µBPF: Using eBPF for Microcontroller Compartmentalization

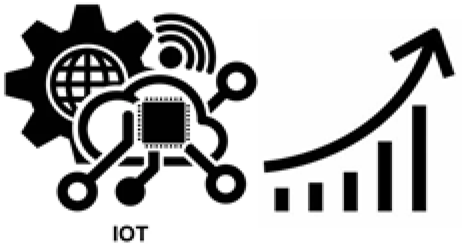
Szymon Kubica Marios Kogias

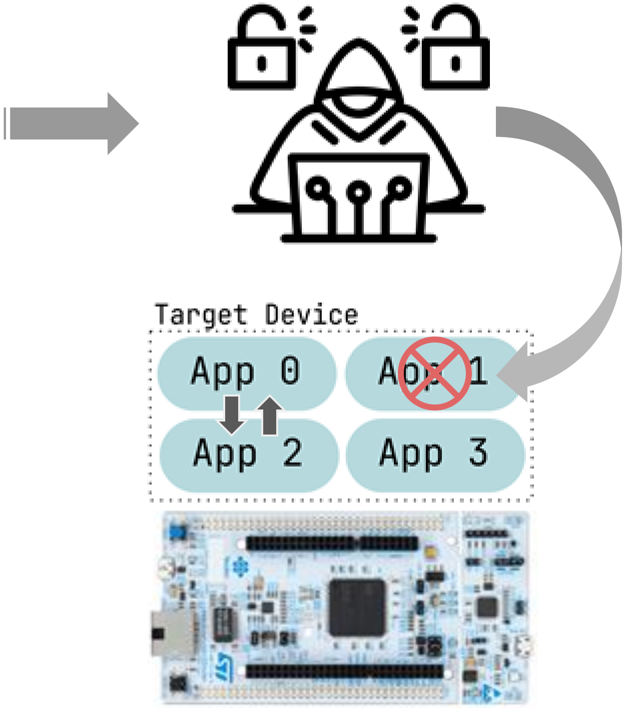
IMPERIAL

eBPF ‘24 Workshop
August 4, 2024

SIGCOMM 2024
— SYDNEY —

Problem: Security in the Internet of Things

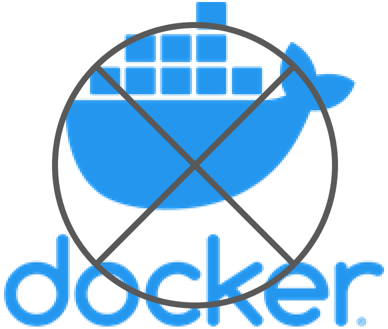




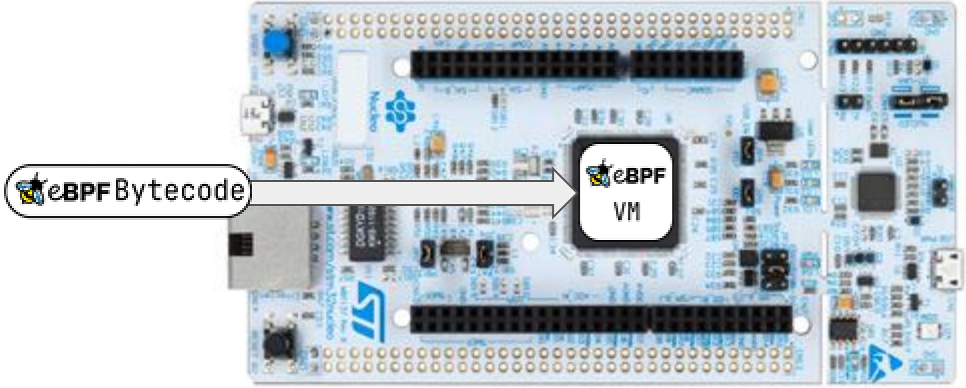
Solution: **compartmentalization**

