barkhausen
institut

M^3: Applying Microkernel-Ideas to Hardware

Nils Asmussen

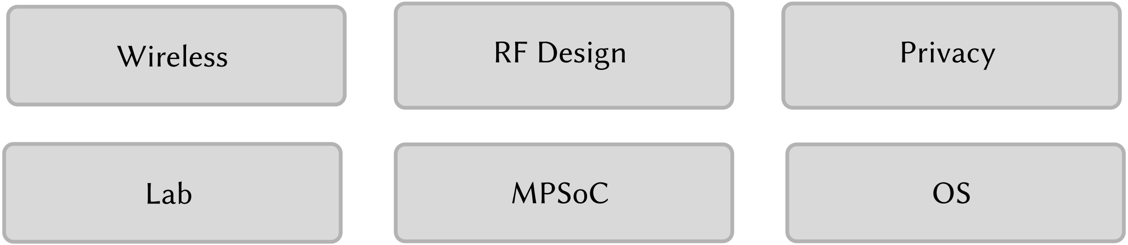
ROSS, 15th November 2021

Barkhausen Institut

- Research institute in Dresden, founded end of 2017
- Currently about 40 people
- Low-latency and secure IoT systems
- Focus on research and demonstrators

Barkhausen Institut

- Research institute in Dresden, founded end of 2017
- Currently about 40 people
- Low-latency and secure IoT systems
- Focus on research and demonstrators



Wireless

RF Design

Privacy

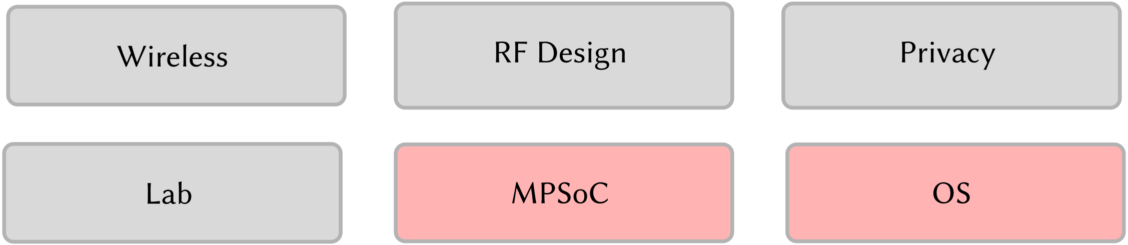
Lab

MPSoC

OS

Barkhausen Institut

- Research institute in Dresden, founded end of 2017
- Currently about 40 people
- Low-latency and secure IoT systems
- Focus on research and demonstrators



Wireless

RF Design

Privacy

Lab

MPSoC

OS

Motivation

- Microkernels in a nutshell
  - No isolation between components in monolithic OS
  - Single exploitable bug anywhere → game over
  - Microkernel-based systems split OS into isolated and unprivileged components
  - 96% of Linux CVEs would no longer be critical, 40% would be eliminated [1]

[1] S. Biggs, et al.: The Jury Is In: Monolithic OS Design Is Flawed. 9th Asia-Pacific Workshop on Systems (APSys'18), 2018

Motivation

- Microkernels in a nutshell
  - No isolation between components in monolithic OS
  - Single exploitable bug anywhere → game over
  - Microkernel-based systems split OS into isolated and unprivileged components
  - 96% of Linux CVEs would no longer be critical, 40% would be eliminated [1]
- Microkernel-based systems have proven valuable for other objectives:
  Low-noise execution, real time, flexibility, …

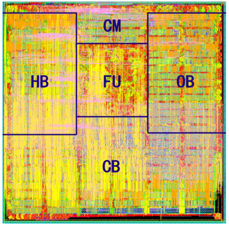
[1] S. Biggs, et al.: The Jury Is In: Monolithic OS Design Is Flawed. 9th Asia-Pacific Workshop on Systems (APSys'18), 2018

Motivation

- Microkernels in a nutshell
  - No isolation between components in monolithic OS
  - Single exploitable bug anywhere → game over
  - Microkernel-based systems split OS into isolated and unprivileged components
  - 96% of Linux CVEs would no longer be critical, 40% would be eliminated [1]
- Microkernel-based systems have proven valuable for other objectives: Low-noise execution, real time, flexibility, ...
- Recently, new challenges are coming from the hardware side
  - Heterogeneous systems
  - Third-party components
  - Security issues of complex general-purpose cores

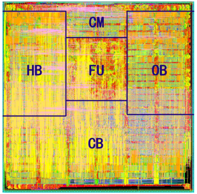
[1] S. Biggs, et al.: The Jury Is In: Monolithic OS Design Is Flawed. 9th Asia-Pacific Workshop on Systems (APSys'18), 2018

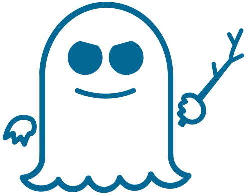
Hardware Complexity: Heterogeneity



- 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 compute units with different feature sets

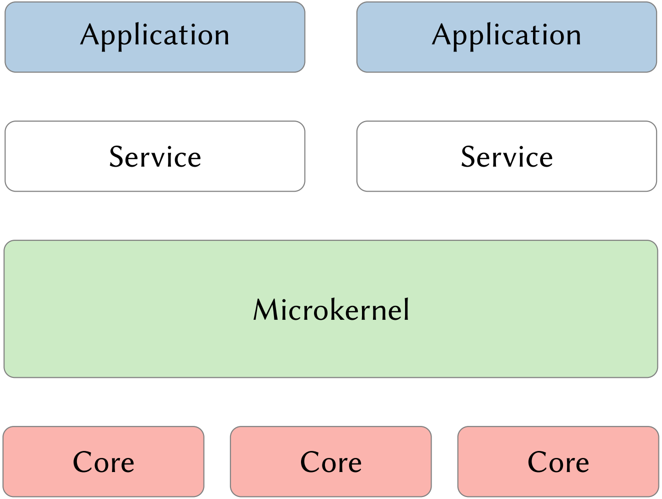
Hardware Complexity: Untrusted Hardware Components



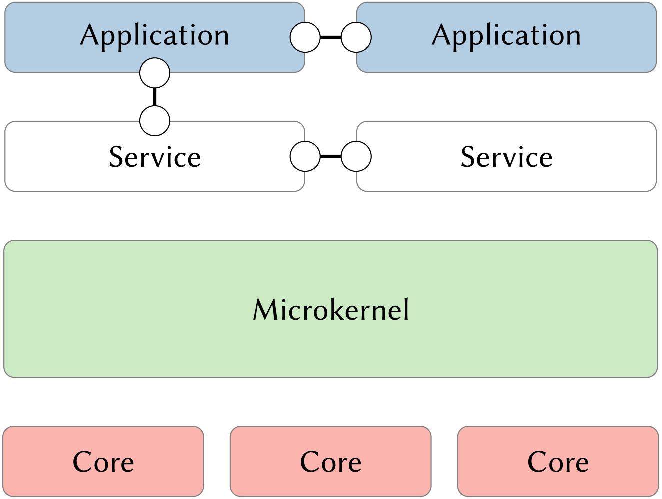


- Provided by third-party vendors
- Bug in such a component can compromise whole system (see Broadcom incident)
- Side channels in modern cores allow attackers to leak private data; some bypass all security measures of the core (address spaces, virtualization, . . .)
- Have been lurking in CPUs for many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

