Nils Matteson

linkedin.com/in/nilsmatteson nilsmatteson.com nilsmatteson@icloud.com github.com/matteso1

CS & Data Science Senior building **ML systems from GPU kernels to production**—speculative decoding with custom Triton kernels, distributed streaming in Go, and RAG on AWS. Seeking ML Infrastructure or Research Engineering roles.

Education

University of Wisconsin–Madison Madison, WI
B.S. Data Science, Minor in Computer Science *Expected May 2026*

– **Relevant**: Big Data Systems, Artificial Intelligence, Causal Inference, Machine Organization, Linear Algebra, Calculus I–III, Data Science Modeling I & II, Discrete Math.

Technical Skills

Languages: Python, Rust, Go, C++, SQL, TypeScript/JavaScript
ML & Systems: PyTorch, Triton, CUDA, Hugging Face, XGBoost, LLM Inference, Quantization (NF4/GPTQ), RAG
Infrastructure: AWS (Bedrock, S3), Docker, K8s, gRPC, Kafka, Redis, PostgreSQL, Git, Linux, CI/CD

Experience

Research Cyberinfrastructure, UW–Madison DoIT Madison, WI
AI Workflows Research Collaborator *Jan 2026 – Present*

– Benchmarked **10 LLMs on AWS Bedrock** (Claude, Llama, DeepSeek R1, GPT-OSS) across 282-question sustainability Q&A dataset; designed **weighted scoring framework** with Jaccard ref-overlap and NA recall metrics to expose real model differentiation.
– Built full **Bedrock integration layer**—async inference with retry/backoff, per-model inference profiles for cost attribution, and HTTP-header token extraction to fix vendor-specific tracking gaps (DeepSeek R1).
– Performed **Pareto cost-efficiency analysis**; discovered GPT-OSS 120B matches 97% of top-model accuracy at 95% lower cost ($0.51 vs $10.91), directly informing production model selection.
– Presented findings at **UW–Madison ML+X Forum** (Feb 2026): demonstrated cloud (AWS Bedrock) vs. local (open-source on GB10-class hardware) RAG deployment tradeoffs to cross-campus ML practitioners.

Selected Projects

Project Gorgon: LLM Inference Acceleration via Speculative Decoding *Python, PyTorch, Triton, CUDA*
Medusa-style speculative decoding engine for Llama-3-8B with custom GPU kernels and adaptive tree search. Technical writeup on nilsmatteson.com.

– Trained **5 Medusa draft heads** on frozen 4-bit Llama-3-8B via self-distillation on 200K conversations with per-head loss weighting ($\lambda_k = 0.8^k$), cosine LR schedule, and identity-initialized residual blocks.
– Wrote **fused Triton kernel** for tree-structured verification that eliminates the $O(N^2)$ attention mask by walking a parent-pointer array in registers—reducing verification overhead vs. materialized mask approach.
– Diagnosed and fixed **train–test hidden-state mismatch**: forward hook captured post-norm hidden states while training used pre-norm, causing double-RMSNorm at inference; added zero-init residual blocks for identity-initialized baseline.
– Built **adaptive tree pruning** with entropy-weighted confidence and path-probability thresholds to dynamically reduce candidate tree size, cutting wasted verification compute on low-confidence branches.

Madison Metro ML: Autonomous Bus Arrival Prediction *Python, XGBoost, Sentinel, PostgreSQL, React*
16K+ LOC end-to-end ML system with ground truth generation, autonomous retraining, and live inference.

– Designed **geospatial ground truth pipeline**: Haversine-based GPS-to-stop matching (30m threshold) joined to predictions for error computation; built streaming ingest via Sentinel (custom MQ) with gRPC and PostgreSQL.
– Implemented **autonomous nightly retraining** via GitHub Actions: XGBoost on rolling 7-day window with metric-gated deployment and Git-versioned model registry; deployed on Railway + Vercel with live MapLibre dashboard.

Sentinel: Distributed Log Streaming Engine *Go, gRPC, Protobuf, LSM Trees, Raft*
Kafka-inspired message queue (5,600+ LOC) powering Madison Metro ML's real-time data pipeline.

– Engineered custom **LSM-tree storage engine** with skip list memtable achieving 1.7M writes/sec and 3.9M reads/sec; implemented CRC32 checksums, bloom filters, and crash-safe write-ahead log.
– Built **Raft consensus layer** for fault-tolerant leader election and log replication; designed gRPC streaming API with topic/partition semantics, consumer groups, and offset tracking.