- isolated components
- safe communication
- fault isolation

Solution: Virtualization for Microcontrollers



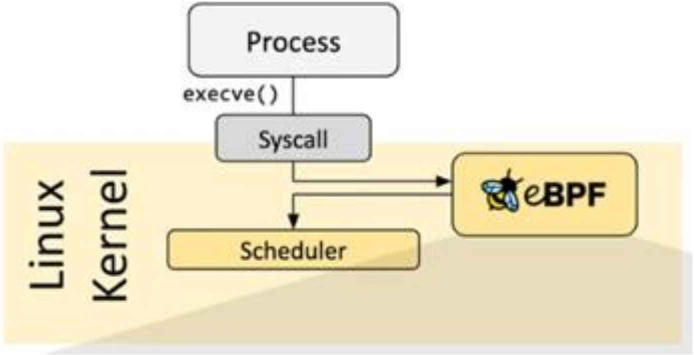
But, IoT devices are **low-end**



Software Isolation with eBPF

eBPF

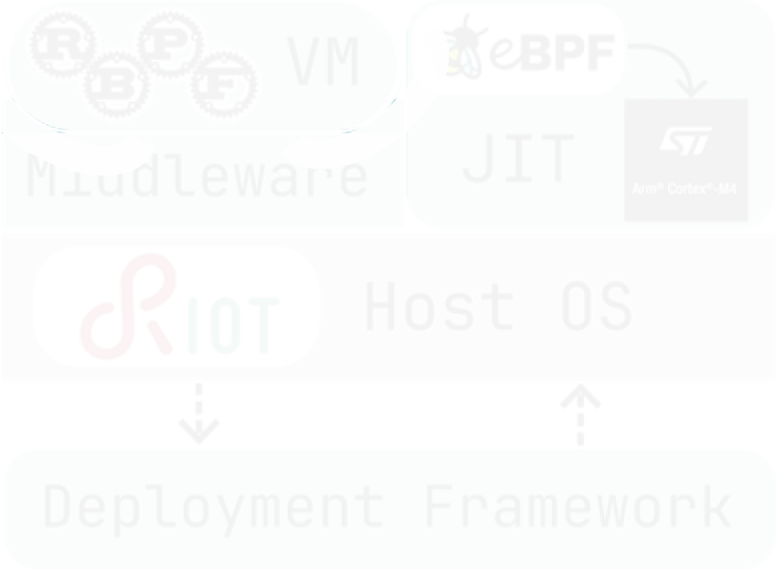
- Sandboxed execution
- Helper functions for OS access
- Simple instruction set
- Verifiable by design*



