Tran Manh Duy

096 326 9684 | tmd.iid2004@gmail.com | LinkedIn | GitHub

Summary

Systems-oriented Rust developer with a strong foundation in Computer Science and low-level programming. Experienced in building high-performance, low-latency services with a deep understanding of systems programming concepts including memory management, concurrency, and inter-process communication (gRPC, message queues). Proficient in designing modular, trait-based architectures and real-time event processing pipelines using Kafka and async Rust (Tokio). Eager to apply systems-level expertise to security infrastructure and real-time telemetry platforms.

Education

University of Engineering and Technology, VNU — Hanoi, Vietnam

BS in Information Technology, CPA: 3.0/4.0 — *Sept 2022 – Sept 2026*

Available for Full-time employment

Skills

Programming Languages: Rust (Advanced), **C/C++** (Intermediate), Golang, Typescript

Systems & Concurrency: Tokio (async runtime), memory management (ownership, lifetimes, unsafe), concurrency primitives, trait-based design patterns, IPC (gRPC, message queues)

Frameworks & Protocols: Axum, Actix, Tonic (gRPC), Prost (Protocol Buffers)

Infrastructure: Kafka, RabbitMQ, Redis, Docker, Kubernetes, Linux, Git/GitHub, CI/CD (GitHub Actions)

Database: PostgreSQL, MySQL, MongoDB, SeaOrm, DieselOrm

Cryptography & Security: Custom ECC implementation, digital signatures (Schnorr, ECDSA), Montgomery field arithmetic, ZKP (zk-SNARKs, zk-STARKs)

Testing: Unit/Integration testing, Testcontainers, Mockall, load testing (k6), benchmarking (Criterion)

Language: English, Japanese

Experience

Sotatek ., JSC — Hanoi, Vietnam

Rust Developer — *Jan 2024 – Present*

- Designed and implemented high-performance **RESTful APIs** (using Axum/Actix) and **gRPC services** (using Tonic, Prost) to power CEX/DEX platforms, serving both web and mobile interfaces with low latency.
- Engineered scalable asynchronous messaging systems using **Kafka** and **RabbitMQ** to handle high-frequency trading data and real-time blockchain event processing.
- Optimized database performance and data reliability by integrating **Redis** for caching and **SeaOrm** for type-safe interactions with PostgreSQL.

Blockchain Developer

- Developed and deployed a Starknet Layer 2 solution integrated with Aptos Layer 1, including data availability on Move, Madara chain configuration, and a Rust-based prover for on-chain verification.

Research Assistant

- Researched, documented, and implemented algorithms related to Zero-Knowledge Proofs (ZKPs), including zk-SNARKs and zk-STARKs.

Projects

High-Performance Trading Engine

- **Stack**: Rust, Cargo Workspace (hexagonal architecture), Linux io_uring, Criterion benchmarks.
- Built a **single-threaded, zero-copy** trading engine with an **L3 limit order book** using hardware-accelerated **bitmap indexing** for O(1) best-bid/ask lookup and cancel operations (~**257 ns** taker match, ~**138 ns** cancel).
- Implemented an **io_uring TCP gateway** for fully asynchronous, zero-syscall batched I/O with real-time **BBO market data broadcast** (fan-out on every book mutation) and a **write-ahead log** for crash-fault tolerance with async persistence and startup replay.

- Designed a **zero-copy wire protocol** using packed C-repr structs transmitted directly over TCP with no serialization overhead, achieving ~**107 µs** wire-to-wire round-trip latency (CI-verified via GitHub Actions).
- Enforced **compile-time layer boundaries** (Domain / Application / Gateway) following hexagonal architecture for deterministic event processing.
- **Github**: https://github.com/Tranduy1dol/trading

High-Performance E-commerce Microservice

- **Stack**: Rust (Axum, Tokio), PostgreSQL (SeaORM), Redis, Docker, GitHub Actions, k6.
- Architected a scalable backend using **Clean Architecture** and **Cargo Workspace** pattern, decoupling business logic (`core`) from infrastructure (`infra`) with **trait-based port/adapter** interfaces.
- Solved critical **race conditions** in inventory management ensuring data integrity under high concurrency (verified via Testcontainers integration tests).
- Achieved ~**8,000 RPS** with 4.51ms avg latency (P95: 7.61ms) through **Redis caching** strategies and **load testing** (`k6`).
- Established a complete **CI/CD pipeline** using GitHub Actions for automated testing and Docker containerization.

- **Github**: https://github.com/Tranduy1dol/shopping-cart