From Browser to Kernel:
Exploring a Lightweight Sandboxed Approach for Unikernel Extensions

Martin Kröning, Jonathan Klimt, Stefan Lankes, Antonello Monti

PLOS 2025

E.ON Energy Research Center | RWTH AACHEN UNIVERSITY

What? Why?

- Unikernels (commonly) are
  - single-process,
  - single-address-space, and
  - single-privilege-level VM images.

What? Why?

- Unikernels (commonly) are
  - single-process,
  - single-address-space, and
  - single-privilege-level VM images.
- No hardware privilege separation

What? Why?

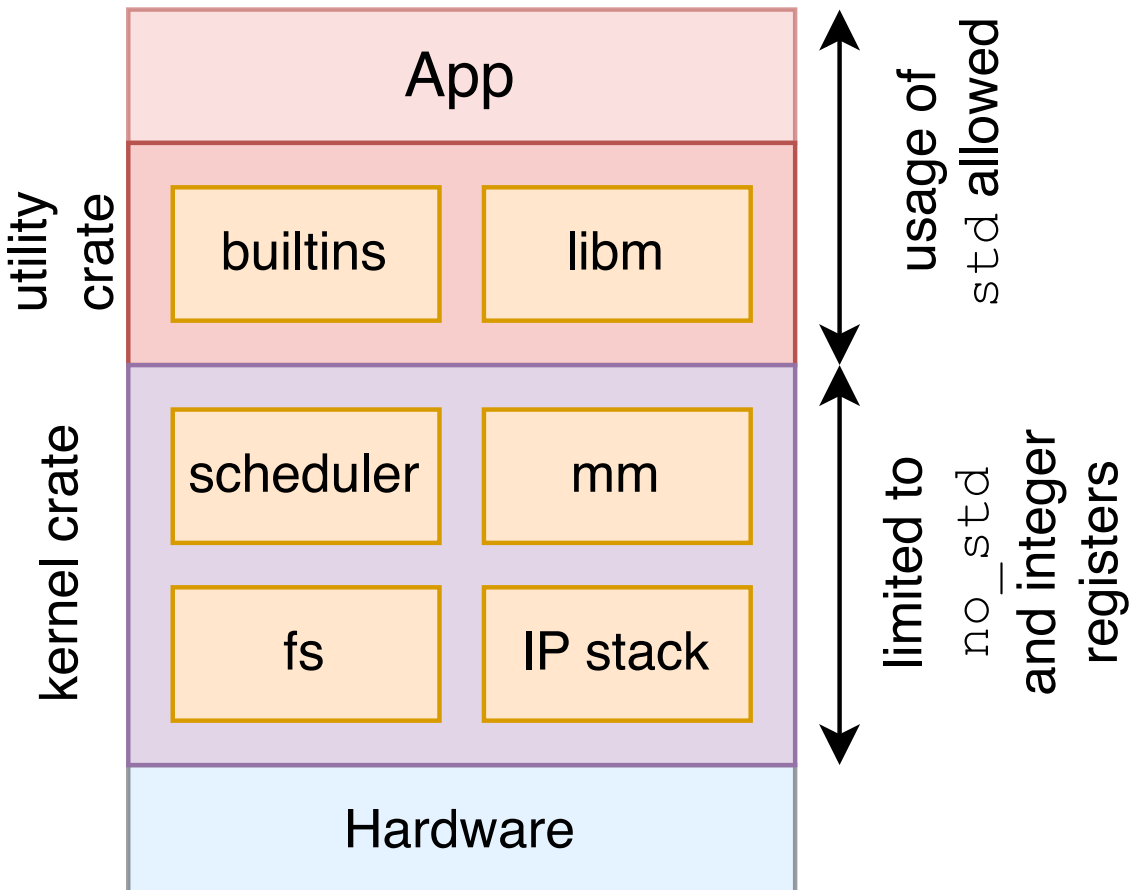
- Unikernels (commonly) are
  - single-process,
  - single-address-space, and
  - single-privilege-level VM images.
- No hardware privilege separation

$\Longrightarrow$ Let's use Webassembly in a sandbox using Wasmtime

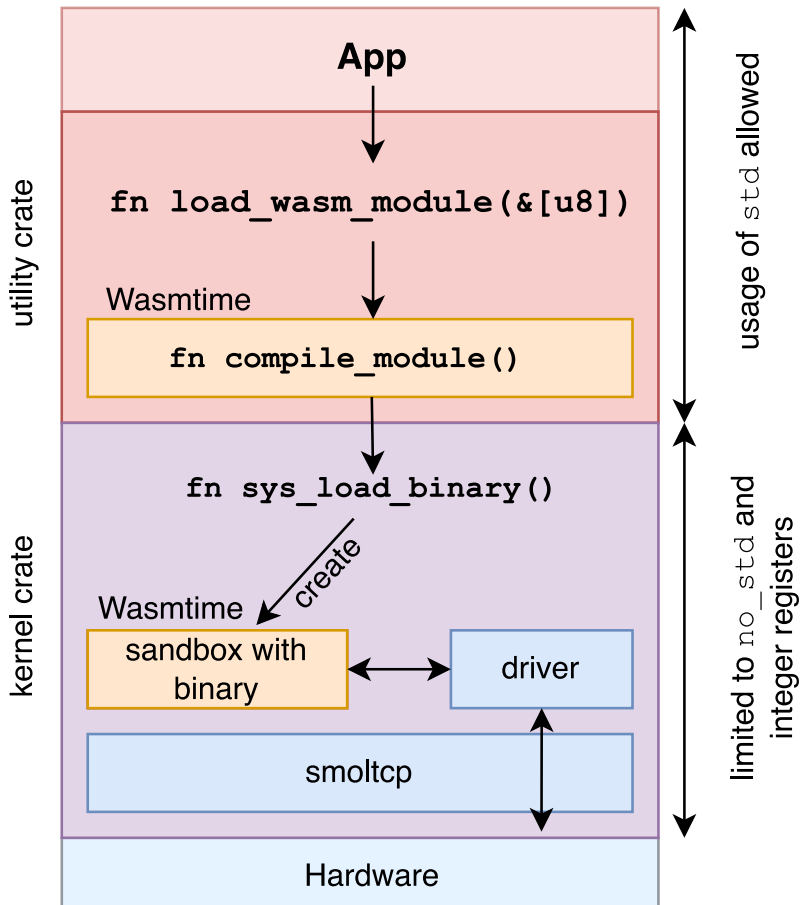
What? Why?

