Exposé for master thesis at the chair of computer engineering of the University of Hagen

Performance Evaluation of Virtualized Unikernels on RISC-V with Hardware-Assisted Virtualization

Stefan Butz

q7538820

stefan.butz@studium.fernuni-hagen.de

Applied Computer Sciences

March 16th, 2026 - September 16th, 2026

Supervisor: Prof. Dr. Lena Oden

1 Problem

1.1 Motivation

This thesis addresses the lack of comprehensive performance evaluations of virtualized unikernels on RISC-V hardware with hypervisor extension support. While RISC-V is gaining adoption as an open instruction set architecture and development boards with hardware virtualization support have recently become available [1], [2], performance characteristics of virtualized unikernels on such hardware remain largely unexplored.

Existing evaluations of unikernels on RISC-V focus primarily on RustyHermit. Schöning evaluates RustyHermit on both bare metal and Quick Emulator (QEMU) [3], while Ruppert-Maas evaluates it on QEMU emulation only [4]. However, neither work evaluates virtualized deployments utilizing hardware hypervisor extensions. Emulation may not accurately reflect real hardware behavior, particularly regarding interrupt handling, memory management, and virtualization overhead [5].

Furthermore, prior studies employ synthetic microbenchmarks rather than realistic application workloads [3], [4]. This limits the applicability of results to practical scenarios.

Unikernels are proposed as lightweight virtualization solutions with potential benefits including reduced boot times, lower memory consumption, and simplified system stacks [6]. Evaluating whether these benefits persist on RISC-V systems with hardware virtualization support is important for systems research.

1.2 Formulation

The objective of this thesis is to evaluate the performance characteristics of virtualized unikernels on RISC-V hardware supporting the hypervisor extension. The evaluation focuses on the RustyHermit unikernel running under Kernel-based Virtual Machine (KVM) and examines execution performance, boot latency, i/o behavior, network performance, and resource usage using a combination of synthetic benchmarks and representative workloads.

To assess the validity of emulation-based studies, results obtained on real hardware are compared with QEMU-based virtualization. Additionally, RustyHermit is compared against a minimal virtualized Linux system to contextualize observed performance characteristics. Where feasible, results are contrasted with other RISC-V–capable unikernels.

Kernel modifications are limited to instrumentation and targeted adjustments necessary to enable accurate measurement and analysis. Functional extensions and general feature development are not primary objectives of this work.

1.3 Intended Results

The thesis is expected to produce a reproducible experimental setup for evaluating virtualized unikernels on RISC-V hardware with hypervisor extension support. It provides quantitative performance measurements of the RustyHermit unikernel running under KVM on real hardware, covering execution performance, boot latency, i/o and network behavior, and resource usage. Based on controlled benchmarking and kernel-level instrumentation, the work analyzes virtualization-related overheads specific to the RISC-V hypervisor extension and compares hardware-based results with QEMU-based virtualization to assess the validity of emulated evaluations. Where feasible, the results are contextualized through comparison with a minimal virtualized Linux system and other RISC-V-capable unikernels. In addition, the thesis produces software artifacts supporting the evaluation, including kernel-level instrumentation and limited, targeted adjustments to the virtualization and i/o stack required to enable reproducible and meaningful experiments.

2 Current State of the Art

Unikernels have recently received increasing attention as lightweight virtualization solutions, and initial efforts have explored their applicability on the RISC-V architecture. Existing work has demonstrated the feasibility of running unikernels such as RustyHermit on RISC-V systems, including execution on bare-metal hardware as well as in emulated or Field-Programmable Gate Array (FPGA)-based environments [3], [4]. However, these studies do not evaluate virtualized deployments making use of hardware-supported hypervisor extensions, despite virtualization being a primary deployment scenario for many unikernel-based systems.

Furthermore, current evaluations are predominantly conducted in emulation environments such as QEMU, which may not accurately reflect the performance characteristics of real hardware, particularly with respect to interrupt handling, memory management, and virtualization overhead [5]. In addition, prior work primarily relies on synthetic microbenchmarks rather than workloads resembling realistic application scenarios. While such benchmarks are useful for isolating specific system properties, their results may not generalize to practical deployments if not carefully designed [7].

In contrast, unikernels have been evaluated extensively on established architectures such as x86 and ARM, including in virtualized environments using hardware-assisted virtualization [6], [8], [9]. However, performance characteristics of software systems depend on both the instruction set architecture and the underlying microarchitectural implementation, and therefore cannot be assumed to transfer across platforms [10]

Independent of unikernel research, several studies have investigated virtualization support for RISC-V itself, including hardware implementations of the hypervisor extension [11], simulator-level integration [12], and its use for memory isolation mechanisms [13]. While these works provide insights into architectural mechanisms and implementation aspects of the RISC-V hypervisor extension, their evaluations focus on hardware behavior, microarchitectural simulation, or security use cases rather than the performance of higher-level software systems such as unikernels in realistic deployment scenarios.

