Project Proposal

A Study into Resource Competition for User-space Runtime Systems

CS546 – Parallel and Distributed Processing

February 2nd, 2026

Godha Pallavi Bhogadi

A20531496

Introduction

Modern operating systems are increasingly adopting microkernel designs, where traditional kernel responsibilities such as I/O management, storage services, and scheduling are delegated to user-space runtime systems. These runtimes rely heavily on asynchronous execution models, polling, and user-level thread scheduling to achieve performance and scalability. While this approach improves flexibility and scalability, it introduces new challenges related to resource competition and interference between runtime threads and application threads that compete for shared CPU resources. In this project, we study resource competition in user-space runtime systems using IOWarp, a distributed, multi-tiered I/O runtime designed for scientific computing workloads. Specifically, we investigate how runtime polling, busy waiting, and thread scheduling interfere with application execution and overall system performance. By systematically evaluating CPU utilization, scheduling fairness, and end-to-end performance under different runtime configurations, this project aims to provide empirical insight into the tradeoffs inherent in user-space OS designs.

Background & Related Work

Prior work on user-space OS scheduling highlights the fundamental limitation that user-level schedulers cannot fully control when or where native threads execute, resulting in unpredictable performance under load [1]. Coroutine based runtimes further demonstrate how multiplexing many tasks onto a small number of threads can increase utilization and polling overhead [2]. Polling-driven storage and I/O systems such as LabStor show that low-latency performance often comes at the cost of increased CPU consumption and reduced efficiency during idle periods [4]. Studies on serverless and cloud scheduling reveal that poor isolation and unfair scheduling policies can cause significant performance degradation when multiple workloads are run concurrently [3].More recent work on user interrupts explores preemptive user-space scheduling as a mitigation strategy, but shows that interrupt-based approaches introduce non-trivial overheads and remain constrained by kernel interactions [5, 6]. These limitations motivate a deeper evaluation of interference in real runtimes such as IOWarp.

Problem Statement

While user-space runtimes provide flexibility and performance benefits, their dependence on polling and user-level thread scheduling can lead to wasting CPU cycles and interfere with application threads. Existing mechanisms such as user interrupts partially address preemption but do not fully eliminate resource competition or guarantee fairness in scheduling policy. There is a lack of systematic, experimental evaluation of how runtime design choices such as thread count and core assignment impact performance and interference in practical systems.

Project Objectives

The objectives of this project are to:

- Measure CPU utilization and interference caused by polling and busy waiting in IOWarp.
- Study the tradeoffs between the number of threads and different core assignments to overall workload performance.
- Evaluate fairness and performance in the scheduling policy when multiple applications run concurrently.
- Develop a synthetic workload generator that models both busy waiting and IOWarp interaction.

Project Evaluations

Evaluation will be conducted using the Jarvis deployment framework for IOWarp. The runtime will be configured with varying numbers of threads and different core assignments. Initial experiments will use a synthetic workload generator with configurable busy waiting and I/O request rates. Later experiments will integrate real scientific applications. We will analyze metrics that includes CPU utilization, end-to-end latency, throughput, and fairness across concurrent workloads.

Conclusions

This project will provide insight into the costs and tradeoffs of user-space runtime designs by empirically evaluating resource utilization in IOWarp. By analyzing polling behavior, thread configuration, and multi-application interference, the study aims to identify runtime configurations that balance performance and efficiency. The results will be directly relevant to the design of future user-space runtimes and OS abstractions for scientific and high-performance computing.

Approximate Timeline

Week 1: Install IOWarp container and learn Jarvis deployment framework.

Week 2-3: Design initial benchmark cases.

Week 4: Evaluate the benchmarks.

Week 5-6: Integrate additional benchmarks.

Week 7: Continue evaluation.

Week 8: Final report and presentations.

References

[1] Jack Tigar Humphries et al. ghOSt: Fast & Flexible User-Space Delegation of Linux Scheduling. ACM SOSP, 2021.

[2] C.J. Williams, J.A. Elliott. Libfork: portable continuation-stealing withstackless coroutines arXiv, 2024.

[3] Yuqi Fu et al. Alps: An Adaptive Learning, Priority OS Scheduler for Serverless Function USENIX ATC, 2024.

[4] Luke Logan et al. LabStor: A Modular and Extensible Platform for Developing High-Performance, Customized I/O Stacks in Userspace SC, 2022.

[5] Linsong Guo et al. The Benefits and Limitations of User Interrupts for Preemptive Userspace Scheduling USENIX NSDI, 2025.

[6] Aleix Roca, Vicenc¸ Beltran. The Benefits and Limitations of User Interrupts for Preemptive Userspace Scheduling PPoPP, 2026

[7] IOWarp Project. https://github.com/iowarp/core