- Extending OS kernels at runtime is great
  - Code injection
  - Network packet tracing
  - Custom drivers
- The Go-to solution: eBPF
  - 👍 Static code analysis by in-kernel verifier
  - 👎 Code has to pass in-kernel verifier
    - = Cannot easily run arbitrary programs

Design of the Hermit libOS



Loading a Wasm Module



Connecting Wasm Space and Kernel Space

```
let mut linker = Linker::new(...);
linker
    .func_wrap("env", "now", || now())
    .unwrap();
```

Connecting Wasm Space and Kernel Space

```
static PCAP_FILE: Mutex<File> = ...;

#[no_mangle]
extern "C" fn handle_pcap() {
    let mut buffer = vec![0; ETHR_SIZE];
    let n = io::stdin().read(&mut buffer)
        .unwrap();

    PcapSink::packet(
        &mut *PCAP_FILE.lock(),
        Instant::ZERO,
        &buffer[..n],
    );
}
```

Key Performance Characteristics of the Wasm Sandbox

- Test platform:
  - AMD EPYC 9354P 32-core processor and 128 GB RAM
  - Hyperthreading was disabled in the BIOS

Benchmark	Result

Key Performance Characteristics of the Wasm Sandbox

- Test platform:
  - AMD EPYC 9354P 32-core processor and 128 GB RAM
  - Hyperthreading was disabled in the BIOS

Benchmark	Result
Compile a small Wasm module	940 µs

Key Performance Characteristics of the Wasm Sandbox

- Test platform:
  - AMD EPYC 9354P 32-core processor and 128 GB RAM
  - Hyperthreading was disabled in the BIOS

Benchmark	Result
Compile a small Wasm module	940 µs
Instantiate a new sandbox	125 µs

Key Performance Characteristics of the Wasm Sandbox

- Test platform:
  - AMD EPYC 9354P 32-core processor and 128 GB RAM
  - Hyperthreading was disabled in the BIOS

Benchmark	Result
Compile a small Wasm module	940 µs
Instantiate a new sandbox	125 µs
Call Wasm NOP function from kernel	127 ns

Key Performance Characteristics of the Wasm Sandbox

- Test platform:
  - AMD EPYC 9354P 32-core processor and 128 GB RAM
  - Hyperthreading was disabled in the BIOS

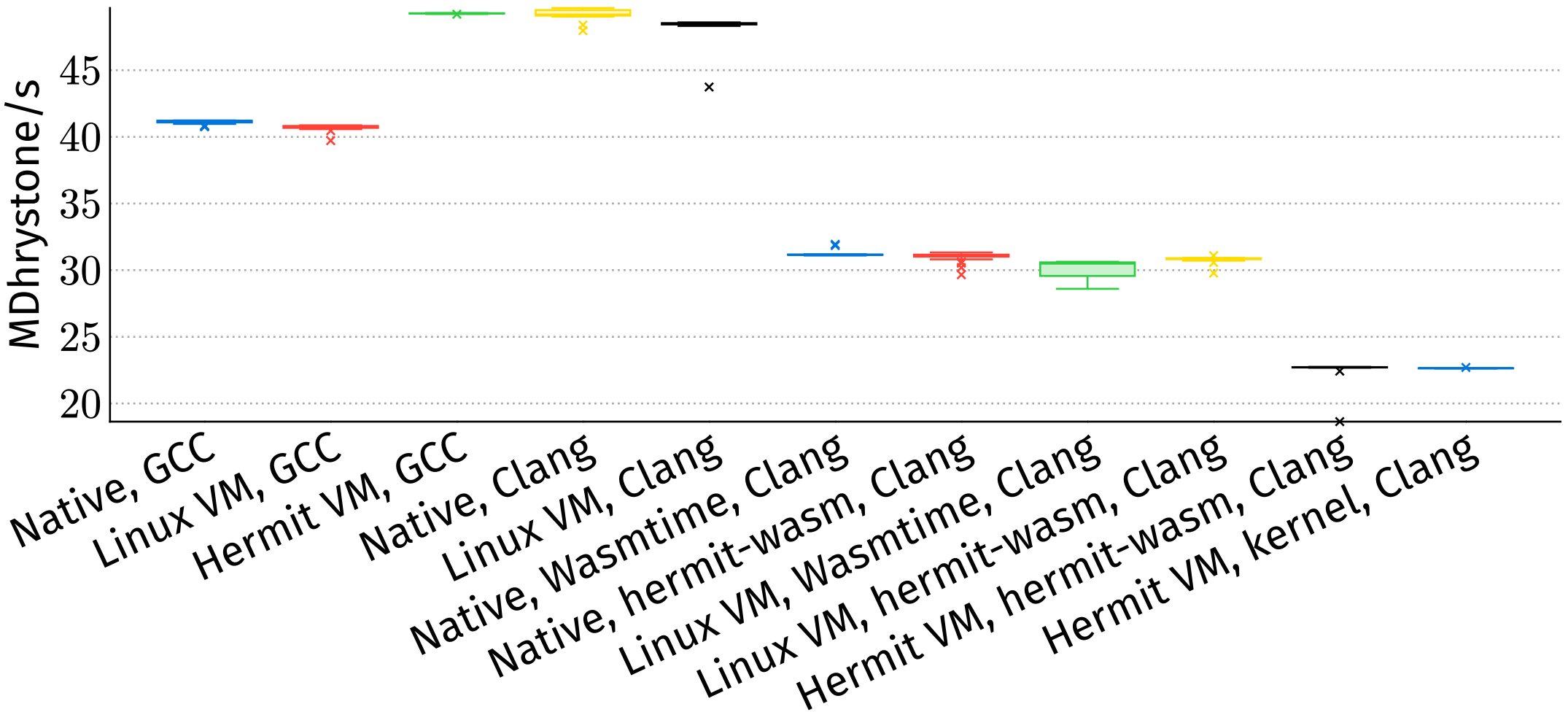
Benchmark	Result
Compile a small Wasm module	940 µs
Instantiate a new sandbox	125 µs
Call Wasm NOP function from kernel	127 ns
Call native NOP function from kernel	0.26 ns