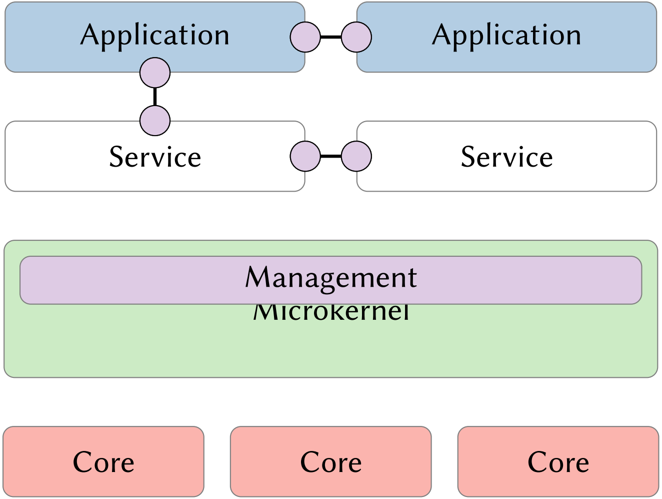
Microkernel-based System as Foundation



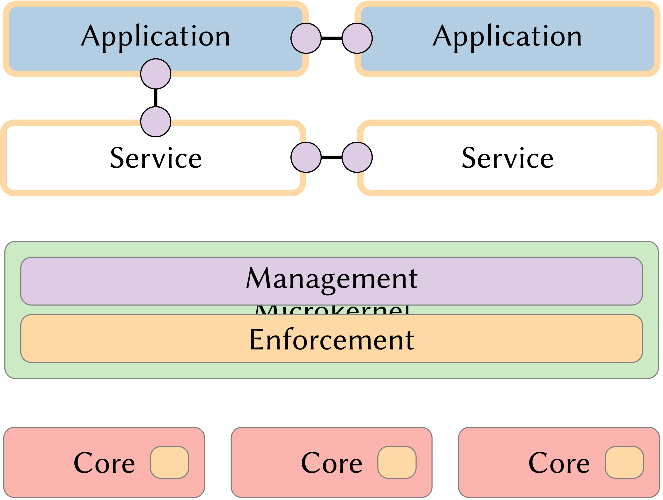
Microkernel-based System as Foundation



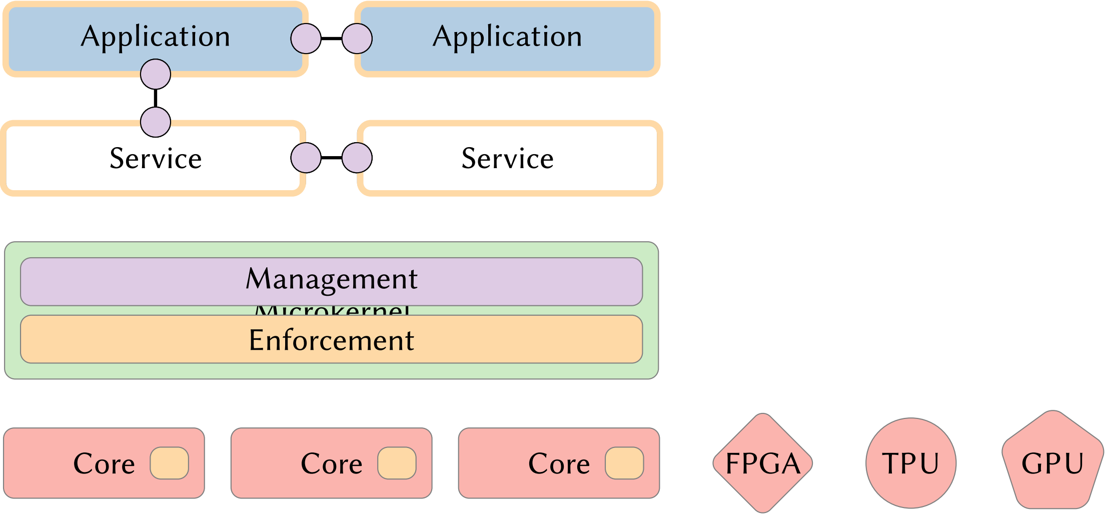
Microkernel-based System as Foundation



Microkernel-based System as Foundation



Microkernel-based System as Foundation



Application

Application

Service

Service

Management

Microkernel

Enforcement

Core

Core

Core

FPGA

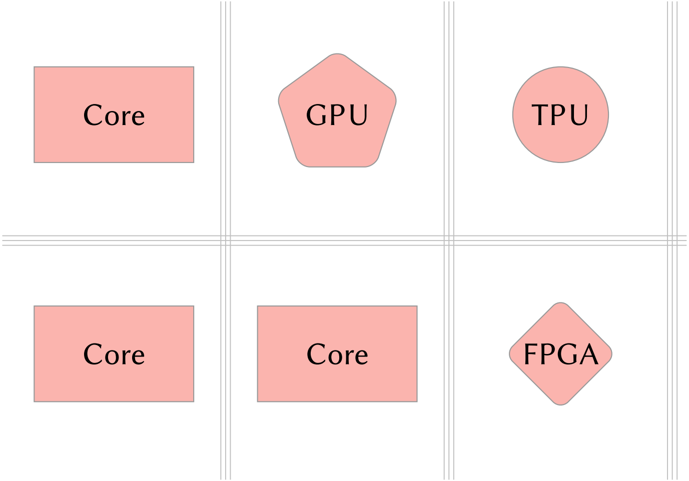
TPU

GPU

M^3 System Architecture [1]

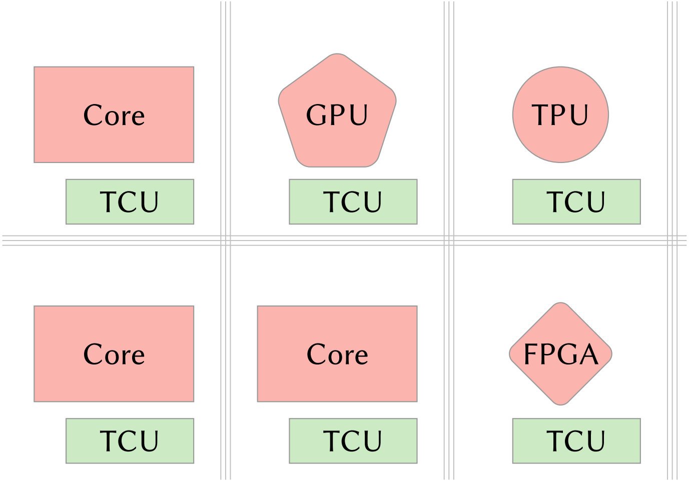
[1] Asmussen et al.; M^3: A Hardware/Operating-System Co-Design to Tame Heterogeneous Manycores, ASPLOS 2016

M^3 System Architecture [1]



[1] Asmussen et al.; M^3: A Hardware/Operating-System Co-Design to Tame Heterogeneous Manycores, ASPLOS 2016

M^3 System Architecture [1]

