M³: Taking Microkernels to the Next Level

Nils Asmussen

FOSDEM, 02/02/2020, Brussels
About Me

- Nils Asmussen
- PhD last year at the OS chair of the TU Dresden
- Low-level system programming and microkernels
- Worked on several microkernel-based OSes in the past
  - NRE, userland for NOVA: https://github.com/TUD-OS/NRE
  - M³, presented today: https://github.com/TUD-OS/M3
- Since 2019 at the Barkhausen Institut
Barkhausen Institut

- Research institute in Dresden, founded end of 2017
- Currently about 30 people
- Low-latency and secure IoT systems
- Focus on research and demonstrators
Barkhausen Institut

- Research institute in Dresden, founded end of 2017
- Currently about 30 people
- Low-latency and secure IoT systems
- Focus on research and demonstrators

Wireless | RF Design | Privacy
---|---|---
Lab | MPSoC | OS
Motivation

- Microkernel-based systems have proven valuable for several objectives
  - Security
  - Robustness
  - Real time
  - Flexibility

- Recently, new challenges are coming from the hardware side
  - Heterogeneous systems
  - Third-party components
  - Security issues of complex general-purpose cores
Heterogeneous Systems

- 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
Third-party Components

- Market pressure forces us to integrate third-party components
- We should not trust these components
- Currently, often no isolation between them
- Bug in such a component can compromise whole system (see Broadcom incident)
Security Issues of Complex General-purpose Cores

- 20 known attacks (and counting …)
- Allow to leak private data, sometimes bypassing all security measures of the core
- Mitigations exist, but these are complex and costly
- These security holes have been lurking in CPUs for many years
- Should we still trust these complex cores to properly enforce the isolation between different software components?
Microkernel-based System as Foundation

Diagram:
- Microkernel
  - Core
  - Service
  - Application
Microkernel-based System as Foundation

Application

Service

Management

Microkernel

Core

Core

Core
Microkernel-based System as Foundation
Microkernel-based System as Foundation

Application → Service → Management → Enforcement

Core → Core → Core → FPGA → TPU → GPU
Outline

1. The New System Architecture
2. $M^3$: The Operating System
3. What are the Benefits?
1. The New System Architecture

2. $M^3$: The Operating System

3. What are the Benefits?
Hardware/Operating System Co-Design

Key ideas:
- TCU as new hardware component
- Kernel on dedicated PE
- Kernel manages, TCU enforces
Key ideas:
- TCU as new hardware component
- Kernel on dedicated PE
- Kernel manages, TCU enforces
Hardware/Operating System Co-Design

Key ideas:
- TCU as new hardware component
- Kernel on dedicated PE
- Kernel manages, TCU enforces
Key ideas:

- TCU as new hardware component
Hardware/Operating System Co-Design

Key ideas:
- TCU as new hardware component
Hardware/Operating System Co-Design

Key ideas:
- TCU as new hardware component
- Kernel on dedicated PE
Hardware/Operating System Co-Design

Key ideas:
- TCU as new hardware component
- Kernel on dedicated PE
- Kernel manages, TCU enforces
Hardware/Operating System Co-Design

Takes $\mu$-kernels to the next level:

- TCU as secure foundation
Hardware/Operating System Co-Design

Takes \( \mu \)-kernels to the next level:

- TCU as secure foundation
- Heterogeneity:
  Uniform interface
Hardware/Operating System Co-Design

Takes $\mu$-kernels to the next level:

- TCU as secure foundation
- Heterogeneity:
  Uniform interface
- Untrusted HW comp.:
  Protected by TCU
Hardware/Operating System Co-Design

Takes $\mu$-kernels to the next level:

- TCU as secure foundation
- Heterogeneity:
  Uniform interface
- Untrusted HW comp.:
  Protected by TCU
- Side channels:
  Physical isolation
TCU provides endpoints to:

- Access memory (contiguous range, byte granular)
TCU provides *endpoints* to:

- Access memory (contiguous range, byte granular)
- Receive messages into a receive buffer
- Send messages to a receiving endpoint
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
TCU-based isolation:
- Additional protection layer
TCU-based isolation:
- Additional protection layer
TCU-based isolation:
- Additional protection layer
- Only kernel tile can establish communication channels
TCU-based isolation:
- Additional protection layer
- Only kernel tile can establish communication channels
- User tiles can only use established channels
Outline