Dhrystone

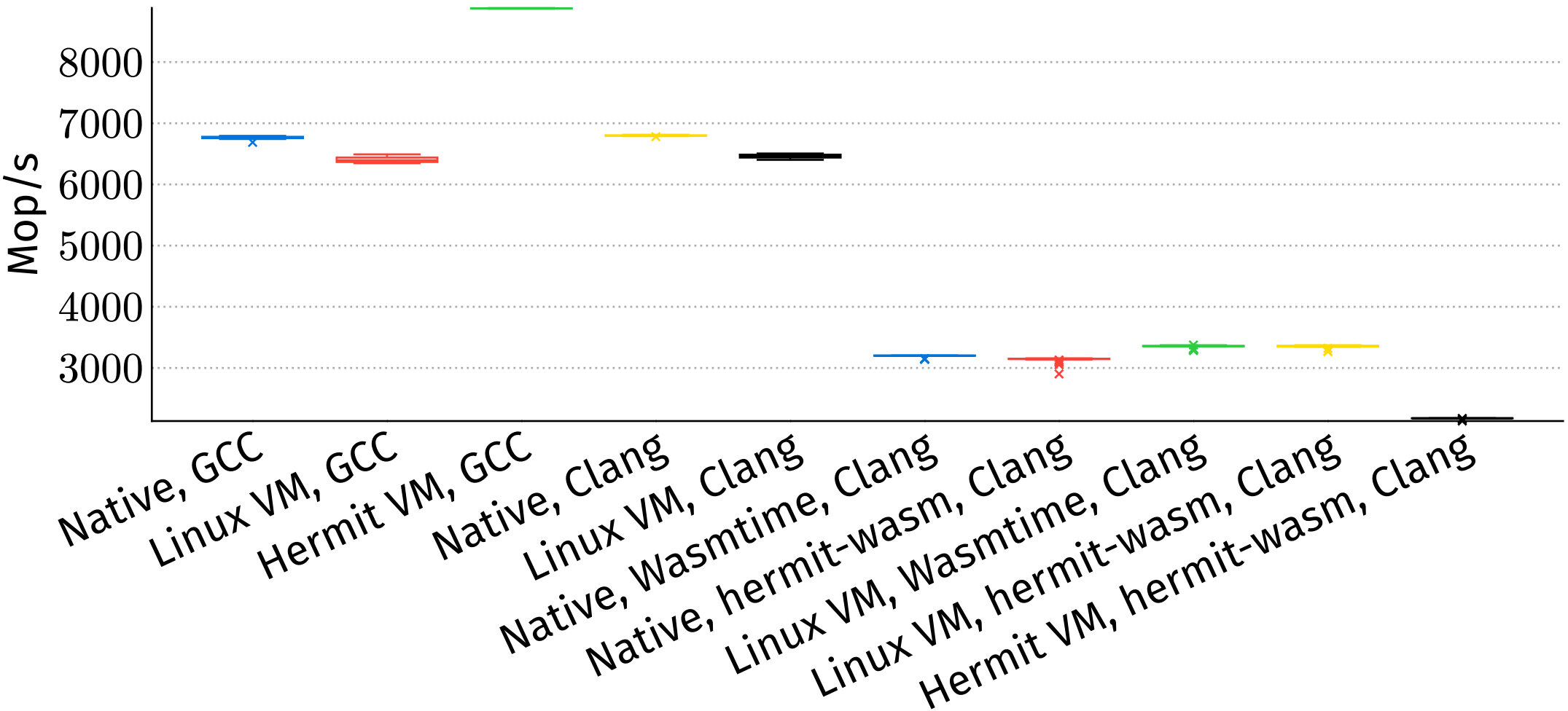
- Integer arithmetic benchmark form 1988
- Is able to run in kernel and user space
- Wasmtime checks missuage of C functions ⟹ Runtime error

```
@@ -384,7 +384,7 @@
 #define CLOCK_TYPE "time()"
 #undef HZ
 #define HZ     (1) /* time() returns time in seconds */
-extern long     time(); /* see library function "time"  */
+extern long long    time(); /* see library function "time"  */
 #define Too_Small_Time 2 /* Measurements should last at least 2 seconds */
 #define Start_Timer() Begin_Time = time ( (long *) 0)
 #define Stop_Timer()  End_Time   = time ( (long *) 0)
```

