



FIU

Engineering
& Computing

Math Unikernel

Eliminating System Noise for Hardware-Accelerated
Workloads

Lucas Arabi, Eduardo Rodriguez, Luis Canessa, Ian Borges,
Julian Manrique



Team Composition



Lucas
Arabi



Eduardo
Rodriguez



Luis
Canessa



Ian Borges
Rivas



Julian
Manrique





Problem and Motivation

Standard operating systems are too loud

- **What is system noise?**
 - Unpredictable jitter caused by background tasks, hardware interrupts, and complex schedulers
- **Why is this a bottleneck for hardware-accelerated mathematical workloads?**
 - Steals critical CPU cycles and destroys execution determinism
 - The bottleneck destroys execution determinism, making it impossible to guarantee exact execution times for precision mathematical operations
- **What is the issue we plan to solve?**
 - Create a specialized, 64-bit bare-metal operating system that creates a deterministic, single-address-space environment



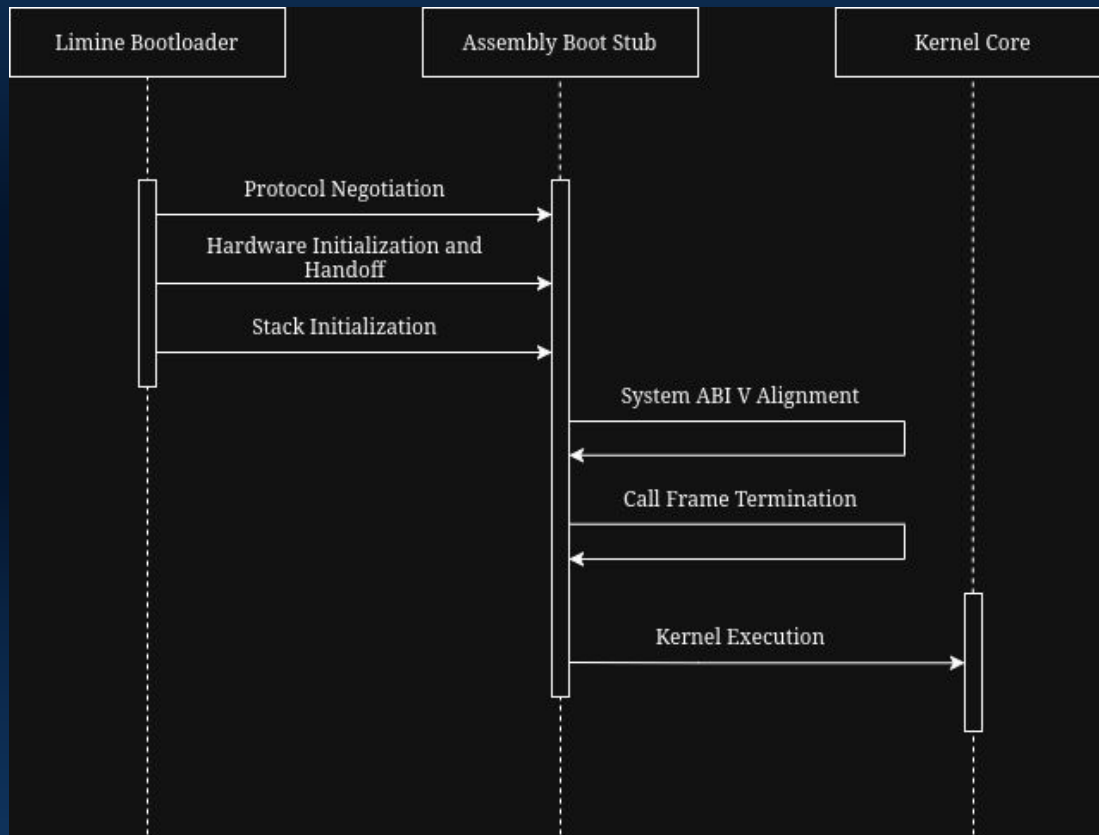
The Solution

A custom-built, specialized 64-bit
Bare-metal environment,

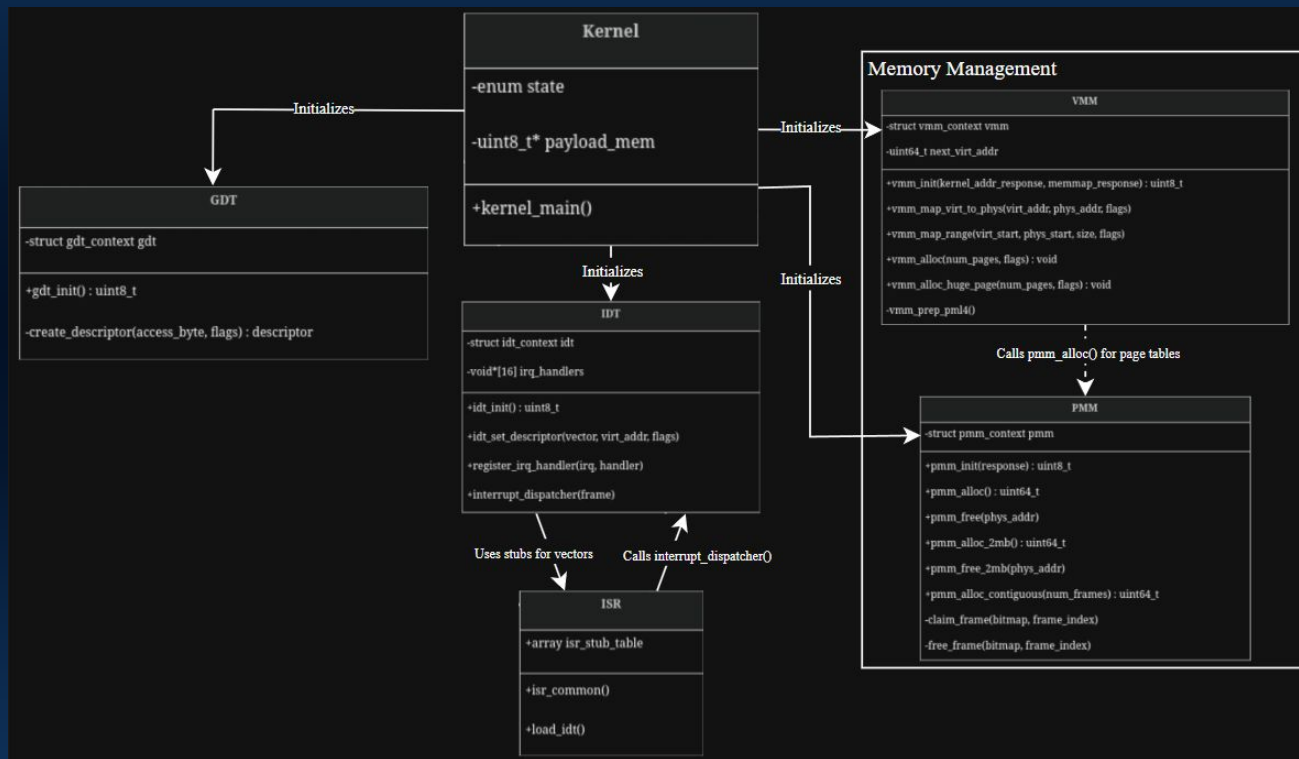
- Created a highly deterministic, single address space environment. Bypasses legacy BIOS/UEFI overhead using the Limine Boot Protocol to immediately enter 64-bit Long Mode
- Created a VS Code extension for users