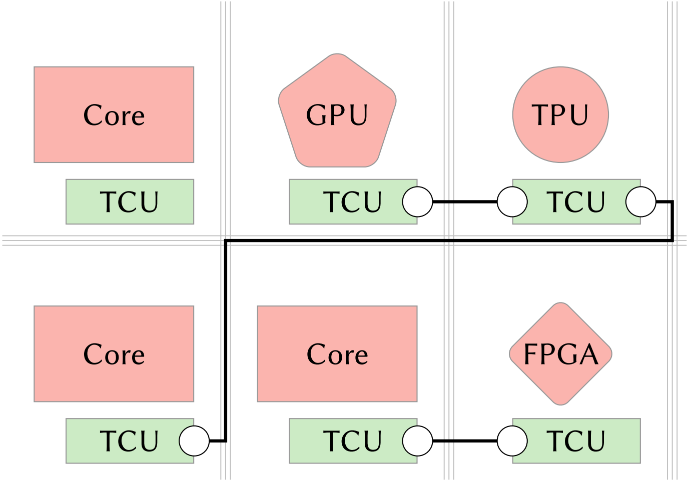


Key ideas:

- TCU as new hardware component

[1] Asmussen et al.; M^3: A Hardware/Operating-System Co-Design to Tame Heterogeneous Manycores, ASPLOS 2016

M^3 System Architecture [1]

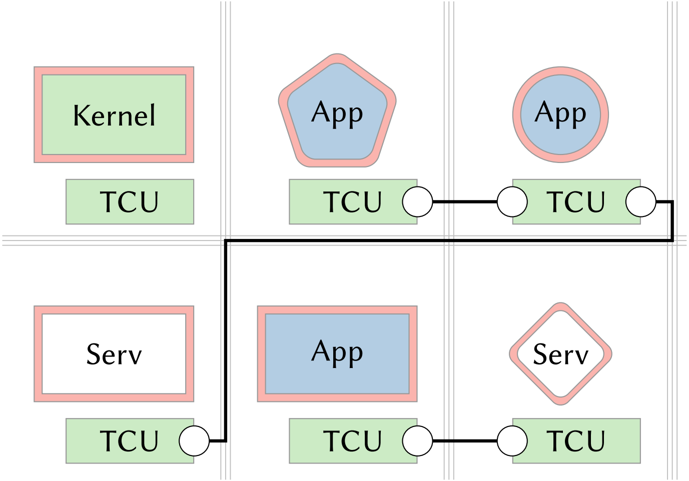


Key ideas:

- TCU as new hardware component

[1] Asmussen et al.; M^3: A Hardware/Operating-System Co-Design to Tame Heterogeneous Manycores, ASPLOS 2016

M^3 System Architecture [1]

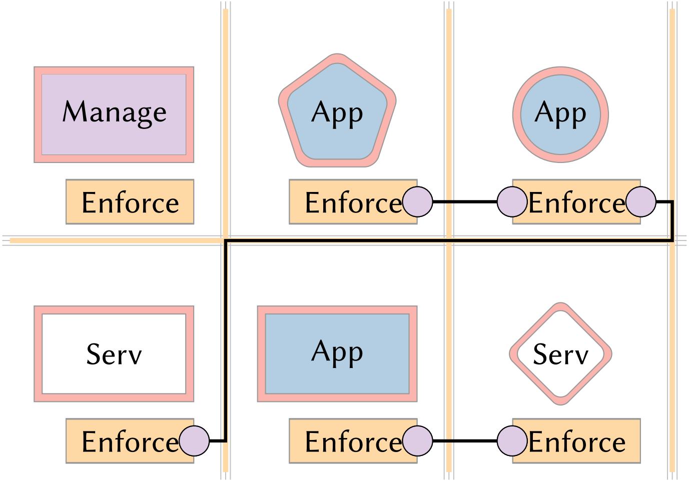


Key ideas:

- TCU as new hardware component
- Kernel on dedicated tile

[1] Asmussen et al.; M^3: A Hardware/Operating-System Co-Design to Tame Heterogeneous Manycores, ASPLOS 2016

M^3 System Architecture [1]

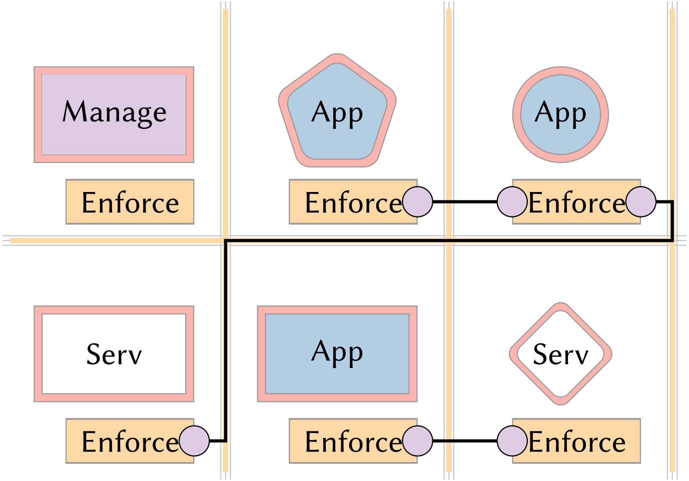


Key ideas:

- TCU as new hardware component
- Kernel on dedicated tile
- Kernel manages, TCU enforces

[1] Asmussen et al.; M^3: A Hardware/Operating-System Co-Design to Tame Heterogeneous Manycores, ASPLOS 2016

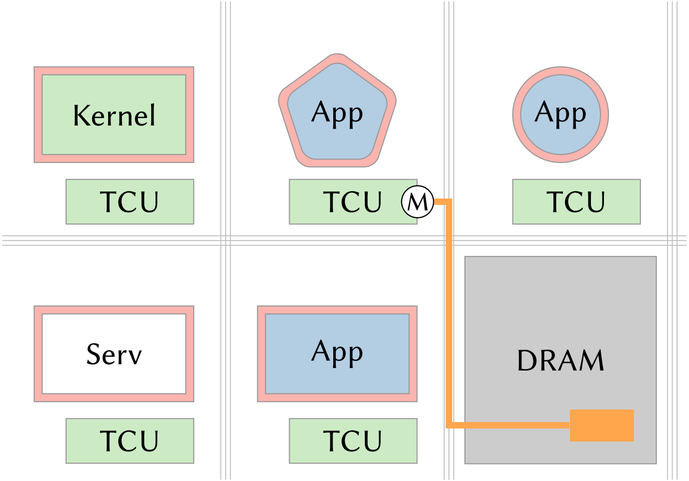
M^3 System Architecture [1]



μ-kernel-ideas applied to HW:

- μ-kernel contains essence of monolithic kernel
- TCU contains essence of μ-kernel

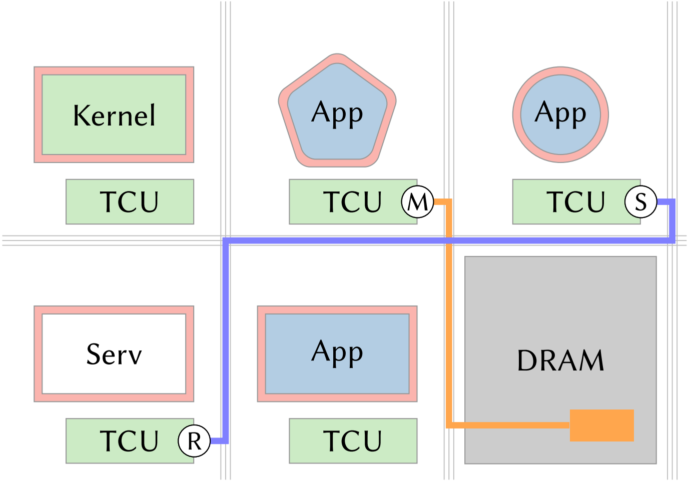
TCU-based Communication



TCU provides *endpoints* to:

- Access memory (contiguous range, byte granular)

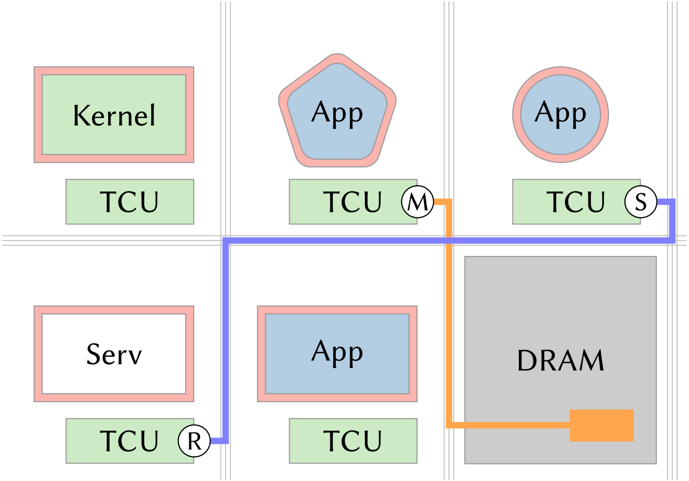
TCU-based Communication



TCU provides *endpoints* to:

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

TCU-based Communication

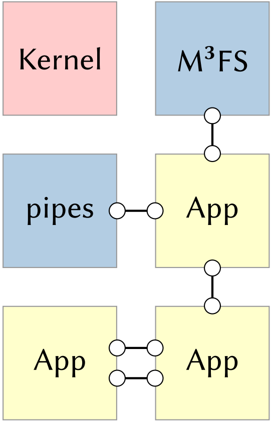


TCU provides *endpoints* to:

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

M3: The Operating System

- 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



M^3-based Projects

- **M^{3}x: Autonomous Accelerators via Context-Enabled Fast-Path Communication**
  Nils Asmussen, Michael Roitzsch, Hermann Härtig, USENIX ATC 2019
- **SemperOS: A Distributed Capability System**
  Matthias Hille, Nils Asmussen, Pramod Bhatotia, Hermann Härtig, USENIX ATC 2019
- **Untrusted Cores in a Shared System**
  Under review for ASPLOS 2022
- Secure communication between devices (WIP)
- Compiler-based separation of components (WIP)

OS Service Access for all Tiles

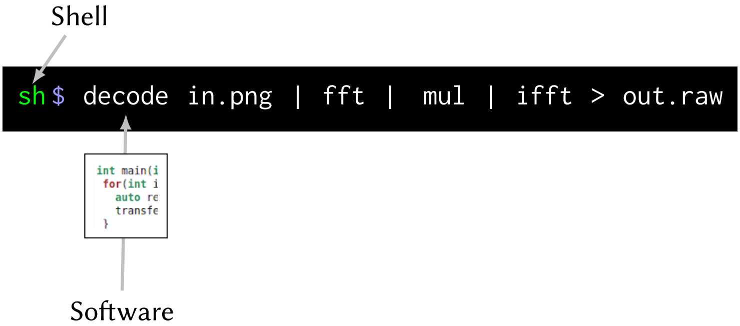
```
sh $ decode in.png | fft |  mul | ifft > out.raw
```

OS Service Access for all Tiles

Shell

```
sh $ decode in.png | fft | mul | ifft > out.raw
```

OS Service Access for all Tiles

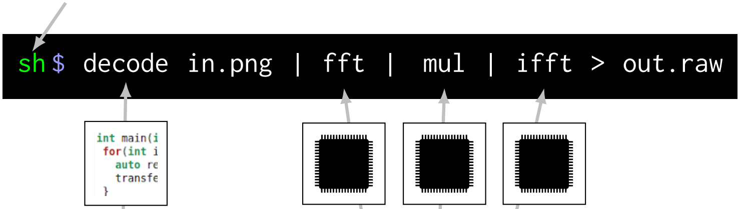


```
sh $ decode in.png | fft | mul | ifft > out.raw
```

```
int main(i
 for(int i
   auto re
   transfe
 }
```

OS Service Access for all Tiles

Shell



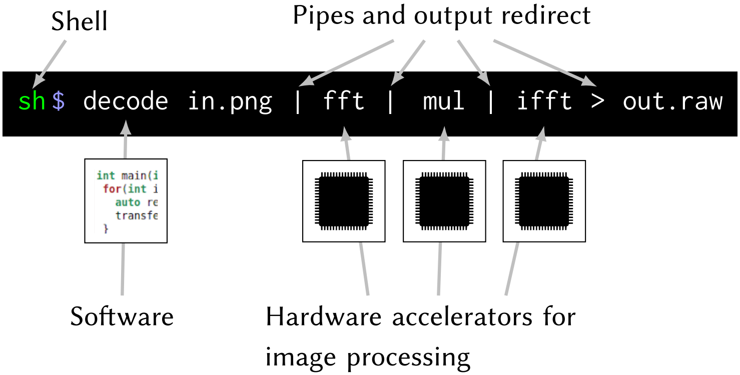
```
sh $ decode in.png | fft | mul | ifft > out.raw
```

```
int main(i
 for(int i
   auto re
   transfe
 }
```

Software

Hardware accelerators for image processing

OS Service Access for all Tiles



Shell

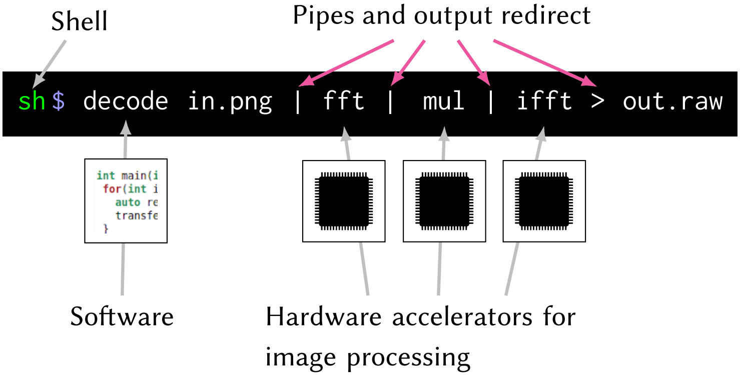
Pipes and output redirect

```
sh $ decode in.png | fft | mul | ifft > out.raw
```

Software

Hardware accelerators for image processing

OS Service Access for all Tiles



Shell

Pipes and output redirect

```
sh $ decode in.png | fft | mul | ifft > out.raw
```

```
int main(i
 for(int i
   auto re
   transfe
 }
```