1. The New System Architecture
2. $M^3$: The Operating System
3. What are the Benefits?
OS Design

- **M³**: Microkernel-based system for het. manycores (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
M³ System Call

User tile

App
TCU

Kernel tile

Kernel
TCU

R
M³ System Call

User tile

App

TCU

S

Kernel tile

Kernel

TCU

R
M³ System Call

User tile

Kernel tile

App

Kernel

TCU

TCU

S

R
M³ System Call

User tile

App

TCU

S

Kernel tile

Kernel

TCU

R


M$^3$ System Call

User tile

App

TCU

S

Kernel tile

Kernel

TCU

R

14 / 23
OS Service Access

File Protocol:
- Used for: files, pipes, ...
- Data in memory
- Msg channel between client and server

Client
- TCU

Server
- TCU

DRAM

req(in)
for next input piece

req(out)
for next output piece

resp(pos, len)
File Protocol:
- Used for: files, pipes, …
OS Service Access

File Protocol:
- Used for: files, pipes, …
- Data in memory
OS Service Access

File Protocol:

- Used for: files, pipes, …
- Data in memory
- Msg channel between client and server
  - req(in) for next input piece
  - req(out) for next output piece
**File Protocol:**

- Used for: files, pipes, …
- 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
OS Service Access

File Protocol:
- Used for: files, pipes, ...
- 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
Outline

1. The New System Architecture
2. M³: The Operating System
3. What are the Benefits?
Example System

Kernel

Core

Core

Core

Core

Core

5G

TCU

TCU

TCU

TCU

TCU

TCU
Example System
Example System
Example System – TCB
Example System – TCB

- Kernel
- Control
- Monitor
- FS
- Net
- Core
- Core
- Core
- Core
- Core
- 5G

TCU

TCU

TCU

TCU

TCU

TCU
Example System – TCB
Example System – Untrusted Core

- Kernel Data
- Control Core
- Monitor OoO
- FS Data Core
- Net Core
- 5G

TCU

TCU

TCU

TCU

TCU

TCU
Example System – Untrusted Core

Kernel
Data
Core
TCU

Control
Core
TCU

Monitor
OoO
FS
Data
Core
TCU

Net
Core
TCU

5G
TCU
Example System – Sharing (WIP)
Prototype Platforms

- gem5 simulator

Global frequency set at 10000000000 ticks per second
info: kernel located at: build/gem5-x86-64-release/bin/kernel
info: kernel located at: build/gem5-x86-64-release/bin/rtmx
info: kernel located at: build/gem5-x86-64-release/bin/rtmx
info: kernel located at: build/gem5-x86-64-release/bin/rtmx
warn: DRAM device capacity (49152 Megabytes) does not match the address range assigned (4896 Megabytes)
info: No kernel set for full system simulation. Assuming you know what you're doing
info: No kernel set for full system simulation. Assuming you know what you're doing
platform.com 1:device: Listening for connections on port 3456
  0: p0:remote gdb: listening for remote gdb on port 7600
  0: p01:remote gdb: listening for remote gdb on port 7601
  0: p02:remote gdb: listening for remote gdb on port 7602
  0: p03:remote gdb: listening for remote gdb on port 7603
warn: CoherencyError p04:char has no snooping ports attached
info: Loaded `root` to 0x0000000000000000...
info: Loaded `root` to 0x00000000000000000000000000000156
info: Loaded `root` to 0x00000000000000000000000000000156
info: Loaded `root` to 0x00000000000000000000000000000156
info: Loaded `root` to 0x00000000000000000000000000000156
info: Loaded `root` to 0x00000000000000000000000000000156
info: Entering event queue @ 0. Starting simulation...
[kernel 0e] Kernel is ready.
Hello World
Hello World
[kernel 0e] Shutting down.
Setting @ tick 835593000000 because @exit instruction encountered
Prototype Platforms

Prototype Platforms

gem5 simulator

FPGA
Microkernels are great!

Their ideas can also be applied to hardware:
- Trusted communication unit per tile
- Isolated software and hardware components on top

Has several additional benefits:
- Allows to securely integrate untrusted third-party components
- Prevents (known) side-channel attacks by physical isolation
- Simplifies heterogeneous systems by uniform interface

\( M^3 \) is available at https://github.com/TUD-OS/M3, gem5 extensions at https://github.com/TUD-OS/gem5-dtu