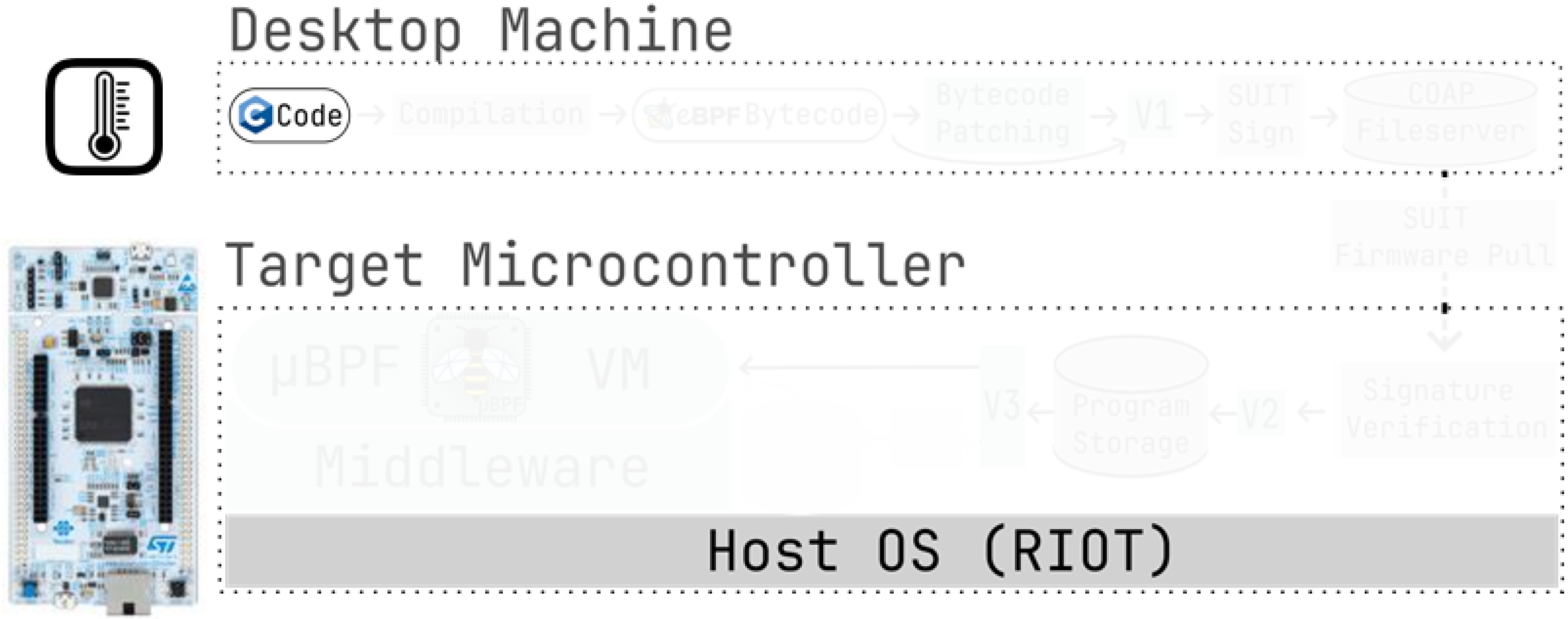
Architecture

Components

- eBPF VM based on **rbpf**
- embedded server hosted on **RIOT**
- eBPF to **ARMv7-eM** JIT compiler
- CLI deployment framework



Program Deployment & Execution Pipeline



Desktop Machine

Code → Compilation → eBPF Bytecode → Bytecode Patching → V1 → SUIT Sign → COAP Fileserver

SUIT Firmware Pull

Target Microcontroller

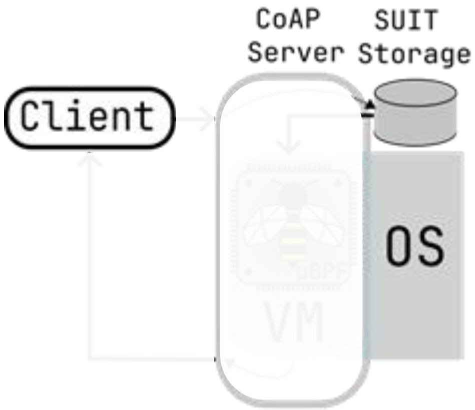