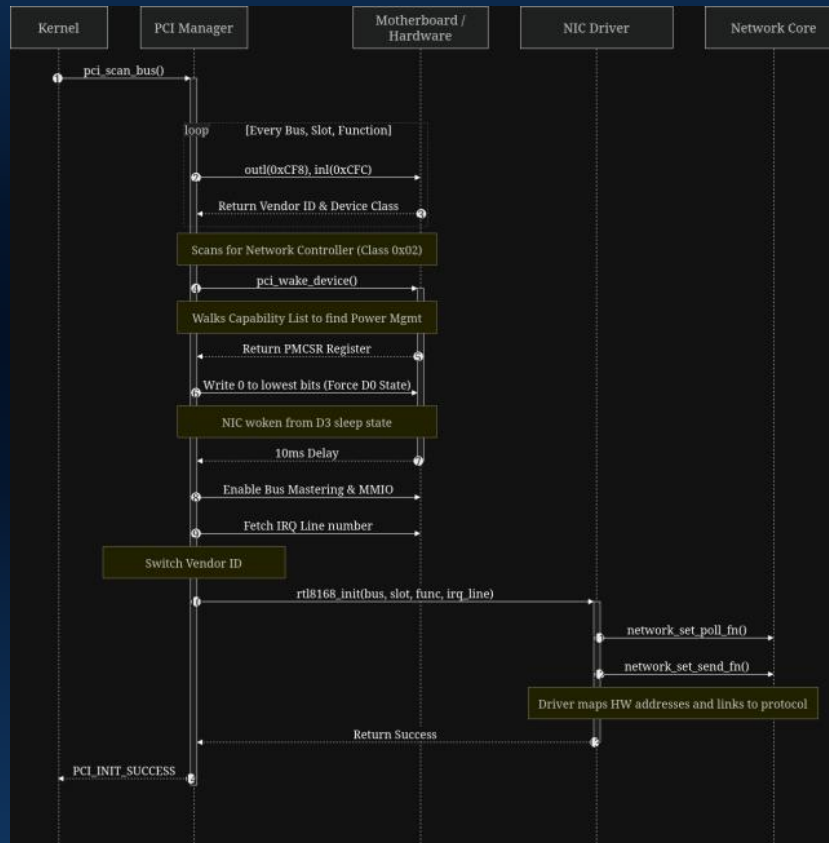
Boot Diagram



Core Kernel Diagram

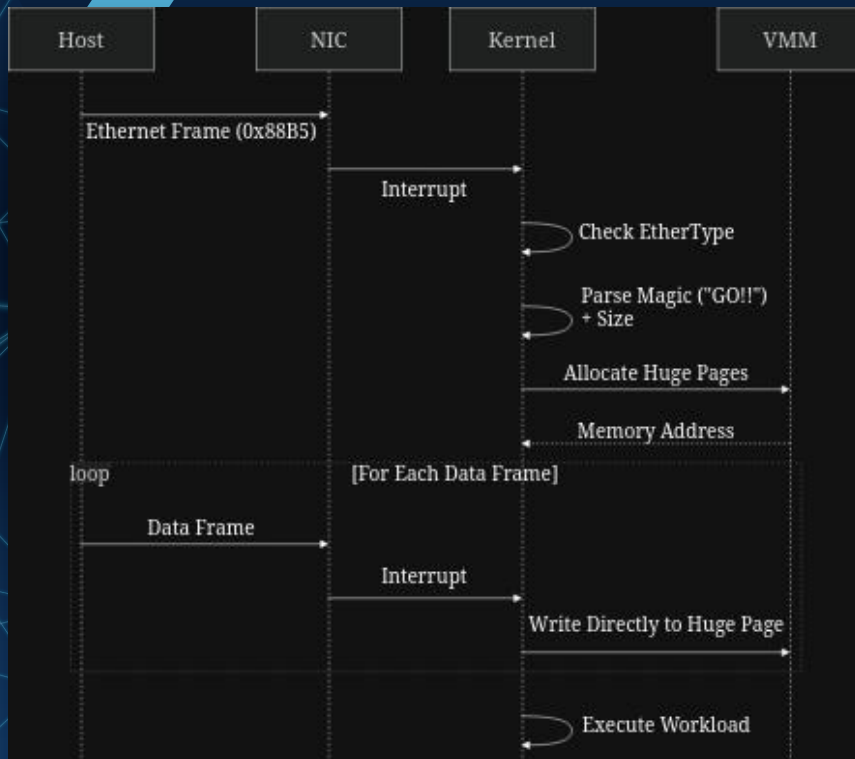


Network Diagram

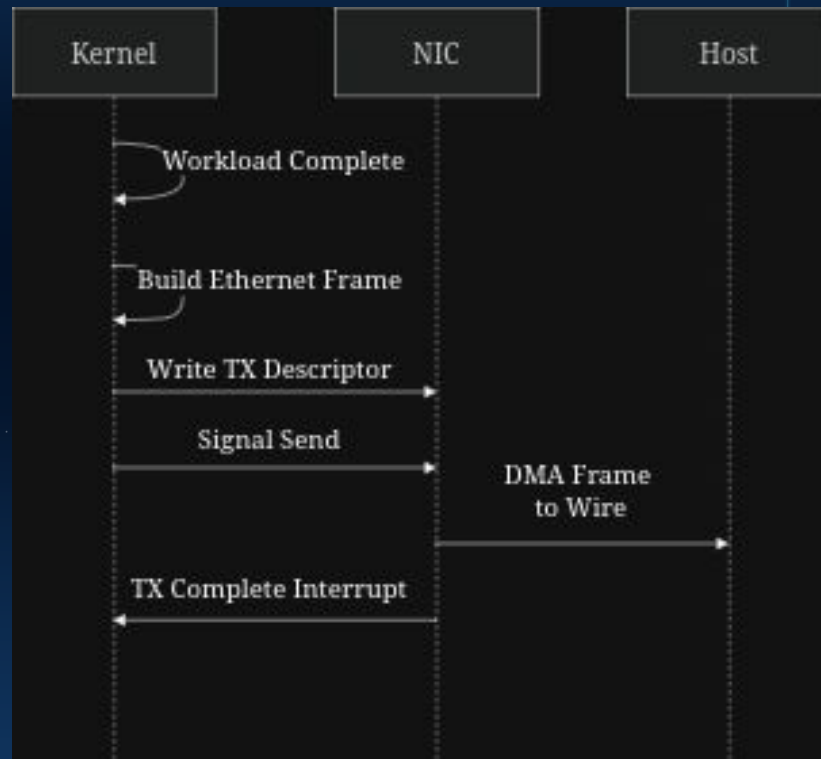


Network Diagram

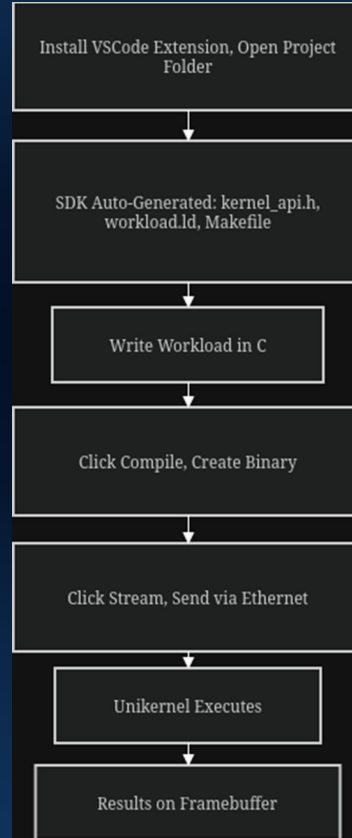
Receive Protocol



Transfer Protocol



User Journey



Benchmark Summary



Benchmark Analysis

The Unikernel matches Linux on simple workloads (dot product) and significantly outperforms it on complex workloads (GEMM).

Dot Product (~equal): A short, memory-bound operation. Each iteration completes in ~9ms — fast enough to slip between Linux scheduler ticks (~4ms). Little opportunity for OS interference.

GEMM (26% faster): A long, compute-bound operation. Each iteration takes ~1–1.5 seconds — enough time for Linux to context switch, handle interrupts, and service background processes. The Unikernel runs uninterrupted, executing the full workload without yielding the CPU.

Under load, the gap widens: Linux GEMM slows to 1771ms (+64% vs Unikernel) with high variance (± 77 ms). The Unikernel remains deterministic — same time every run.

Conclusion: Bare-metal execution eliminates OS overhead. The benefit scales with workload duration and compute intensity.

+++

Product Demo

+++

Challenges and lessons learned

For most of our CS careers, operating systems have been black boxes that simply abstract away the lowest level of the software we use every day. At best, we study system calls and algorithms that are commonly used in an OS.

- Stripped away every layer of abstraction — down to assembly — to understand how real systems are constructed
- Built hardware drivers, paging, IDT, and AVX/SSE from datasheets with no OS safety net
- System cohesion was the hardest design problem — every component (bootloader, memory, networking, workload execution) had to slot together precisely
- Debugging bare-metal systems with no standard tools developed techniques that transfer to any complex software environment

The result: deep OS intuition that directly improves how we write software at every level

Future Work

There are countless ways this project can (and hopefully will be) pushed forward!

- More driver support, fixing TX
- Expanding the math library
- Expanded networking protocol – enabling internet connection
- Algorithmic optimization in core systems
- Rewrites in other languages (i.e. Rust)

The background is a dark blue gradient. In the top-left corner, there is a complex network of thin, light blue lines forming a web-like structure. In the top-right corner, there are several 3D cubes of varying sizes, some in shades of blue and others in a lighter cyan, appearing to float or be part of a larger structure. In the bottom-left corner, there are three white plus signs stacked vertically. In the bottom-right corner, there is another network of thin, light blue lines, similar to the one in the top-left but less dense.

The End

Thank you!