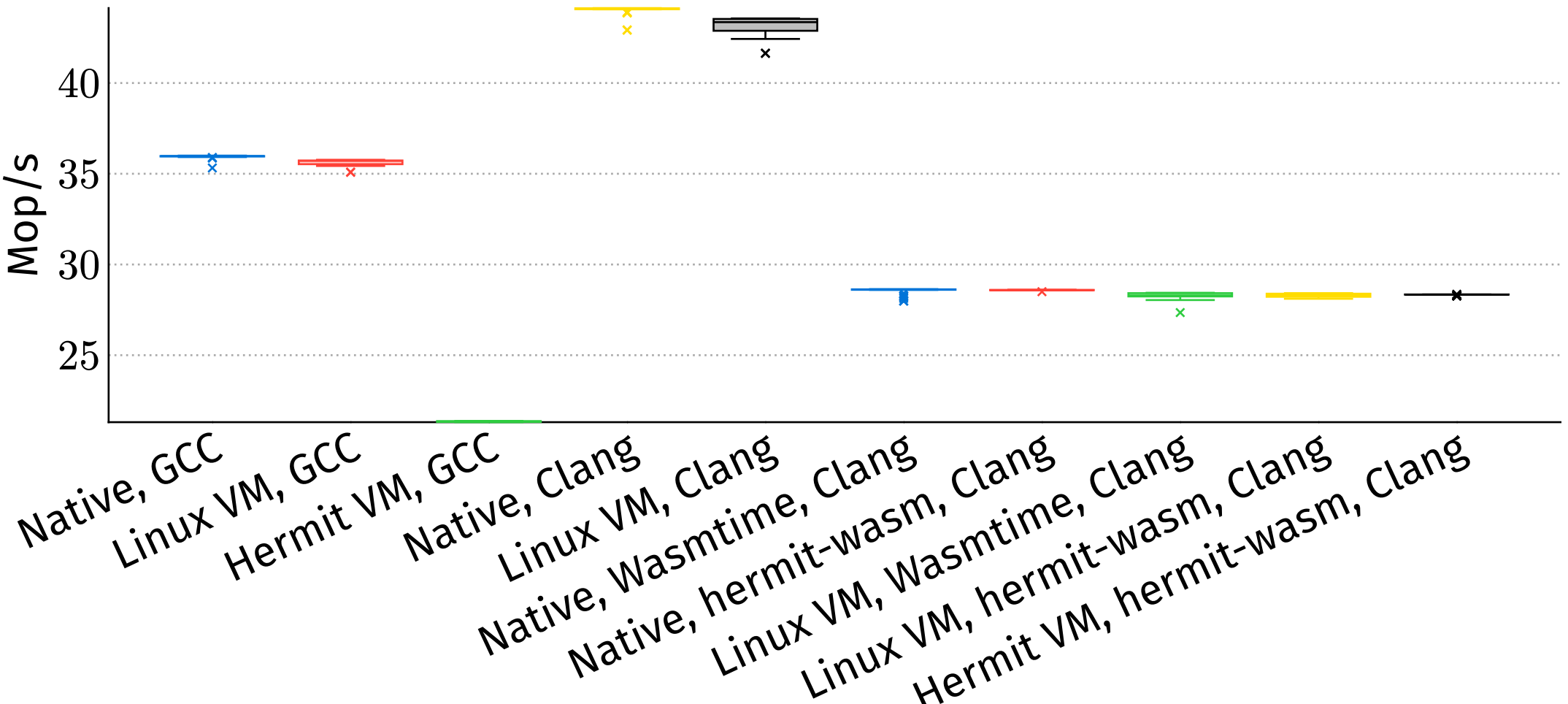
Dhrystone


MDhrystone/s
45
40
35
30
25
20
Native, GCC
Linux VM, GCC
Hermit VM, GCC
Native, Clang
Linux VM, Clang
Native, Wasmtime, Clang
Native, hermit-wasm, Clang
Linux VM, Wasmtime, Clang
Linux VM, hermit-wasm, Clang
Hermit VM, hermit-wasm, Clang
Hermit VM, kernel, Clang

SNU-NPB MG


Mop/s
8000
7000
6000
5000
4000
3000
Native, GCC
Linux VM, GCC
Hermit VM, GCC
Native, Clang
Linux VM, Clang
Native, Wasmtime, Clang
Native, hermit-wasm, Clang
Linux VM, Wasmtime, Clang
Linux VM, hermit-wasm, Clang
Hermit VM, hermit-wasm, Clang

SNU-NPB EP


Mop/s
40
35
30
25
Native, GCC
Linux VM, GCC
Hermit VM, GCC
Native, Clang
Linux VM, Clang
Native, Wasmtime, Clang
Native, hermit-wasm, Clang
Linux VM, Wasmtime, Clang
Linux VM, hermit-wasm, Clang
Hermit VM, hermit-wasm, Clang

Summary

Summary

Status quo

- OS extensions with eBPF
  - have to be verified statically,
  - cannot be arbitrary user space code, and
  - are the sensible choice for multi-process OSes.