μBPF VM

Middleware

V3 ← Program Storage ← V2 ← Signature Verification

Host OS (RIOT)

Program Execution Modes



Short-lived Programs

Direct Access to Network Packets

Long-running Programs

Evaluated Criteria

Execution time

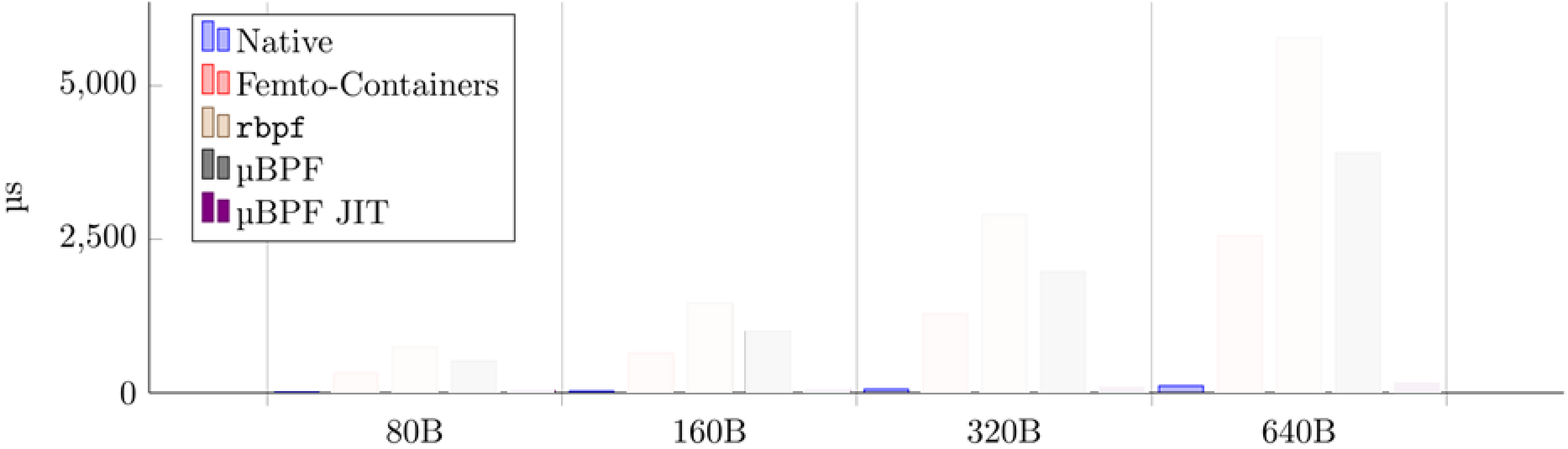
- baselines:
    - native C
    - Femto-Containers VM (current state-of-the-art)
    - default **rbpf** VM implementation