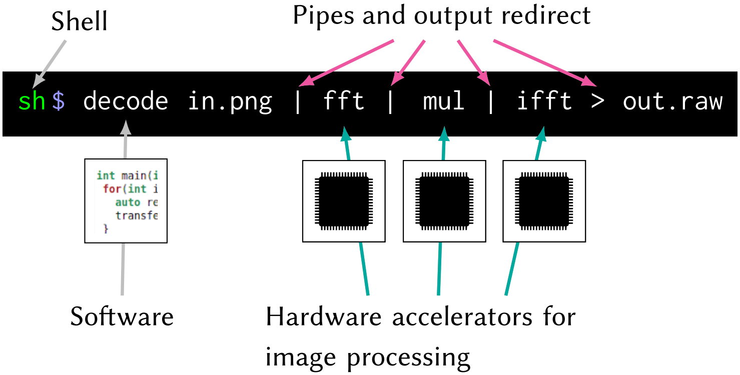
Software

Hardware accelerators for image processing

Challenges:

- OS must provide generic protocols

OS Service Access for all Tiles



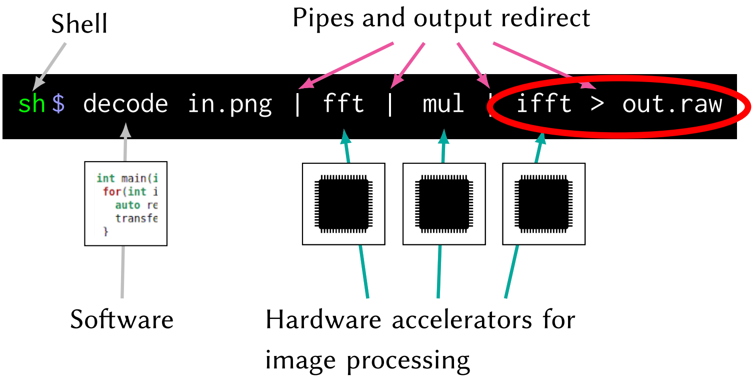
```
sh $ decode in.png | fft | mul | ifft > out.raw
```

Shell
Pipes and output redirect
int main(i
for(int i
auto re
transfe
}
Software
Hardware accelerators for image processing

Challenges:

- OS must provide generic protocols
- Accelerators need support for protocols

OS Service Access for all Tiles



Shell

Pipes and output redirect

```
sh $ decode in.png | fft | mul | ifft > out.raw
```

```
int main(i
 for(int i
   auto re
   transfe
 }
```

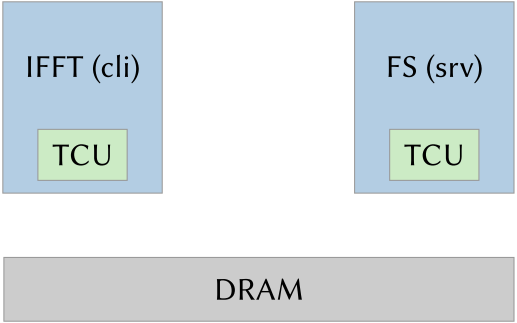
Software

Hardware accelerators for image processing

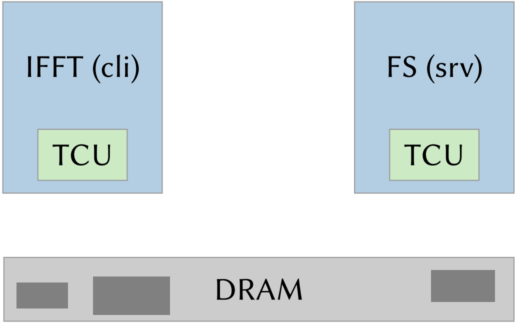
Challenges:

- OS must provide generic protocols
- Accelerators need support for protocols

Generic Protocol



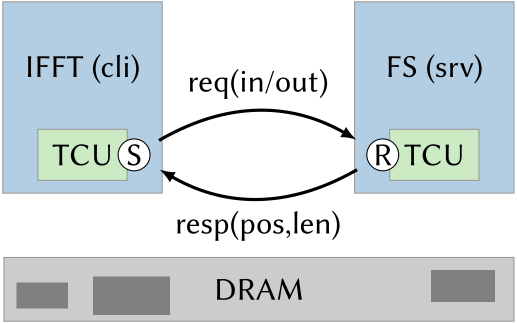
Generic Protocol



File protocol:

- Data in memory

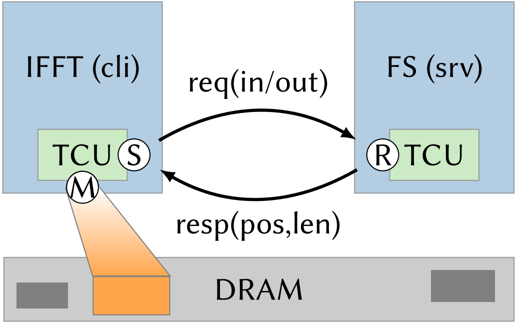
Generic Protocol



File protocol:

- Data in memory
- Msg channel between client and server
  - `req(in)` for next input piece
  - `req(out)` for next output piece

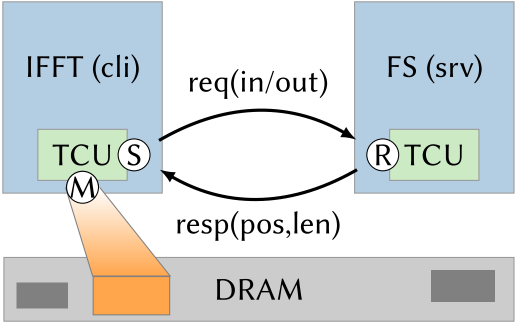
Generic Protocol



File protocol:

- Data in memory
- Msg channel between client and server
  - `req(in)` for next input piece
  - `req(out)` for next output piece
- Server configures client's memory EP

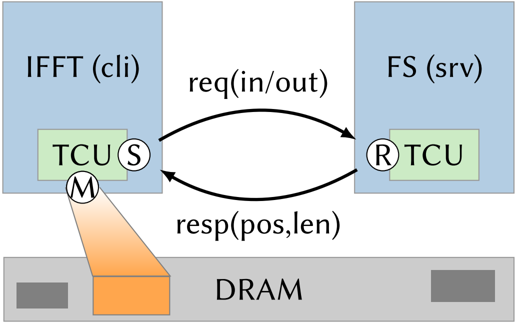
Generic Protocol



File protocol:

- Data in memory
- Msg channel between client and server
  - `req(in)` for next input piece
  - `req(out)` for next output piece
- Server configures client's memory EP
- Client accesses data via TCU

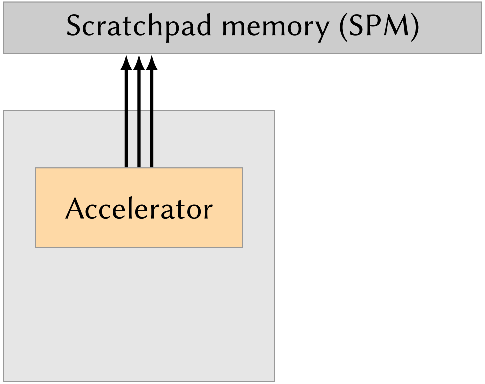
Generic Protocol



File protocol:

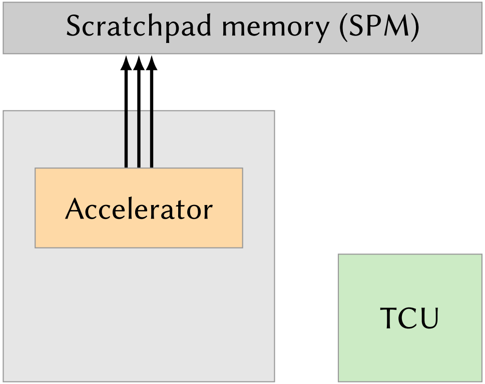
- Data in memory
- Msg channel between client and server
  - `req(in)` for next input piece
  - `req(out)` for next output piece
- Server configures client’s memory EP
- Client accesses data via TCU
- Used by *all* tiles

Additions to Accelerator



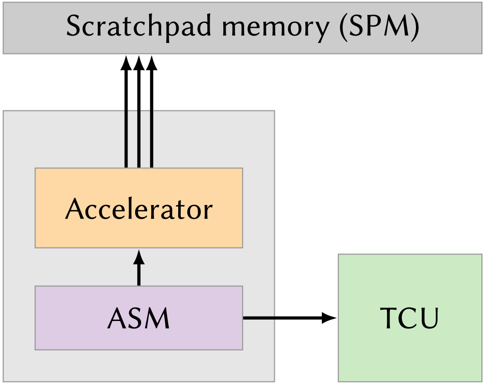
Off-the-shelf accelerators

Additions to Accelerator



Off-the-shelf accelerators

Additions to Accelerator

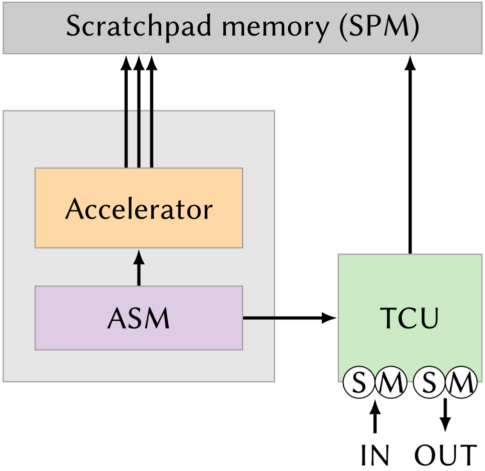


Off-the-shelf accelerators

Accelerator Support Module (ASM):

- Interacts with TCU and accelerator

Additions to Accelerator

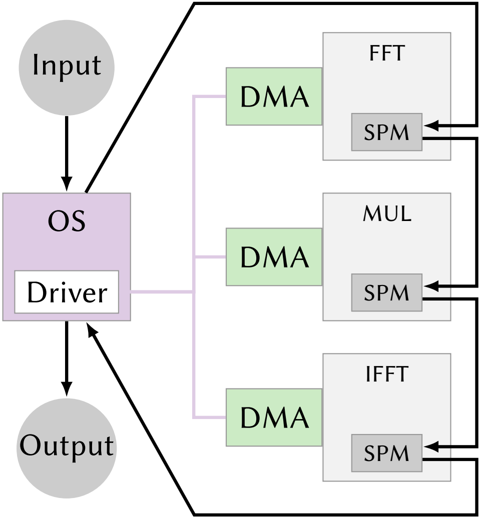


Off-the-shelf accelerators

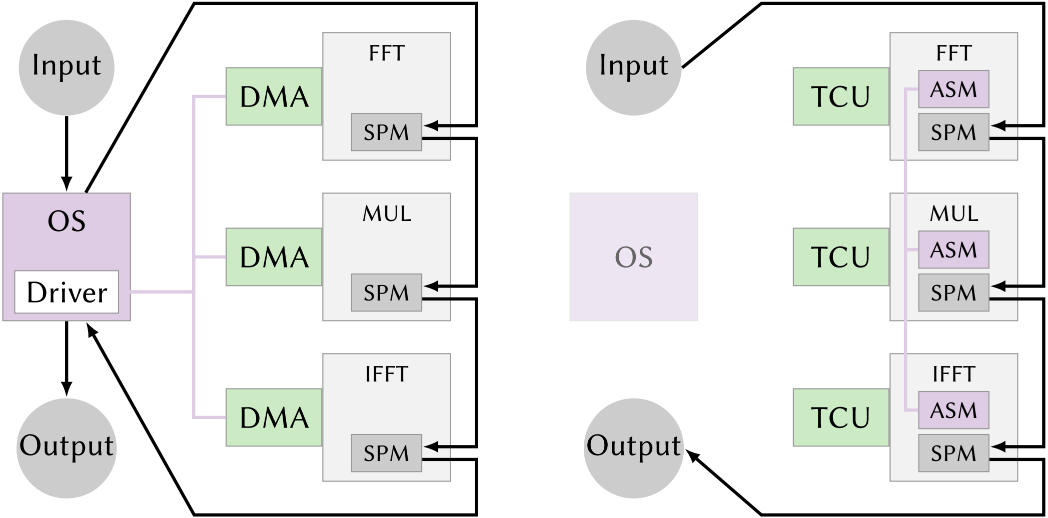
Accelerator Support Module (ASM):

- Interacts with TCU and accelerator
- Implements file protocol for input and output channel