3 Approach

The proposed approach consists of a systematic experimental evaluation of virtualized unikernels on RISC-V hardware supporting the hypervisor extension. The RustyHermit unikernel running under KVM serves as the primary evaluation platform. Its performance is compared with a minimal virtualized Linux system and corresponding deployments in QEMU-based virtualization environments in order to assess the representativeness of emulation-based evaluations.

Relevant performance metrics, including execution performance, interrupt handling latency, and virtualization overhead, are defined and measured using instrumentation at hypervisor, kernel, and user-space levels. Existing tracing tools (e.g. perf, rftrace or KVM Virtual Machine Introspection (KVM-VMI)) are employed to enable quantitative performance analysis. Controlled experiment design is applied to isolate performance-relevant variables, including the selection of appropriate timing sources, mitigation of measurement noise through repeated measurements and core pinning, and separation of system warm-up and steady-state execution phases.

Reproducible synthetic benchmarks for cpu, memory, i/o, and network performance are complemented by representative application workloads such as web server or database scenarios. Where necessary, limited, targeted adjustments to the hypervisor or kernel are introduced to enable reproducible and meaningful experiments. Measurement results are comparatively analyzed across different deployment scenarios (e.g., real hardware vs. QEMU, unikernel vs. Linux) in order to assess performance characteristics under

virtualization. Reproducibility across repeated experiments and baseline comparisons with a minimal virtualized Linux system are used to support the interpretation of observed results.

Potential limitations arise from restricted hardware scalability (e.g., limited core count), which constrains the evaluation of parallelism and scalability. Differences between currently available boards supporting the RISC-V hypervisor extension draft version 0.6 [1] and future implementations of the ratified version 1.0 [14] may limit the transferability of results to subsequent hardware platforms.

4 Preliminary Outline

Abstract

Table of Contents

1. Introduction
   1. Motivation
   2. Problem Statement
   3. Objectives
2. Fundamentals
   1. Unikernel Architecture [6], [15], [16], [17], [18], [19], [20]
      - General OS architecture concepts
      - Unikernel design principles
      - RustyHermit overview
   2. Virtualization Concepts [15], [21], [22], [23], [24], [25], [26]
      - Process vs System VMs
      - Virtualization vs Emulation
      - VM exits and traps
      - Interrupt virtualization
      - Address translation and memory management
      - Hardware-assisted virtualization
      - Paravirtualization, Virtio
      - QEMU / KVM overview
3. Architectural Background [1], [11], [12], [13], [14], [27]
   1. RISC-V Architecture
      - Privilege levels (M, S, U)
      - Trap handling
      - Interrupt delegation
      - Supervisor Binary Interface (SBI)
      - Single Stage Address Translation
   2. RISC-V Hypervisor Extension
      - Hypervisor execution modes (HS, VS)
      - Guest context management
      - Two-stage address translation

- Virtual interrupt handling
- VM entry and exit mechanisms
- Timer virtualization
- Differences draft v0.6 vs ratified v1.0
- Availability of hardware implementations

4. Evaluation Methodology [7], [28], [29]
   1. Evaluation objectives
      - Performance characteristics to be measured: execution performance, interrupt latency, i/o performance, network performance, resource usage, virtualization overhead
   2. Evaluation scenarios
      - RustyHermit on KVM on real hardware
      - Minimal virtualized Linux on KVM on real hardware (baseline)
      - RustyHermit in QEMU-based virtualization
      - Minimal virtualized Linux in QEMU-based virtualization
      - If feasible: other unikernels (e.g., unishyper, nanos) in similar scenarios
   3. Workloads
      - Synthetic benchmarks for cpu, memory, i/o, and network performance
      - Representative application workloads (e.g., web server, database)
   4. Measurement strategy
      - Instrumentation levels: hypervisor, kernel, user-space
      - Tracing tools
      - Timing sources and measurement techniques
   5. Experiment Design
      - Noise reduction techniques: Core pinning, disabling frequency scaling, warm-up vs steady-state performance, repetition and statistical analysis
   6. Data Analysis
      - Comparative analysis across deployment scenarios
      - Aggregation and visualization of results
   7. Validity
      - Reproducibility and consistency checks
   8. Limitations
      - Hardware scalability
      - Hypervisor extension version differences

5. Implementation
   - Software artifacts supporting the evaluation
   - Instrumentation of hypervisor, kernel, and user-space programs
   - Benchmark implementation
   - Hardware and software configuration and deployment details