Program binary size

- baselines:
    - eBPF object files
    - Femto-Containers custom patched binaries

Evaluation

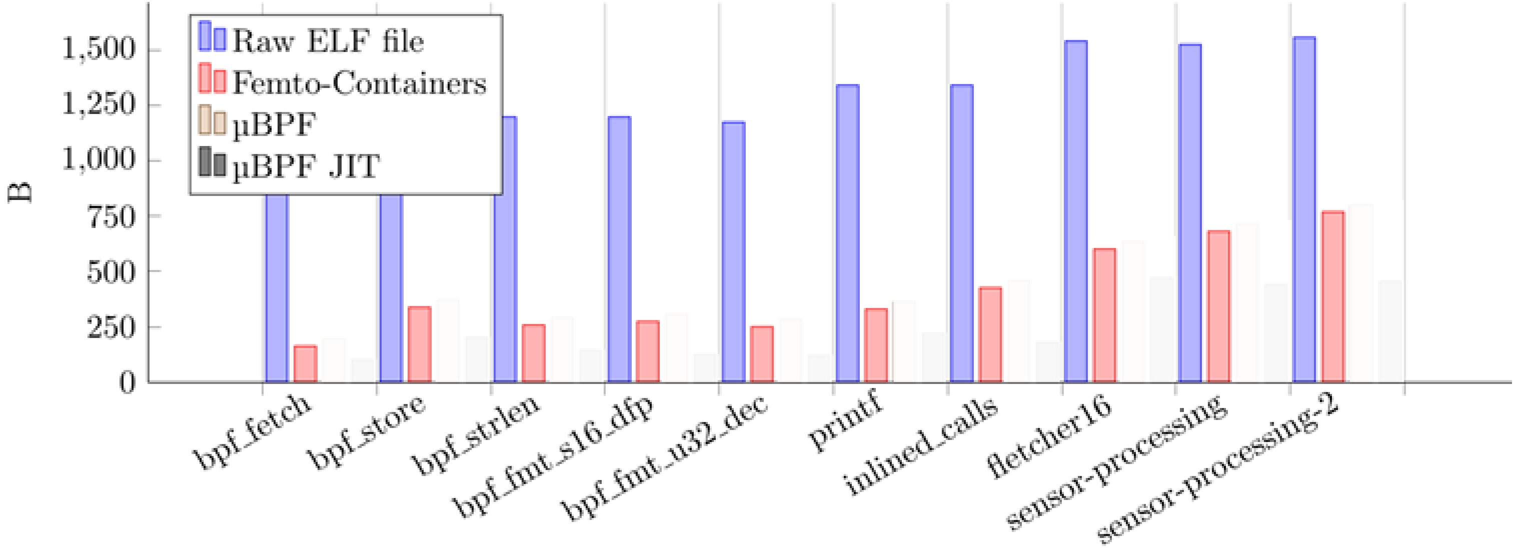
Execution Time: Fletcher-16 Benchmark



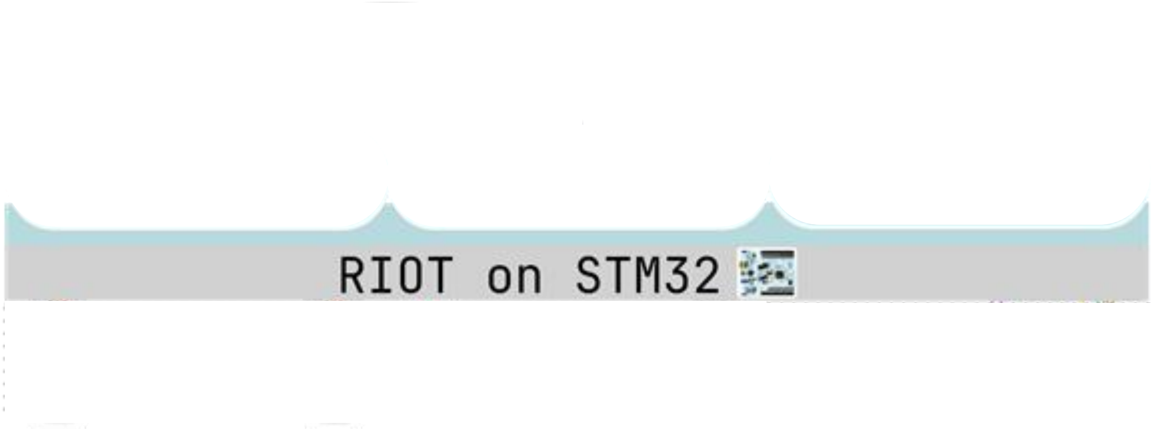
Fletcher16 checksum algorithm execution time

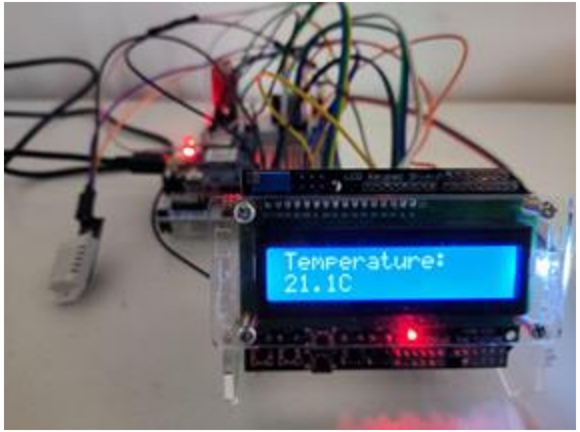
Evaluation

Program Binary Size: Example Programs



Example Application: Weather Sensor Station





µBPF: Using eBPF for Microcontroller Compartmentalization

- compartmentalize embedded device deployments using eBPF VMs
- compartments communicate using eBPF helper functions
- fault isolation and easy redeployment
- JIT compiler achieving native performance and up to 50% program size reduction