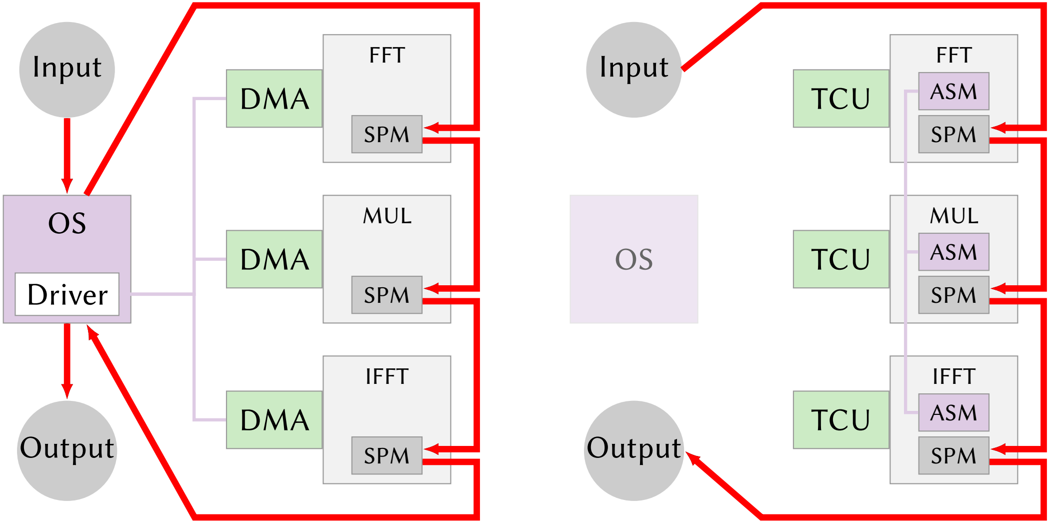
Assisted vs. Autonomous



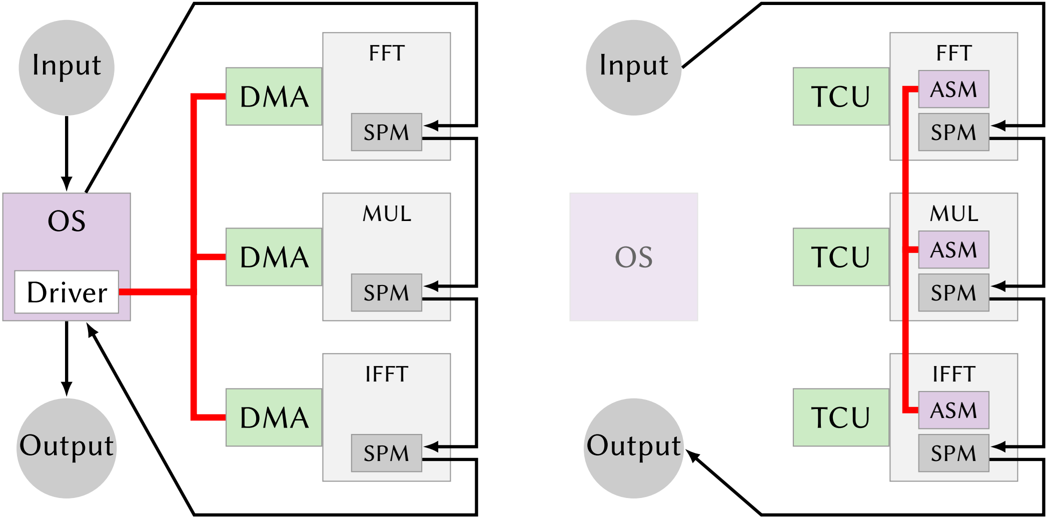
Assisted vs. Autonomous



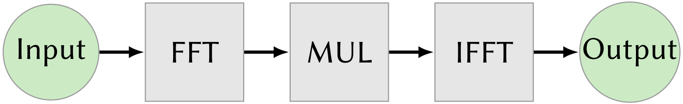
Assisted vs. Autonomous



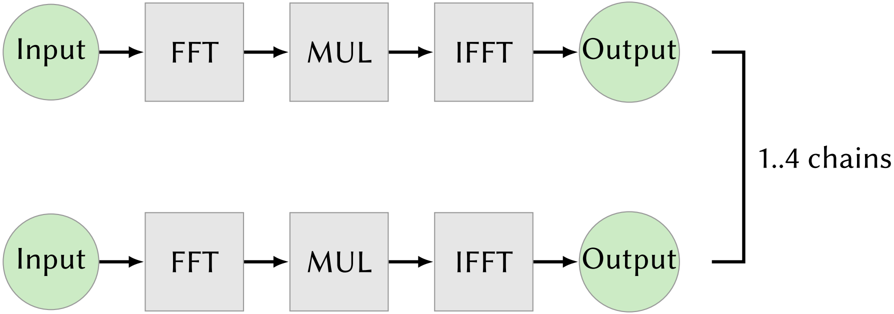
Assisted vs. Autonomous



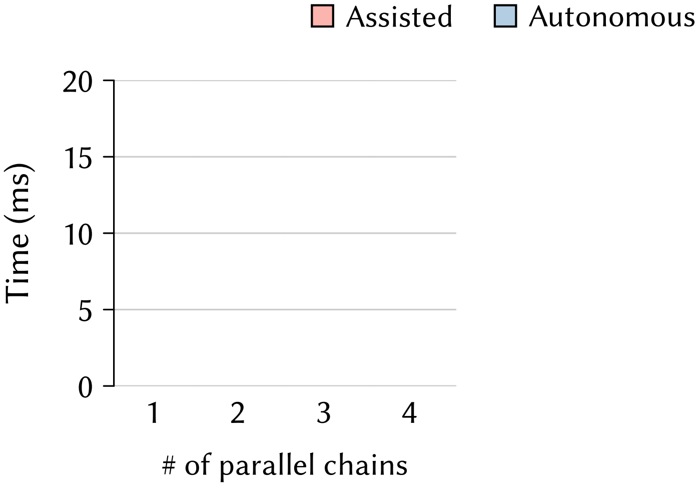
Accelerator Chains: Evaluation



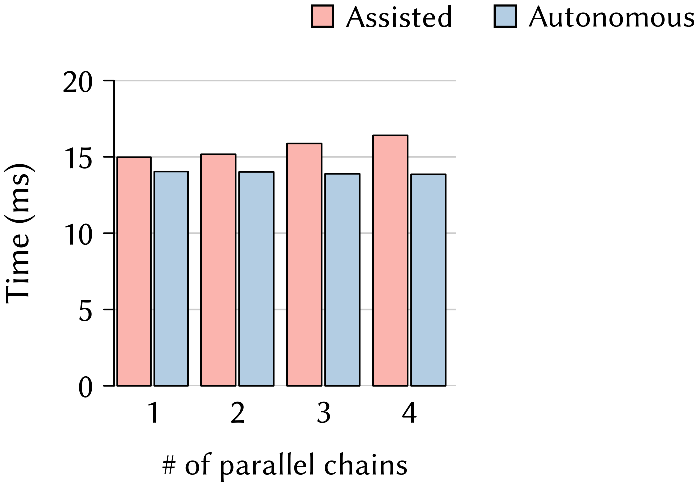
Accelerator Chains: Evaluation



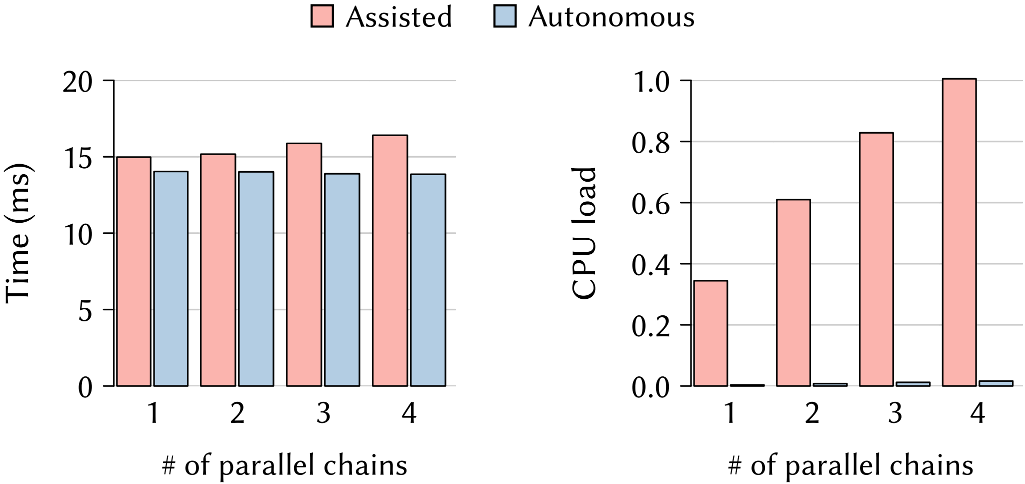
Accelerator Chains: Results



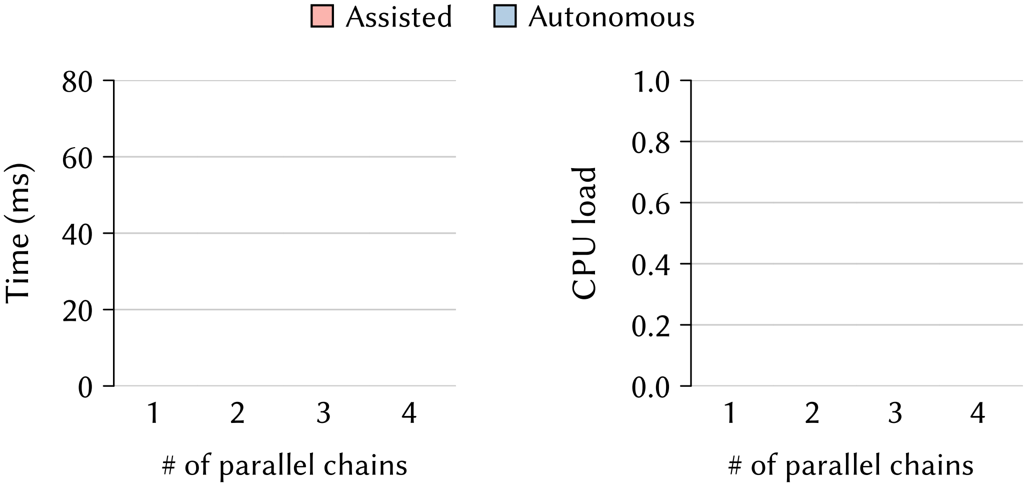
Accelerator Chains: Results



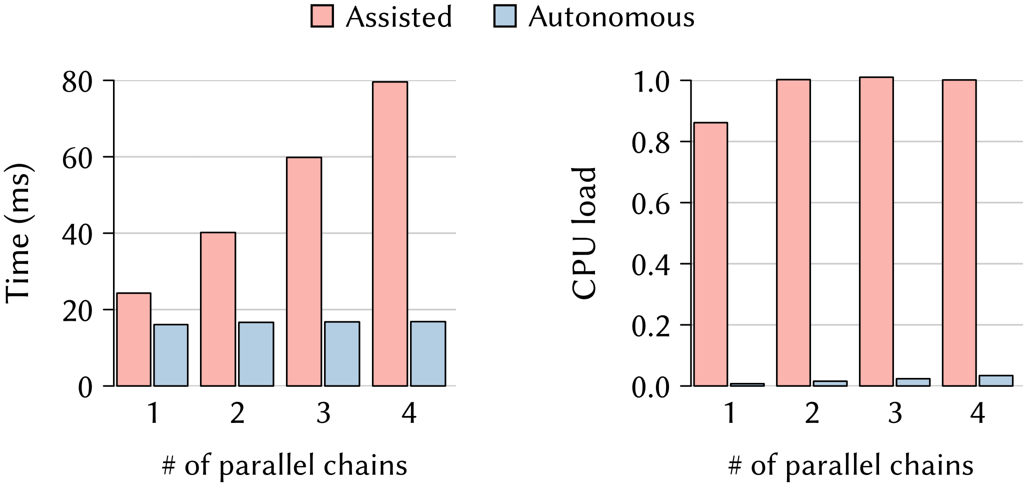
Accelerator Chains: Results



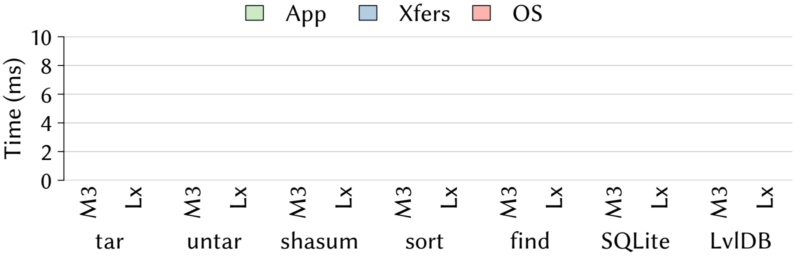
Accelerator Chains: Results (PCIe-like Latency)



Accelerator Chains: Results (PCIe-like Latency)

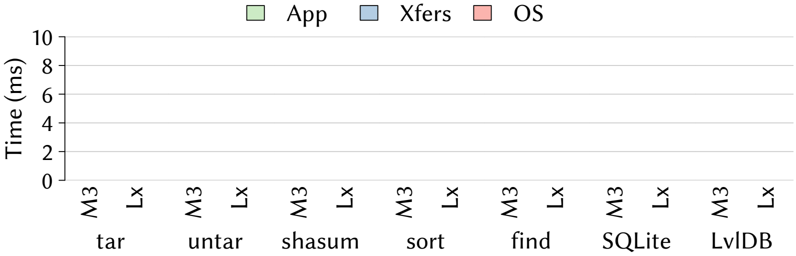


Performance Comparison with Linux

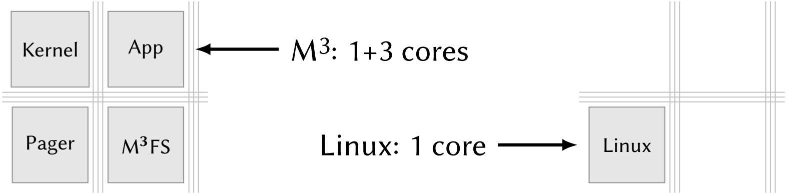


- M^3 vs. Linux 4.10
- Traced on Linux, replayed on M^3
- M^3FS vs. Linux tmpfs

Performance Comparison with Linux



- M^3 vs. Linux 4.10
