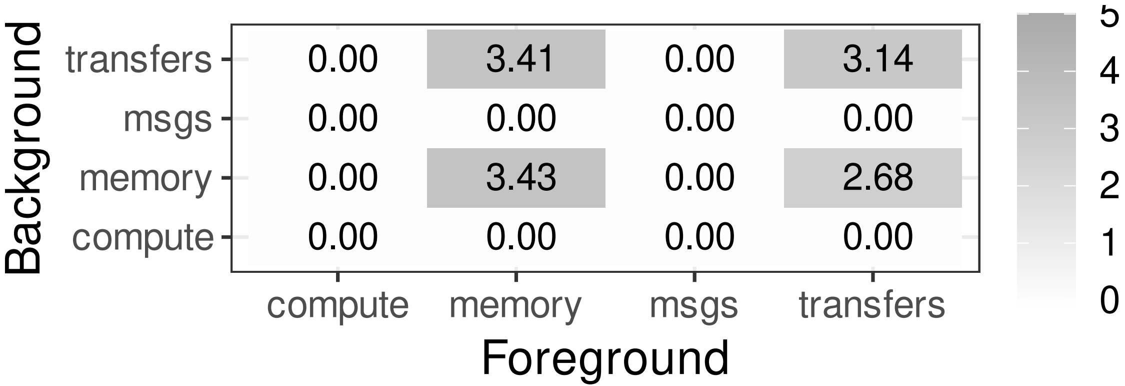


Tile-Interference on FPGA platform



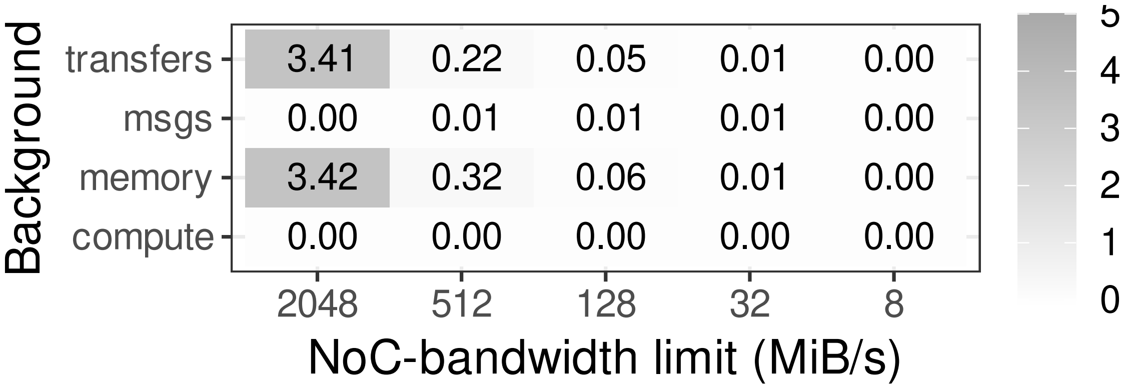
Background \ Foreground	compute	memory	msgs	transfers
transfers	0.00	5.59	0.01	3.42
msgs	0.00	0.13	0.01	0.03
memory	0.00	0.42	0.00	0.04
compute	0.00	0.00	0.00	0.00

Tile-Interference on gem5 platform



Background \ Foreground	compute	memory	msgs	transfers
transfers	0.00	3.41	0.00	3.14
msgs	0.00	0.00	0.00	0.00
memory	0.00	3.43	0.00	2.68
compute	0.00	0.00	0.00	0.00

Tile-Interference for Memory Accesses


Background
transfers
msgs
memory
compute
NoC-bandwidth limit (MiB/s)