SIGCOMM 2024
SYDNEY

eBPF ‘24 Workshop
August 4, 2024

https://github.com/SzymonKubica/micro-bpf

IMPERIAL

Appendix

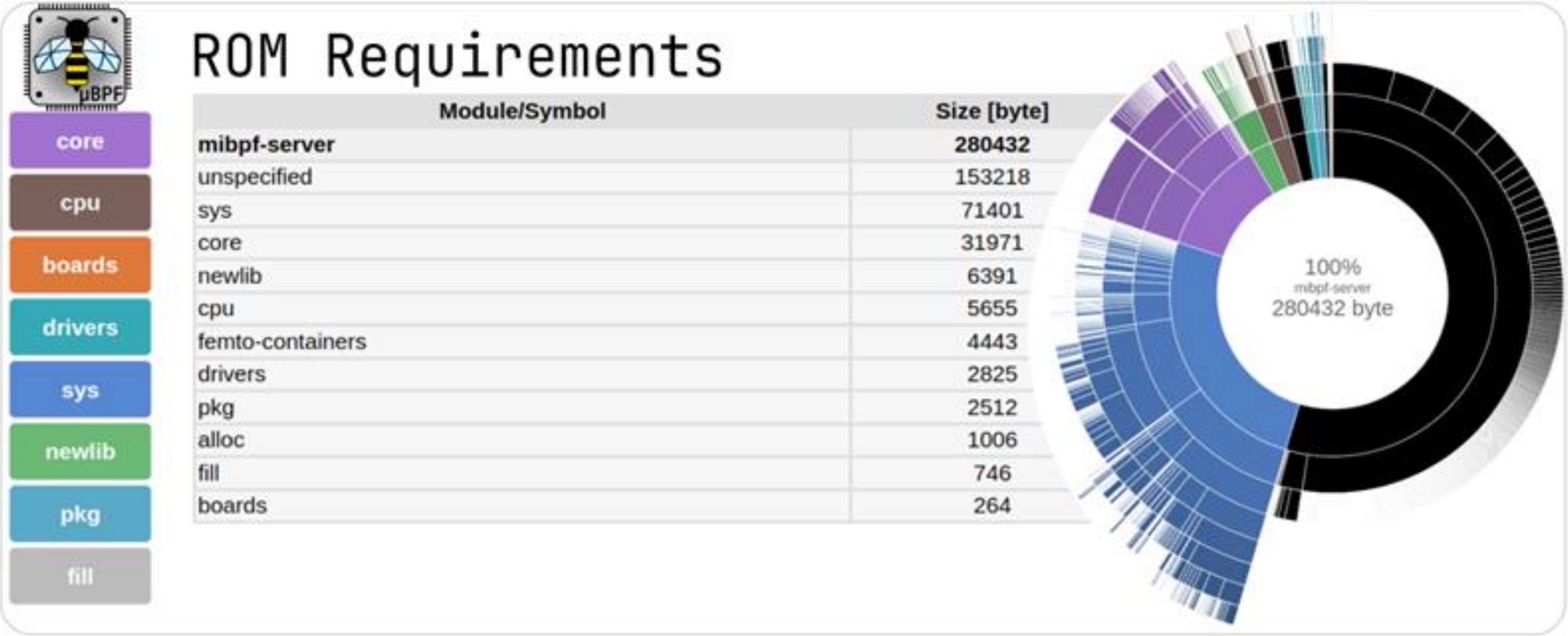
Solution Compatibility

System requirements:

- RAM: 45 KiB required / 256 Kib available
- ROM: 0.28 MiB required / 2 MiB available
- 74% of the boards supported by RIOT are compatible

