Luis Felipe Barro Varoni

luisf.vbarro@gmail.com linkedin.com/in/luisfvbarro github.com/Patea4

Education

University of Toronto **Sep. 2023 – Expected May 2027**

Bachelors of Science in Computer Science and Mathematics; GPA: 3.90/4.0 *Toronto, Canada*

- Relevant Coursework: Operating Systems (99), Machine Learning (92), Computer Organization (92), Cybersecurity (91)

Awards

University of Toronto — President's Scholars of Excellence; Dean's List

CTF Achievements — Representing **UoftCTF** (Top 2 in Canada) and **Byte Babies** (Top 3 in Canada)

Skills

Programming Languages: Python, C, C++, Java, Rust, TypeScript, RISC-V/x86-64 Assembly
Systems/HPC: GPU Programming (CUDA), Parallel Programming, Docker, Linux, CI/CD, RabbitMQ, REST APIs
Frameworks/Tools: PyTorch, FastAPI, Next.js/React, Git

Experience

Roblox **May 2026 – September 2026**

Software Engineer Intern (Incoming) *Vancouver, Canada*

- Joining Roblox's Rendering & Graphics team to contribute to their in-house C++ rendering engine.

University of Toronto **September 2025 – Present**

Teaching Assistant — CSC236, CSC258, CSC398 *Toronto, Canada*

- Teaching assistant for Theory of Computation, Computer Organization and Software Development courses

Google Developer Student Club **January 2025 – April 2025**

Backend Developers *Mississauga, Canada*

- Worked with the GDSC team and a UofT professor to develop an educational platform leveraging AI to generate educational animated videos for math related topics
- Designed and implemented a scalable video rendering pipeline with Docker and RabbitMQ, enabling real-time streaming of animations that previously required several minutes per render.
- Oversaw deployment of the FastAPI backend with Nginx and the LLM on the Niagara supercomputer

UofT & The Sick Kids Foundation **January 2025 – April 2025**

Software Developer *Toronto, Canada*

- Partnered with the team at The Sick Kids Foundation to develop a platform tracking usage statistics of IoT devices.
- Coordinated 6-person dev team using Scrum methodologies, ensuring on-time delivery for research partners
- Developed a web application using Next.JS and FastAPI, where patient data can be monitored and analyzed efficiently.
- Containerized the Next.JS + FastAPI stack with Docker and set up CI/CD using Railway

Projects

Scalable Parallel Memory Allocator | *C, Pthreads*

- Implemented a Hoard-style parallel `malloc/free` in C with per-CPU Heaps and size-class freelists, achieving near-linear throughput scaling across cores.
- Eliminated false sharing via page-aligned superblocks with per-heap metadata and origin-aware block reuse on `free`.
- Minimized lock contention through fine-grained per-heap mutexes and a lock-free thread-local fast path.

Fault-Tolerant Distributed Key-Value Store | *C, POSIX Threads, Distributed Systems*

- Implemented a replicated in-memory distributed key-value store with a Coordinator and multiple KV servers.
- Added heartbeat-based failure detection and replica recovery while maintaining client availability.
- Built concurrent client/server messaging for consistent replication, failover, and recovery-aware routing.

Systems Programming Projects | *C, Linux*

- Implemented a FUSE-based extent file system (EXFS) supporting inline data, extent-based allocation with greedy coalescing, data checksums, and full read/write/truncate via user-space callbacks.
- Developed a user-level threading library with context switching, cooperative scheduling, and synchronization primitives.
- Created a lightweight parallel web server supporting concurrency and request multiplexing, handling 1,000+ simultaneous connections with low latency.

SauceEngine | *C++, Vulkan, Slang, GLFW, GLM*

- Designed and implemented a Vulkan 1.3 rendering backend using dynamic rendering and RAII resource management, with double-buffered frame submission and manual synchronization.
- Implemented a Slang PBR shader with five texture maps, SSBO lighting, HDR tonemapping, and glTF 2.0 model/texture loading in a two-pass post-processing pipeline.