Summary

Status quo

- OS extensions with eBPF
  - have to be verified statically,
  - cannot be arbitrary user space code, and
  - are the sensible choice for multi-process OSes.

Our work

- OS extensions with WebAssembly
  - cannot be verified statically,
  - can be (mostly) arbitrary user space code,
  - are the flexible choice for single-process OSes.

Summary

Status quo

- OS extensions with eBPF
  - have to be verified statically,
  - cannot be arbitrary user space code, and
  - are the sensible choice for multi-process OSes.

Our work

- OS extensions with WebAssembly
  - cannot be verified statically,
  - can be (mostly) arbitrary user space code,
  - are the flexible choice for single-process OSes.
- Performance is not terrible! (thanks to Wasmtime)

Conclusion

Future Work

Conclusion

Future Work

- Investigate Hermit-specific performance overhead

Conclusion

Future Work

- Investigate Hermit-specific performance overhead
- Explore more advanced extensions
  - Device drivers
  - Standard hooks

Conclusion

Future Work

- Investigate Hermit-specific performance overhead
- Explore more advanced extensions
  - Device drivers
  - Standard hooks

Thanks for listening! 😊

Acknowledgments

Funded by the European Union

This work was funded by the European Union's Horizon Europe projects NEMO (101070118) and CyberNEMO (101168182).

Thank you for your kind attention!

Martin Kröning, Jonathan Klimt, Stefan Lankes, Antonello Monti – slankes@eonerc.rwth-aachen.de

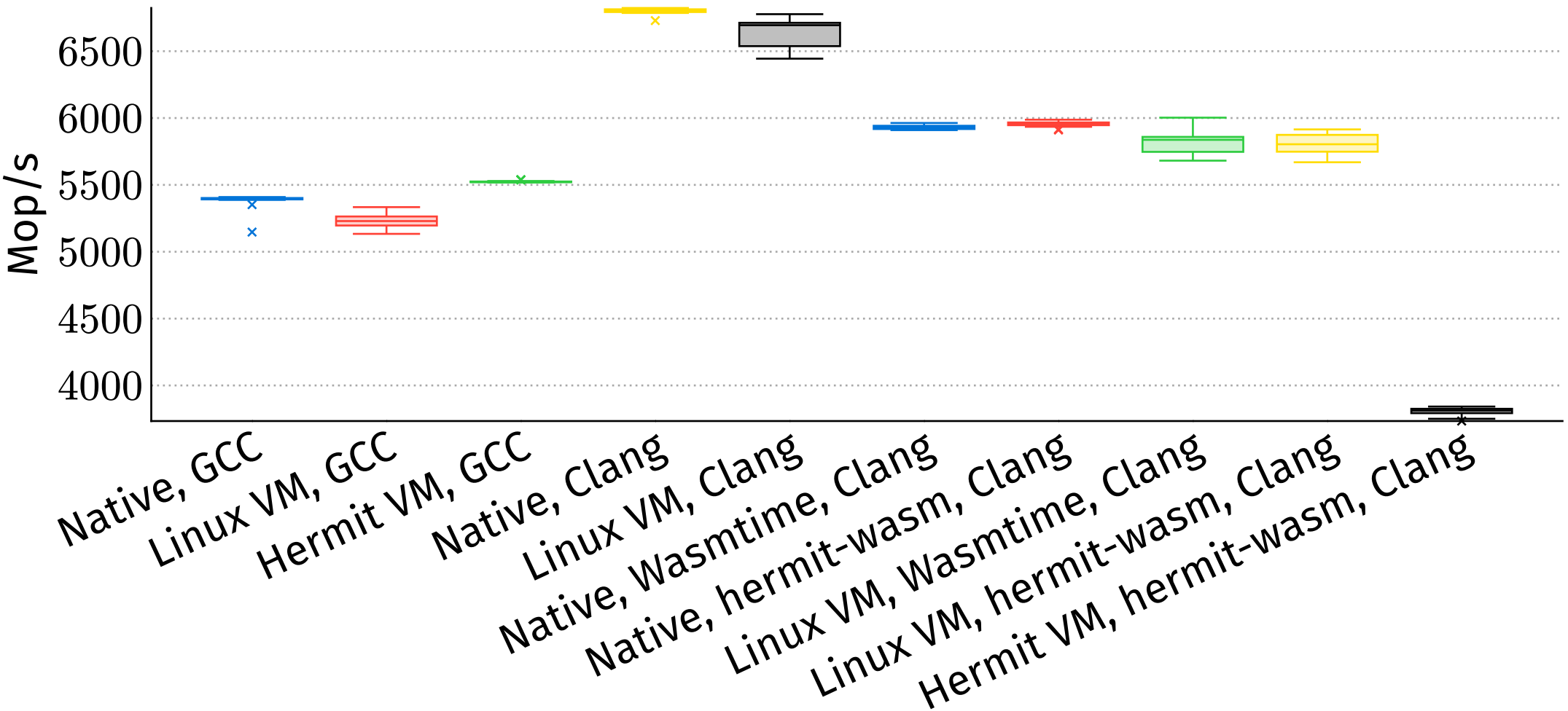
Institute for Automation of Complex Power Systems
E.ON Energy Research Center, RWTH Aachen University
Mathieustraße 10
52074 Aachen

