Thesis Proposal: Performance and Microarchitectural Evaluation of the CHERI Architecture

Martin Fink

October 2024

Details

Type:	MA/BA/GR
Timeframe:	Wintersemester 2024
Keywords:	CHERI, Evaluation, Microarchitectural Analysis, Morello

1 Introduction

Despite ongoing efforts, memory vulnerabilities remain a relevant topic. Several studies have shown that in large software projects, memory safety bugs make up between 70 % and 75 % of their high-impact security vulnerabilities [1, 8, 9]. Software-based approaches either require modifications to source code or incur high performance overheads [2, 4–7]. The CHERI (Capability Hardware Enhanced RISC Instructions) architecture implements memory safety primitives at the hardware-level by extending 64-bit pointers to 128+1 bits, including a validity bit, bounds, and permissions. On memory accesses, the hardware checks the bounds and permissions, promising better performance compared to existing software-based solutions. Still, this protection comes at a performance cost.

2 Objective of the Thesis

In this thesis/guided research, you will perform a detailed microarchitectural analysis of Arm's Morello [3] board, which implements the CHERI architecture. You will implement a framework to measure and investigate several performance and architectural metrics, including:

- Instruction cycles and latencies of CHERI instructions.
- Microarchitectural details such as the cache and pipeline behavior when handling capabilities.

- The cost of performing CHERI access checks on memory operations.
- Potential trade-offs between security benefits and performance overhead.

References

[1] Memory safety. Accessed on March 14, 2024.

[2] Periklis Akritidis, Manuel Costa, Miguel Castro, and Steven Hand. Baggy bounds checking: An efficient and backwards-compatible defense against out-of-bounds errors. In *USENIX Security Symposium*, volume 10, page 96, 2009.

[3] Richard Grisenthwaite, Graeme Barnes, Robert N. M. Watson, Simon W. Moore, Peter Sewell, and Jonathan Woodruff. The Arm Morello Evaluation Platform—Validating CHERI-Based Security in a High-Performance System. 43(3):50–57.

[4] Trevor Jim, J Gregory Morrisett, Dan Grossman, Michael W Hicks, James Cheney, and Yanling Wang. Cyclone: a safe dialect of c. In *USENIX Annual Technical Conference, General Track*, pages 275–288, 2002.

[5] Santosh Nagarakatte, Jianzhou Zhao, Milo MK Martin, and Steve Zdancewic. Softbound: Highly compatible and complete spatial memory safety for c. In *Proceedings of the 30th ACM SIGPLAN Conference on Programming Language Design and Implementation*, pages 245–258, 2009.

[6] George C Necula, Scott McPeak, and Westley Weimer. Ccured: Type-safe retrofitting of legacy code. In *Proceedings of the 29th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, pages 128–139, 2002.

[7] Konstantin Serebryany, Derek Bruening, Alexander Potapenko, and Dmitriy Vyukov. Addresssanitizer: A fast address sanity checker. In *2012 USENIX annual technical conference (USENIX ATC 12)*, pages 309–318, 2012.

[8] Gavin Thomas. A proactive approach to more secure code. Accessed on March 14, 2024.

[9] Jeff Vander Stoep and Chong Zhang. Queue the hardening enhancements. Accessed on March 14, 2024.