Mahdi Ali-Raihan

mma2268@columbia.edu | linkedin.com/in/mahdi-ali-raihan | github.com/asder8215 | asder8215.github.io/

Education

Columbia University New York, NY

Bachelor of Science in Computer Science *Sep. 2021 – May 2025*

Relevant Coursework: Advanced Systems Programming, Artificial Intelligence, Cloud Computing, Computer Architecture, Distributed Systems, Embedded Systems, Malware Analysis and Reverse Engineering, Operating Systems, Parallel Optimization for Robotics

Experience

Teaching Assistant Sep. 2024 - May 2025

Columbia University *New York, NY*

- Held weekly office hours for 300+ students in COMS W4701 Artificial Intelligence and 80+ students in COMS 4160 Computer Graphics
- Aided with grading and writing test scripts assignments and exams, answering students' questions on discussion board, and hosting review recitations

Software Engineer Intern June 2022 - Aug. 2022

Ceros *New York, NY*

- Created end-to-end automated testing scripts using Webdriver IO, Cucumber, and TypeScript, which assisted Ceros in efficiently finding bugs in their TextPlus and Previewer tools
- Performed manual testing and ticketed issues using Jira and Google Doc that expedited the resolution of bugs and enhanced quality of Ceros' TextPlus tool
- Worked in a cross functional team setting with Software Engineers, UX/UI Designers, and Product Managers through agile meetings

Projects

NES Emulator (In Progress) | *Rust, Emulation, Concurrent Systems, Retro Console Architecture, Open Source*

- Developing an open source NES console emulator in Rust for the purpose of learning about the NES architecture and how emulation works
- Contains modules for CPU (complete), Bus, Cartridge ROM mapping, PPU, GamePad parsing, and APU

Sharded Ring Buffer | *Rust, Data Structure, Async, Atomics, Concurrency, Open Source*

- Developed novel concurrent data structures for Rust's async ecosystem, including a hybrid lock-free/async-waiting shard buffer and a fully atomic sharded queue design
- Introduced batch enqueue/dequeue APIs on top of a CAS-based shard lock, achieving throughput of **600+ million ops per second** (8 threads, 8 shards, cap=1024), which is at least **9.5x** better than many asynchronous Rust channels (i.e. kanal-async, flume-async, async-channel, etc.)

Sharded Cache Map | *Rust, Data Structure, Async, Atomics, Cache, Concurrency, Open Source*

- Developed an efficient and concurrent cache data structure for Rust's async ecosystem using atomic primitives, a sharded Hashtable with bounded queue design, and FIFO/LIFO/Double Hashing eviction policies
- Achieved throughput of **8+ million operations per second per thread** (with 10,000 cached keys randomly queried using 8 threads) competing with widely used and known cache crates in Rust like moka, cached, and stretto

Gmail Management | *Rust, Gmail API, Serde JSON, Command Line Argument Parser (CLAP), Lettre, Tokio*

- Created a command line program for users to trash messages, query related searches, and send messages with attachments in their Gmail client or send messages through a third party mail service
- Trashing and querying email messages is optimally performed through asynchronous parallelization of workload
- Querying or sending messages within a user's Gmail client is enabled through deserializing JSON-formatted files

2v2 ESP32 Spaceteam | *C++. Concurrency, Embedded Systems, ESP32, ESPNOW, WiFi*

- Collaborated on the development of a competitive 2v2 team-based interactive game inspired by the Spaceteam mobile game
- Designed the potential UI of the game, navigation flow between screens, and the game mechanisms behind button clicks

- Implemented the Room Screen UI and logic for the lobby navigation flow, allowing players to be placed in private rooms
- Handled and synchronized communication logic between ESP32s and what is drawn on screen with tasks including: join and leave requests, command exchange requests, win requests, etc.

Arcade Poker: Balatro Minus | *C, Embedded Systems, Game Design*
- Collaborated on an arcade poker game inspired by Balatro using an FPGA, VGA monitor, and SNES controller
- Created all pixel art and game assets as well as fleshed out the navigation flow and interaction design behind what tiles are modified on screen from button inputs
- Implemented a tileset and tilemap generator script to translate all indexed png assets into a .hex file that the hardware can read and appropriately display the graphics of the tiles onto the screen

TeXiT | *C, Bash, GTK4, Libadwaita, Blueprint Language for GTK interface, JSON-C*
- Simple UNIX-based text editor application with remote editing capabilities
- Developed file save functionality, tab features, GTK signal handlers for tab pages + file action buttons (e.g. new file, open file, save file), and serialization/deserialization of JSON strings for communication between server and clients
- Developed file save functionality, tab features, GTK signal handlers for tab pages + file action buttons (e.g. new file, open file, save file), and serialization/deserialization of JSON strings for communication between server and clients

Debugger | *C, x86-64 Assembly, Linux, ptrace*
- Developed a debugger using ptrace library that operates similarly to GNU Debugger Project, allowing users to dynamically analyze and debug other executable programs
- Functionality includes: setting breakpoints, performing single or continuous steps, inspecting/editing memory addresses, and displaying a stack backtracing for errors on provided executable

Linker | *C, x86-64 Assembly, Linux, ELF*
- Created a simple linker that can link, perform multi-file resolution and relocation on an arbitrary number of object files generated from C or Assembly code, and output a .out executable with an appropriately filled out ELF table

Simple Shell | *C, Shell, Syscalls, Signals*
- Created a Bash-like shell in C equipped with signal handling capabilities enabled through system calls

Paxos-based Key Value Service | *Go, Distributed Systems, RPC*
- Implemented a fault-tolerant key/value storage system using Paxos consensus protocol to ensure reliable replication and ordering of client operations across multiple servers
- Contains robust RPC-based communication for client-server and server-server interactions, ensuring consistency and correctness in the presence of network failures and server crashes

Gnosis | *Python, TypeScript, AWS, MySQL, FastAPI, Flask, OpenAI, OpenAPI*
- Collaborated in developing a cloud-based AI-powered personal learning platform deployed on the cloud that enhances user engagement with reading materials by allowing uploads of PDFs, articles, and books
- Implemented a REST API that managed the resources for the Conversation microservice, such as creating or deleting conversations, adding replies, and fetching conversations
- Optimized the parameters for the response agent to provide engaging and intellectual conversations between the user and the agent

AI Upscaler | *Python, AI, PyTorch, Scikit-Learn, Scikit-Image, OpenCV*
- Developed an AI image upscaling tool to enhance the resolution and quality of low-resolution images to high-resolution versions and extended it to support video upscaling
- Created a PatchGAN-based upscaler model to promote image quality for generated high-resolution images by adversarially scoring patches of the images
- Trained and tested the model on an anime dataset, passing the training dataset through 10 epochs, performing 5 fold cross validation, and using MSE for pixel losses and BCE for adversarial and scoring losses as well as an Adam optimizer

Technical Skills

Languages: Rust, C/C++, Go, Python, TypeScript, Bash
Operating Systems: Windows 10, MacOS, Ubuntu, Fedora Linux
Developer Tools: AddressSanitizer, Cargo, Criterion (Rust microbenchmarking), Git/GitHub, Google Cloud Platform, PyCharm, Valgrind, Vim, VS Code