6. Evaluation Results
   - Execution performance results
   - Interrupt handling latency results
   - I/O performance results

- Network performance results
- Resource usage results
- Virtualization overhead results
- Comparision across deployment scenarios

7. Discussion
   - Interpretation of observed performance characteristics
   - Identification of architecture-specific performance effects and bottlenecks
   - Implications for the design and deployment of unikernels on RISC-V hardware
   - Comparative analysis with baseline system and potential other unikernels
   - Impact of methodical choices on results
   - Relation to existing work
8. Conclusion
   - Summary of findings
   - Relation to objectives and problem statement
9. Outlook
   - Open questions
   - Future work

List of Figures

List of Tables

Acronyms

Bibliography

Appendix

5 Work Plan

Total time: 6 months = 24 weeks

Week 1-4: Literature review completed
- Review of related work
- Refinement of problem statement and objectives
- Draft of chapter 1 (Introduction)
- Outline for chapter 2 (Fundamentals)
- Outline for chapter 3 (Architectural Background)

Week 5-8: Evaluation methodology finalized
- Draft of chapter 2 (Fundamentals)
- Draft of chapter 3 (Architectural Background)
- Specification of workloads and benchmarks

Week 9-12: Functional experimental setup
- Setup of experimental environment
- Instrumentation

- Debugging
- Draft of chapter 4 (Evaluation Methodology)

Week 13-16: Measurments completed
- Execution of measurements
- Refinement of setup
- Preliminary analysis of results
- Outline for chapter 5 (Implementation)

Week 17-20: Analysis completed
- Finalization of evaluation and data analysis
- Chapter 5 (Implementation)
- Chapter 6 (Evaluation Results)
- Chapter 7 (Discussion)

Week 21-24: Thesis writing completed
- Chapter 8 (Conclusion)
- Chapter 9 (Outlook)
- Abstract

Week 23-24: Thesis submitted
- Review, Finalization
- Buffer

References

[1] ESWIN, "EIC7700X SoC Technical Reference Manual." Accessed: Aug. 02, 2026. [Online]. Available: https://github.com/eswincomputing/EIC7700X-SoC-Technical-Reference-Manual/releases

[2] Pine64, "September Update: Check Your Notes." Accessed: Aug. 02, 2026. [Online]. Available: https://pine64.org/2024/10/02/september_2024/

[3] S. Schöning, "Porting and evaluation of a virtualized unikernel on RISC-V," Master's thesis, 2021.

[4] M. Ruppert-Maas, "Systematische Analyse und Optimierung eines Unikernels auf RISC-V: Vergleich mit einem minimalen Linux in ressourceneingeschränkten Systemumgebungen," Master's thesis, 2025.

[5] J. L. B. Fuentes Morales, "Evaluating Gem5 and QEMU Virtual Platforms for ARM Multicore Architectures," Master's thesis, 2016.

[6] A. Madhavapeddy *et al.*, "Unikernels: library operating systems for the cloud," *ACM SIGARCH Computer Architecture News*, vol. 41, no. 1, pp. 461–472, Mar. 2013, doi: 10.1145/2490301.2451167.

[7] K. Sreenivasan and A. J. Kleinman, "On the construction of a representative synthetic workload," *Commun. ACM*, vol. 17, no. 3, pp. 127–133, Mar. 1974, doi: 10.1145/360860.360863.

[8] S. Kuenzer *et al.*, "Unikraft: fast, specialized unikernels the easy way," in *Proceedings of the Sixteenth European Conference on Computer Systems*, Online Event United Kingdom: ACM, Apr. 2021, pp. 376–394. doi: 10.1145/3447786.3456248.

[9] K. Hu *et al.*, "Unishyper: A Rust-based unikernel enhancing reliability and efficiency of embedded systems," *Journal of Systems Architecture*, vol. 153, p. 103199, Aug. 2024, doi: 10.1016/j.sysarc.2024.103199.

[10] J. L. Hennessy and D. A. Patterson, *Computer Architecture: A Quantitative Approach*. Elsevier, 2011.

[11] B. Sá, J. Martins, and S. Pinto, "A First Look at RISC-V Virtualization From an Embedded Systems Perspective," *IEEE Transactions on Computers*, vol. 71, no. 9, pp. 2177–2190, Sept. 2022, doi: 10.1109/TC.2021.3124320.

[12] G.-M. Fragkoulis, N. Karystinos, G. Papadimitriou, and D. Gizopoulos, "Advancing Cloud Computing Capabilities on gem5 by Implementing the RISC-V Hypervisor Extension." Accessed: Feb. 19, 2026. [Online]. Available: http://arxiv.org/abs/2411.12444

[13] S. Park, H. Kang, and D. Kwon, "Beyond address spaces: In-process memory isolation for RISC-V," *Computers & Security*, vol. 163, p. 104812, Apr. 2026, doi: 10.1016/j.cose.2025.104812.

