Akshith Macharla

Chico, CA | amacharla@csuchico.edu | +1 530-965-7864 | github.com/amacharla15 | linkedin.com/in/akshith-macharla-

Education

California State University, Chico Chico, CA
Master of Science in Computer Science *Dec 2026*
Relevant Coursework: Operating Systems, Machine Learning, Computer Vision, Data Structures and Algorithms, Software Design

Technical Skills

- **Languages:** C++, CUDA, Python, PostgreSQL
- **ML / AI Infrastructure:** PyTorch, PyTorch Distributed, Transformers, vLLM, JAX, ONNX, MPI, NCCL
- **GPU / Distributed Systems:** CUDA kernels, NCCL, A100 benchmarking, Linux, Docker, Kubernetes, Grafana, Prometheus
- **Compiler / Runtime:** torch.compile, PyTorch FX graphs, graph capture, tracing, quantization, Nsight Systems, PyTorch Profiler
- **Systems Programming:** multithreading, thread pools, synchronization, performance profiling, AWS, Spring Boot, TCP/IP, UDP
- **AI / LLM Engineering:** RAG, Vector Databases, FAISS, LangChain, MCP, Agentic Frameworks, Cursor

Open Source Contribution

NVIDIA CUTLASS – Blockwise FP8 GEMM Split-K Fix (GitHub – PR #3615) | C++, CUDA, CUTLASS, GEMM, FP8 *Sep 2026*

- Submitted an upstream fix to NVIDIA CUTLASS for a serial split-K correctness bug where blockwise GEMM kernels applied incorrect scale factors because scale indexing ignored each slice's global K-tile offset.
- Traced the failure through the CUTLASS GEMM mainloop and propagated the split-K tile offset through the kernel path so each slice selects the correct K-block scales while preserving existing single-slice behavior.
- Added SM89 regression coverage using FP8 GEMM with $K = 512$, four distinct scale blocks, split-K configurations of 1, 2, and 4 slices, and multiple threadblock swizzles; validated the fix on an RTX 4090 with CUDA 12.4.

Experience

Cognizant Technology Solutions Hyderabad, India
Programmer Analyst / Associate Software Engineer *Dec 2021 – Aug 2024*

- Built and maintained Java/Spring Boot backend services for policy lookup, claims intake, workflow status updates, customer validation, and audit-history retrieval for 10,000+ active users.
- Designed REST API workflows with request validation, structured error responses, exception handling, and consistent response contracts for frontend and internal consumers.
- Implemented business-rule validation for policy status, claim eligibility, mandatory document checks, user permissions, and workflow state transitions across transaction-heavy insurance flows.
- Improved p95 API latency from ∼250ms to ∼200ms by optimizing SQL/Hibernate data access, reducing redundant reads, and refactoring service-layer logic. Triaged 100+ production issues involving API failures, database inconsistencies, service errors, and performance regressions; supported post-release monitoring to maintain 99.9% service availability.
- Wrote JUnit/Mockito tests for core validations and workflow transitions; validated REST APIs using Postman before QA handoff and release sign-off.
- Supported CI/CD and release activities using Git, Maven, Jenkins, Linux, and JIRA, including build validation, deployment checks, regression testing.

California State University, Chico Chico, CA
Research Assistant – 3D Computer Vision / Sensor Data Processing *Feb 2026 – Present*

- Built C++/OpenCV tooling around Intel RealSense D455F RGB-D recordings to replay sensor data, align depth/color frames, process raw depth, and generate road-surface analysis outputs.
- Developed reproducible Linux workflows for depth filtering, road-plane fitting, deviation heatmaps, candidate masks, ROI measurement, overlays, and frame-level reports.

Selected ML Systems Project

LLMPort-A100 – Distributed Transformer Inference Benchmarks | Python, PyTorch Distributed, NCCL, vLLM, torch.compile, Linux *May 2026*

- Benchmarked vLLM serving for Qwen2.5-1.5B-Instruct on 2× NVIDIA A100-SXM4-80GB GPUs using tensor parallel size 2, measuring TTFT, TPOT, tokens/sec, prompt length, output length, and concurrency behavior.
- Implemented NCCL benchmarks for all-reduce, all-gather, reduce-scatter, and broadcast across tensor sizes up to 256 MB, reaching 172.56 GB/s all-reduce and 252.87 GB/s broadcast bandwidth.
- Captured PyTorch FX and `torch.compile` graphs for a mini transformer block and built an operator support matrix covering Linear, matmul, softmax, LayerNorm, GELU, residual connections, and tensor layout operations.

GPU & LLM Systems

CUDA and Triton Transformer Kernels, Post-Training | CUDA, C++, PyTorch, JAX, Nsight Systems *Nov 2025 – May 2026*

- Built a CUDA and JAX transformer-operator benchmark suite for LLM inference, implementing RMSNorm, softmax, tiled GEMM, top-k sampling, multi-head attention, and FlashAttention-style kernels with PyTorch parity tests.
- Profiled kernels with Nsight/PyTorch timers to analyze memory-bound behavior, coalesced access, shared-memory tiling, synchronization overhead, register pressure, occupancy limits, and warp-level bottlenecks.
- Implemented post-training workflows covering knowledge distillation loss, teacher/student logits, soft targets, temperature scaling, DPO, PPO/RLHF pipeline structure, reward models, and preference data.

LLM Inference Runtime Design | Python, C++, Continuous Batching, KV Cache, Scheduling *Jun 2026*

- Designed LLM serving schedulers covering continuous batching, decode-first scheduling, chunked prefill, arrival queues, max batch-size limits, token budgets, EOS handling, cancellation, timeout, and latency metrics.
- Implemented KV-cache manager designs with fixed-size block allocation, free-block pools, request-to-block tables, prompt allocation, decode-token append, capacity checks, ref-counted freeing, physical-location lookup, and OOM handling.

Coding Profiles & Certifications

Coding Profiles: GitHub | LeetCode: 1600+ rating | LeetGPU | AWS Certified Developer Associate | Meta Backend Developer (Python)