- Traced on Linux, replayed on M^3
- M^3FS vs. Linux tmpfs



Kernel

App

Pager

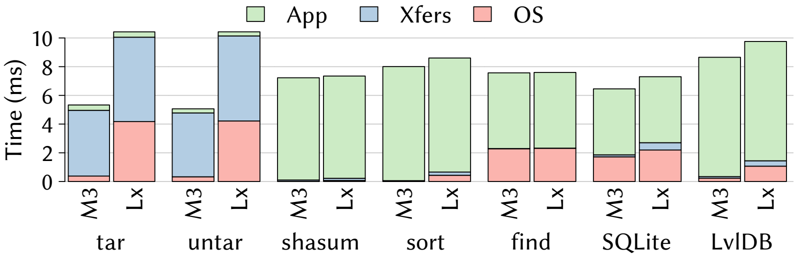
M^3FS

M^3: 1+3 cores

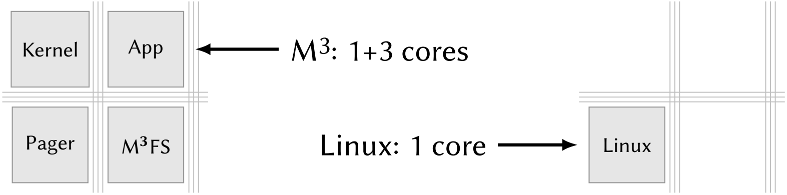
Linux: 1 core

Linux

Performance Comparison with Linux



- M^3 vs. Linux 4.10
- Traced on Linux, replayed on M^3
- M^3FS vs. Linux tmpfs



Kernel

App

Pager

M^3FS

M^3: 1+3 cores

Linux: 1 core

Linux

Conclusion

- M^3 applies microkernel ideas to hardware
  - Add trusted communication component (TCU) next to each compute unit
  - TCU contains essence of a traditional microkernel
  - Microkernel-based system called M^3 takes advantage of TCU

Conclusion

- M^3 applies microkernel ideas to hardware
  - Add trusted communication component (TCU) next to each compute unit
  - TCU contains essence of a traditional microkernel
  - Microkernel-based system called M^3 takes advantage of TCU
- M^3x introduced accelerator chaining
  - Improves performance compared to traditional approach
  - Reduces CPU load to almost zero → accelerators run autonomously