[14] "The RISC-V Instruction Set Manual: Volume II: Privileged Architecture." Accessed: Feb. 18, 2026. [Online]. Available: https://docs.riscv.org/reference/isa/_attachments/riscv-privileged.pdf

[15] A. Tanenbaum and H. Bos, *Modern Operating Systems*. Pearson International, 2014. Accessed: Feb. 20, 2026. [Online]. Available: https://elibrary.pearson.de/book/99.150005/9781292061955

[16] A. Bratterud, A.-A. Walla, H. Haugerud, P. E. Engelstad, and K. Begnum, "IncludeOS: A Minimal, Resource Efficient Unikernel for Cloud Services," in *2015 IEEE 7th International Conference on Cloud Computing Technology and Science (CloudCom)*, Nov. 2015, pp. 250–257. doi: 10.1109/CloudCom.2015.89.

[17] A. Madhavapeddy and D. J. Scott, "Unikernels: Rise of the Virtual Library Operating System: What if all the software layers in a virtual appliance were compiled within the same safe, high-level language framework?," *Queue*, vol. 11, no. 11, pp. 30–44, Nov. 2013, doi: 10.1145/2557963.2566628.

[18] S. Lankes *et al.*, "The Hermit Kernel." Accessed: Feb. 20, 2026. [Online]. Available: https://zenodo.org/records/14918547

[19] S. Lankes, S. Pickartz, and J. Breitbart, "HermitCore: A Unikernel for Extreme Scale Computing," in *Proceedings of the 6th International Workshop on Runtime and Operating Systems for Supercomputers*, Kyoto Japan: ACM, June 2016, pp. 1–8. doi: 10.1145/2931088.2931093.

[20] S. Lankes, J. Breitbart, and S. Pickartz, "Exploring Rust for Unikernel Development," in *Proceedings of the 10th Workshop on Programming Languages and Operating Systems*, in PLOS '19. New York, NY, USA: Association for Computing Machinery, Oct. 2019, pp. 8–15. doi: 10.1145/3365137.3365395.

[21] J. E. Smith and R. Nair, Eds., *Virtual machines: versatile platforms for systems and processes*. in The Morgan Kaufmann Series in Computer Architecture and Design. Amsterdam Boston: Morgan Kaufmann Publishers, 2010.

[22] A. Whitaker, M. Shaw, and S. D. Gribble, "Scale and Performance in the Denali Isolation Kernel," 2002. Accessed: Feb. 20, 2026. [Online]. Available: https://www.usenix.org/conference/osdi-02/scale-and-performance-denali-isolation-kernel

[23] R. Russell, "virtio: towards a de-facto standard for virtual I/O devices," *SIGOPS Oper. Syst. Rev.*, vol. 42, no. 5, pp. 95–103, July 2008, doi: 10.1145/1400097.1400108.

[24] A. Kivity, Y. Kamay, D. Laor, U. Lublin, and A. Liguori, "kvm: the Linux virtual machine monitor," in *Proceedings of the Linux symposium*, Dttawa, Dntorio, Canada, 2007, pp. 225–230.

[25] F. Bellard and others, "QEMU, a fast and portable dynamic translator.," in *Usenix ATC, Freenix Track*, 2005, pp. 41–46.

[26] C. Dall and J. Nieh, "KVM/ARM: the design and implementation of the linux ARM hypervisor," *Acm Sigplan Notices*, vol. 49, no. 4, pp. 333–348, 2014.

[27] R. Scheffel, "Simulation of RISC-V based Systems in gem5," Diploma Thesis, 2018. Accessed: Feb. 18, 2026. [Online]. Available: https://cfaed.tu-dresden.de/files/Images/people/chair-cc/theses/1808_Scheffel.pdf

[28] T. Mytkowicz, A. Diwan, M. Hauswirth, and P. F. Sweeney, "Producing wrong data without doing anything obviously wrong!," *SIGPLAN Not.*, vol. 44, no. 3, pp. 265–276, Mar. 2009, doi: 10.1145/1508284.1508275.

[29] A. Georges, D. Buytaert, and L. Eeckhout, "Statistically rigorous java performance evaluation," in *Proceedings of the 22nd annual ACM SIGPLAN conference on Object-oriented programming systems, languages and applications*, in OOPSLA '07. New York, NY, USA: Association for Computing Machinery, Oct. 2007, pp. 57–76. doi: 10.1145/1297027.1297033.