Aman Karki

Software Engineer | Systems, AI Infrastructure & LLM Inference | C++, CUDA, Python
India | GMT+5:30 | itsamankarki@gmail.com | +91 7310808520 | LinkedIn | GitHub | Portfolio

SUMMARY

Software Engineer specializing in high-performance C++ systems, CUDA, and applied AI. Engineered three production-grade systems from scratch — an LLM inference engine, an embedded vector database, and an AI-powered architectural analysis platform — each independently verified for correctness and benchmarked, not assumed. Two CUDA kernels merged into llama.cpp (128,000+ GitHub stars), both directly reviewed or merged by the project's creator, Georgi Gerganov.

TECHNICAL SKILLS

Languages: C++20, Python, CUDA, SQL
AI, LLM & Agentic Systems: LLM Inference, Retrieval-Augmented Generation (RAG), GraphRAG, Agentic AI Workflows, Vector Databases, HNSW, INT8 Quantization, pybind11
Low-Level Design & Systems: Object-Oriented Programming (OOP), SOLID Principles, Design Patterns, Multithreading (std::jthread, std::atomic), RAII & Smart Pointers, Cache-Aware Design
Data Structures & Algorithms: Graphs (BFS/DFS, Topological Sort, Cycle Detection), Dynamic Programming, Hashing, Heaps, Two Pointers, Bit Manipulation
Tools & DevOps: CMake, Git (Version Control), GitHub Actions (CI/CD), Docker, FastAPI (REST APIs), PyPI Packaging, Linux

EXPERIENCE

llama.cpp (ggml-org) — CUDA Backend Contributions PR #27573 | PR #28897
C++, CUDA

- Implemented a CUDA kernel for one-dimensional pooling (average and max modes), closing a GPU backend coverage gap; verified across 216 test cases spanning all kernel size, stride, and padding combinations on two Nvidia T4 GPUs, merged into master by the project's creator, Georgi Gerganov.
- Enabled i16 and i32 support for GGML_OP_DUP on CUDA, fixing a gate that silently fell back to CPU; verified against the full backend test suite (16,097/16,097 tests, zero regressions) on two T4 GPUs, reviewed and approved by the project's creator, Georgi Gerganov.

PROJECTS

verbum.cpp — LLM Inference Engine GitHub | Writeup
C++20, CUDA, Python, pybind11

- Engineered a transformer-based LLM inference engine from scratch in C++ (tokenizer, attention, KV-cache, sampling) with no dependency on PyTorch or existing runtimes; validated output against HuggingFace to a maximum logit deviation of 3e-5.
- Designed custom CUDA kernels (tiled matmul, RMSNorm, RoPE, grouped-query attention) achieving 723 GFLOP/s, a 343x speedup over the CPU baseline, with output parity verified exactly against the CPU path.
- Built an INT8 quantization pipeline reducing quantized-layer memory 4x with exact-match correctness; integrated via pybind11 with a custom HNSW vector database for a memory-augmented offline demo, correctly managing the GIL to keep the app responsive.

Lattice — Embedded Vector Database GitHub | Writeup
C++20, Python, HNSW, pybind11, FastAPI, GitHub Actions, CMake

- Engineered an embedded vector database from scratch in C++20: a hand-implemented HNSW index, a write-ahead log storage engine with crash recovery, and a concurrent query path validated under ThreadSanitizer.
- Benchmarked at 583µs p50 latency, 95.4% recall on SIFT (10K vectors, 128-dim) — ~3.5x faster than Qdrant's in-memory mode; reduced per-vector storage 4x via scalar quantization (float32 → uint8), accuracy tradeoff directly measured.
- Published pylattice-db to PyPI via pybind11 bindings, backed by 30 automated GoogleTest cases and a CI pipeline that fails builds on benchmark regressions.

RAAG — AI-Powered Architectural Analytics Platform GitHub | Writeup
C++20, Python, GraphRAG, Docker, GitHub Actions

- Engineered a parallel C++20 source-parsing engine using a std::jthread pool with cooperative cancellation via std::stop_token, achieving a 3.69x speedup (1,290.2 vs. 349.4 files/sec) across 579 real-world files with zero parse failures.
- Designed a dependency-graph analytics engine computing coupling, instability, and LCOM (Lack of Cohesion of Methods) metrics, surfacing 9 threshold violations and 1 circular dependency across 913 real dependency edges.
- Built a RAG pipeline scoping AI refactoring suggestions to a computed blast radius via metadata-filtered vector search in Qdrant, assembling grounded prompts from retrieved code chunks.
- Implemented a self-hosted CI/CD gate in GitHub Actions (86% coverage, 307 tests) detecting real architectural violations on every pull request; published a VS Code extension wrapping the CLI for guaranteed output parity.

EDUCATION

Bachelor of Technology, Computer Science and Engineering (Cyber Security and Forensics)
University of Petroleum and Energy Studies | 2024