www.acs.eonerc.rwth-aachen.de

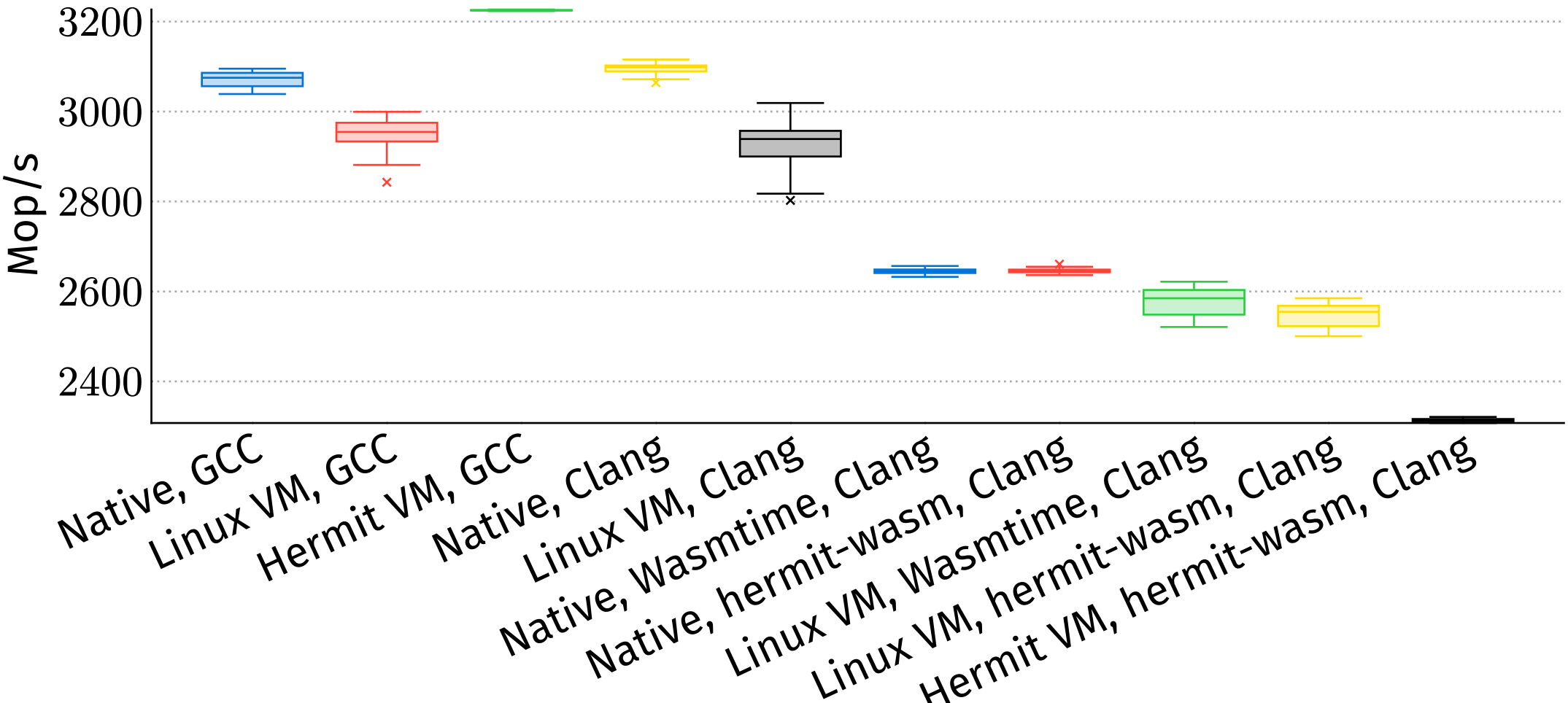
E.ON Energy Research Center | RWTH AACHEN UNIVERSITY