Appendix

Detailed ROM requirements

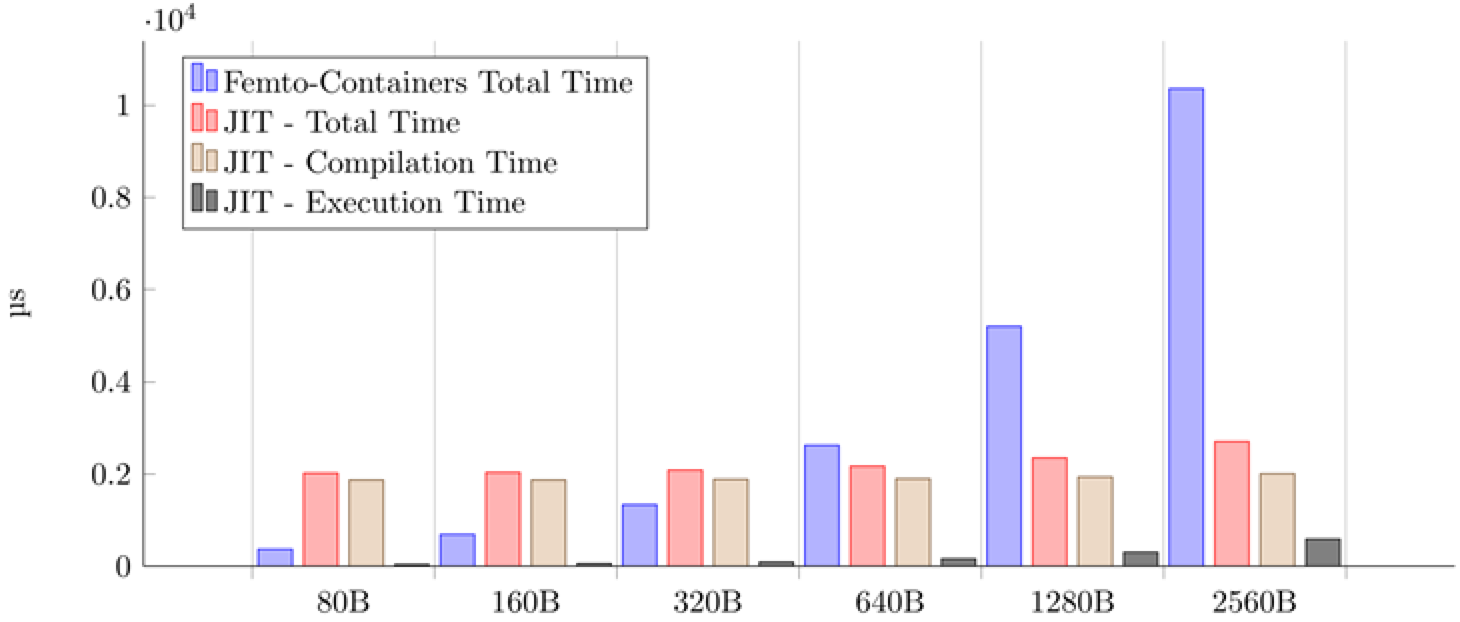


ROM Requirements

Module/Symbol	Size [byte]
mibpf-server	**280432**
unspecified	153218
sys	71401
core	31971
newlib	6391
cpu	5655
femto-containers	4443
drivers	2825
pkg	2512
alloc	1006
fill	746
boards	264

Appendix

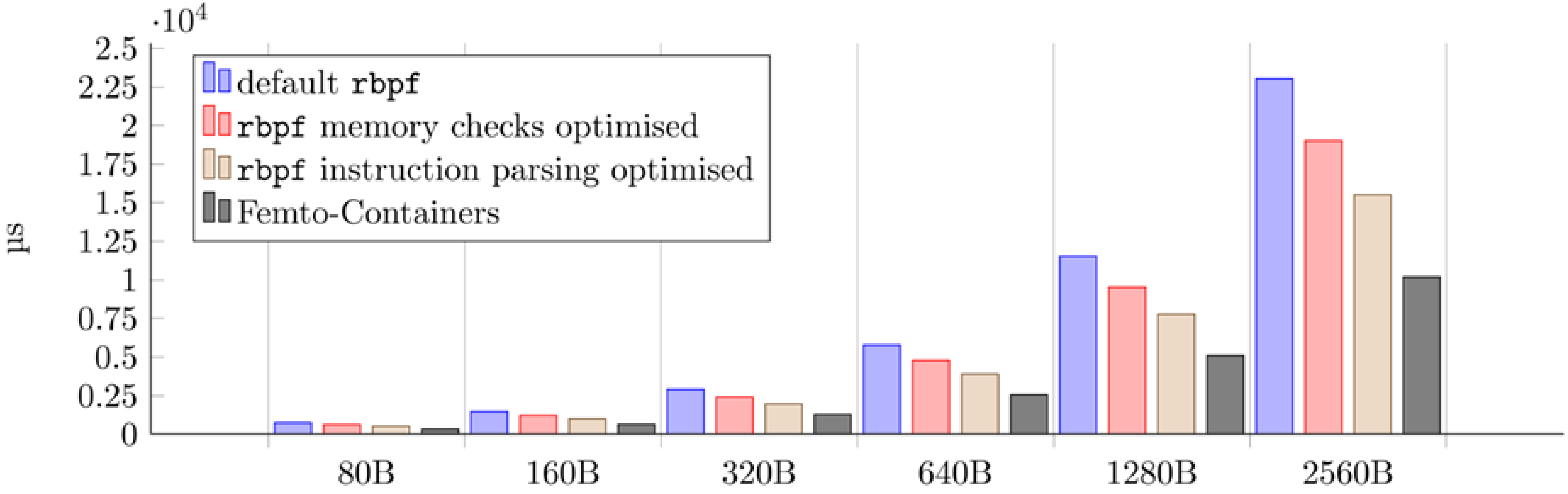
JIT no amortisation investigation



Fletcher16 checksum execution time for varying string sizes

Appendix

Optimisations added to **rbpf**



Fletcher16 checksum execution time for varying string sizes