Shlok Limbhare

Bengaluru, India · Shlokvfx2003@gmail.com · +91-8698820100 · LinkedIn · GitHub · LeetCode · Tensara

About Me

GPU-focused Inference Engineer with expertise in high-performance LLM deployment and optimization. Specializes in designing efficient attention kernels, Flash Attention, and Blockwise GEMM across **CUDA, Triton, HIP,** and **CuTe**. Deep understanding of KV-cache behavior, batching strategies, speculative decoding, and parallelism for low-latency, high-throughput workloads. Passionate about bridging algorithmic innovation with production-ready systems — building scalable inference infrastructure on heterogeneous GPU architectures.

Education

Birla Institute of Technology and Science (BITS) Pilani **2024 – 2028**

B.Sc. Computer Science

Relevant Coursework: Data Structures & Algorithms, Algorithm Design, Operating Systems, Databases, Networked Services, Software Design

Experience

DNEG — *Effects Technical Director* **2022 – 2024**

- Engineered large-scale **physics-based simulations** (fluids, smoke, debris, particles) in **Houdini FX**, optimizing solver performance and memory stability under strict render-budget constraints — building hardware-bottleneck intuition directly applicable to GPU kernel profiling.
- Operated under tight production deadlines with disciplined **version control**, written **runbooks**, and cross-team incident handoffs; mirrors the operational discipline of production inference on-call rotations.

Actively Building Projects

Mini-Attention: Efficient GPU Attention Kernels with Flash Attention GitHub

- Implemented Standard, Ring, KV-Cached/Paged, and **Flash Attention** across **PyTorch, Triton, and CUDA**; benchmarked single- and multi-GPU throughput and memory efficiency with modular design supporting future MoE and multi-task extensions.
- Maintained **three parallel branches**: RTX 3060 (default), **NVIDIA Blackwell B200**, and **AMD MI300X (HIP/ROCm)** — demonstrating cross-architecture portability required by production backends like vLLM and TensorRT-LLM.
- Profiled **SM occupancy** and **HBM bandwidth** using **nsys/ncu**; results documented alongside modular kernel interfaces for continuous extension.

WaveBoost: Inference Optimization Practice GitHub

- Developed standalone **PyTorch/CUDA/Triton kernels** for **prefill vs. decode separation**, **selective and continuous batching**, and **MHA/GQA** reimplementations — mirroring production vLLM and TensorRT-LLM patterns.
- Profiled kernel execution for **SM occupancy** and **HBM throughput** using **ncu/nsys** on **H100**; maintained structured benchmarks and design notes per experiment.
- Structured as a reproducible research scaffold with per-kernel tests and documented performance targets — analogous to writing design docs and CI-gated tests for model-serving features.

Awards

AMD 2025 Inference Hackathon **2025**

- Secured **top-performing ranking** implementing Mixed-Precision **Blockwise GEMM** (FP8 → BF16) on **AMD MI300X**, achieving **6.7× speedup** over PyTorch baseline.
- Applied **Matrix Cores, double buffering, XOR-based swizzling**, and multi-level tiling — same low-level techniques used in production TensorRT-LLM and vLLM GEMM kernels.
- Published a detailed by Akash (Team Lead)technical blog post documenting methodology, profiling data, and results.

Technical Skills

Languages: Python, C++, CUDA, Triton, HIP/ROCm · Go (exposure)

Inference Stack: vLLM, TensorRT-LLM, Triton Inference Server, Ray Serve, PyTorch (`torch.compile`)

Kernel-Level Optimization: CUDA, Triton, CuTe; Nsight Systems/Compute (`nsys/ncu`), inter-kernel tracing

Inference Internals: Prefill vs. Decode, KV-cache & Paged Attention, CUDA Graphs, Continuous Batching, Speculative Decoding

Hardware & Infra: NVIDIA B200/H100/A100, AMD MI300X · Kubernetes, dstack, FastAPI

Observability: Prometheus/Grafana, OpenTelemetry, `ncu/nsys` dashboards