Backup Slides

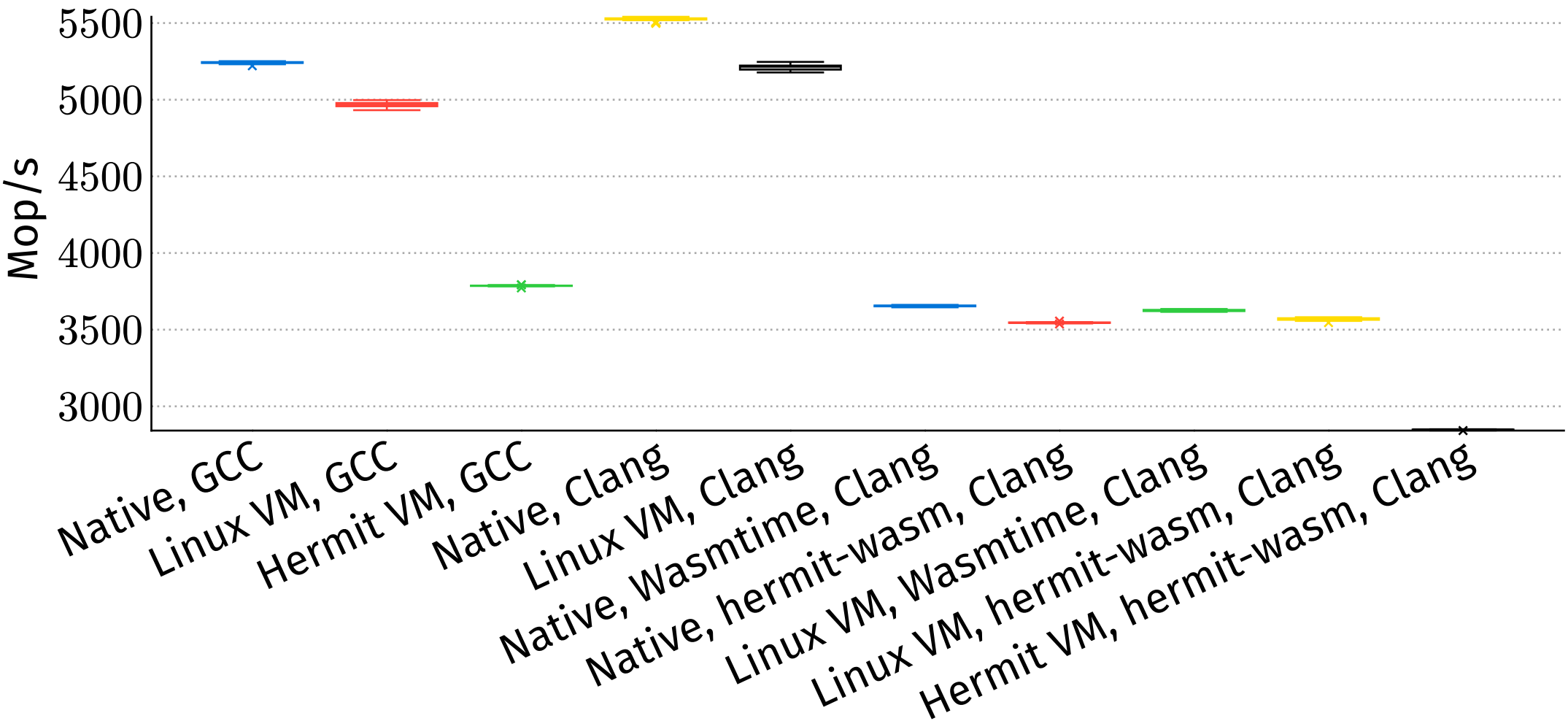
SNU-NPB BT


Mop/s
6500
6000
5500
5000
4500
4000
Native, GCC
Linux VM, GCC
Hermit VM, GCC
Native, Clang
Linux VM, Clang
Native, Wasmtime, Clang
Native, hermit-wasm, Clang
Linux VM, Wasmtime, Clang
Linux VM, hermit-wasm, Clang
Hermit VM, hermit-wasm, Clang

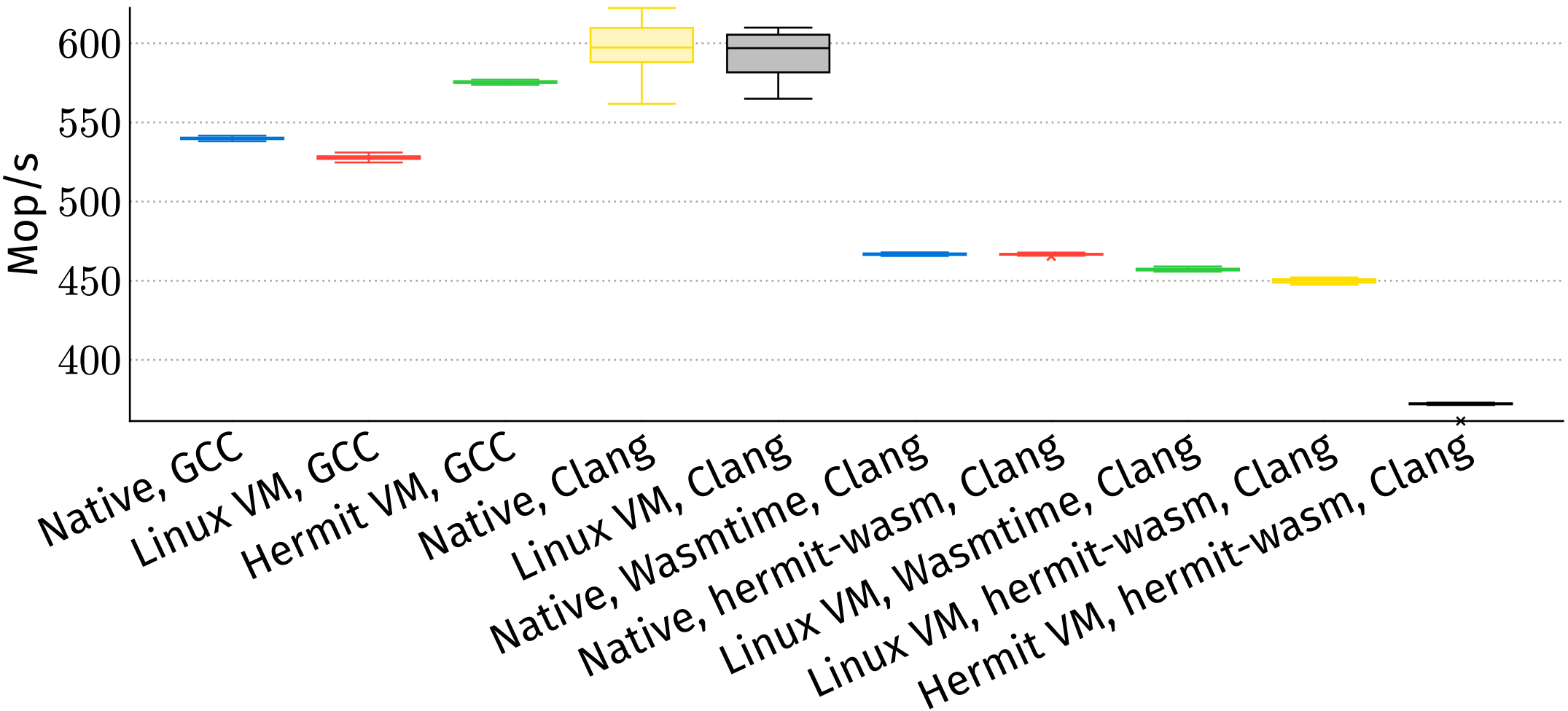
SNU-NPB CG


Mop/s
3200
3000
2800
2600
2400
Native, GCC
Linux VM, GCC
Hermit VM, GCC
Native, Clang
Linux VM, Clang
Native, Wasmtime, Clang
Native, hermit-wasm, Clang
Linux VM, Wasmtime, Clang
Linux VM, hermit-wasm, Clang
Hermit VM, hermit-wasm, Clang

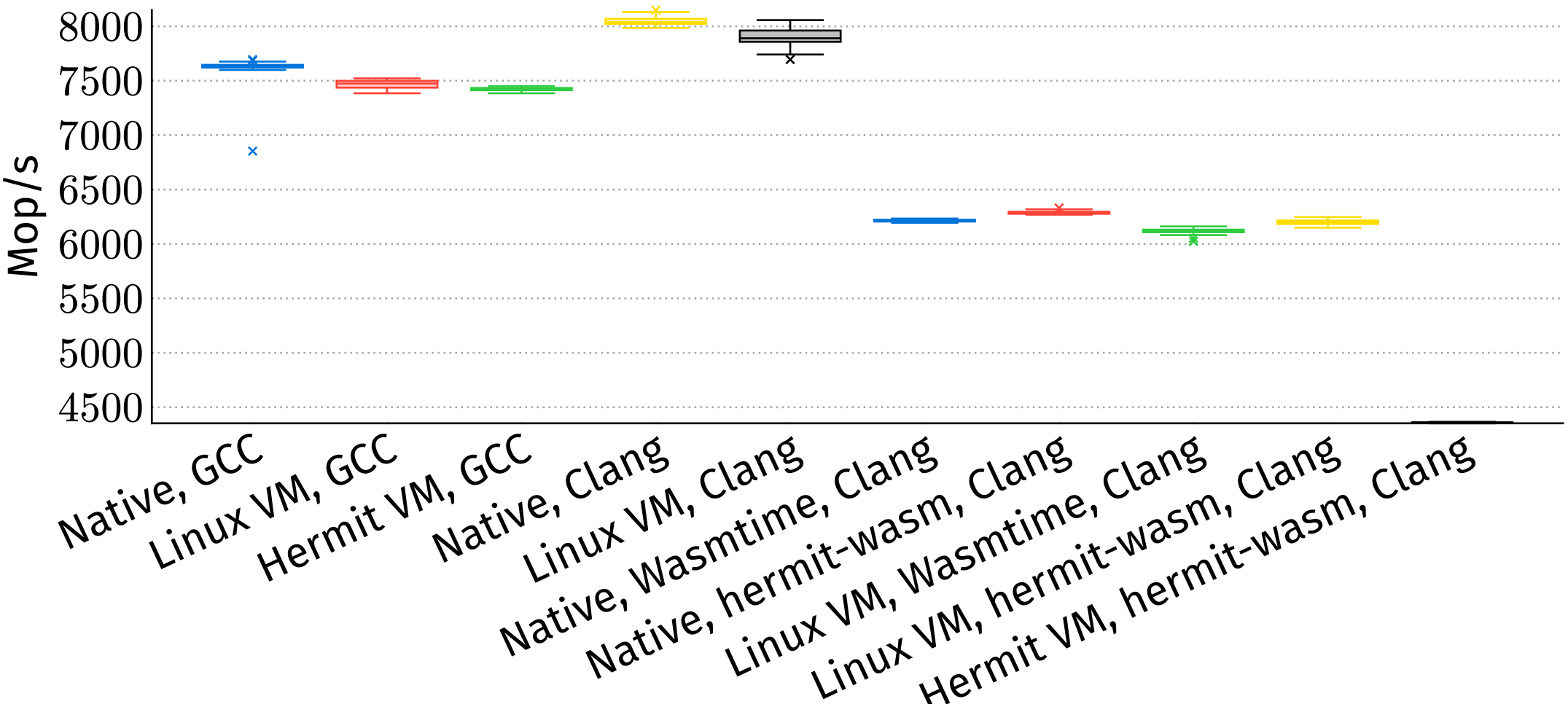
SNU-NPB FT


Mop/s
5500
5000
4500
4000
3500
3000
Native, GCC
Linux VM, GCC
Hermit VM, GCC
Native, Clang
Linux VM, Clang
Native, Wasmtime, Clang
Native, hermit-wasm, Clang
Linux VM, Wasmtime, Clang
Linux VM, hermit-wasm, Clang
Hermit VM, hermit-wasm, Clang

SNU-NPB IS


Mop/s
600
550
500
450
400
Native, GCC
Linux VM, GCC
Hermit VM, GCC
Native, Clang
Linux VM, Clang
Native, Wasmtime, Clang
Native, hermit-wasm, Clang
Linux VM, Wasmtime, Clang
Linux VM, hermit-wasm, Clang
Hermit VM, hermit-wasm, Clang

SNU-NPB SP


Mop/s
8000
7500
7000
6500
6000
5500
5000
4500
Native, GCC
Linux VM, GCC
Hermit VM, GCC
Native, Clang
Linux VM, Clang
Native, Wasmtime, Clang
Native, hermit-wasm, Clang
Linux VM, Wasmtime, Clang
Linux VM, hermit-wasm, Clang
Hermit VM, hermit-wasm, Clang