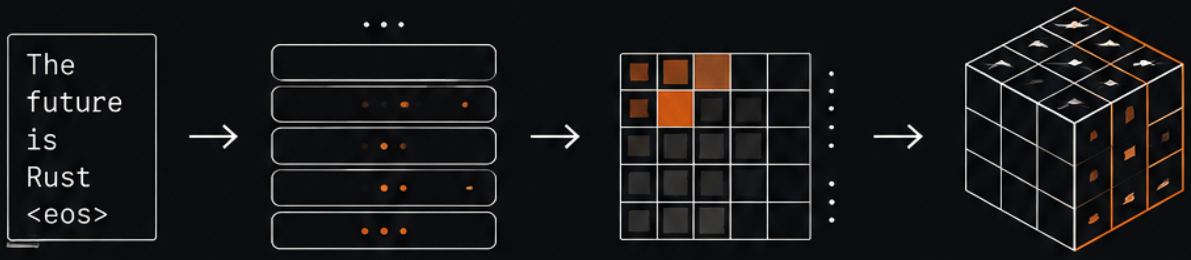
RUST FOR LLM INFERENCE

BUILDING HIGH-PERFORMANCE LLM INFERENCE ENGINE FROM SCRATCH



STEVE T.

Rust for LLM Inference

Building High-Performance LLM Inference Engine from Scratch

Steve T. Team Publications

This book is available at https://leanpub.com/rustforllminference

This version was published on 2026-07-03

Contents

Building High-Performance LLM Inference Engine from Scratch

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rustforllminference.

About This Book

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rustforllminference.

Introduction: Why Rust for LLM Inference

Imagine you are designing a high-performance web server. You reach for Rust because it gives you C-like speed with memory safety guarantees, fearless concurrency through its ownership model, and an ecosystem of battle-tested async libraries. Now imagine you are building the inference engine that powers a language model serving thousands of concurrent requests. The same forces apply. Inference is fundamentally a systems problem: massive tensor computations must move through memory hierarchies with minimal latency, dozens of concurrent requests must be scheduled and batched efficiently, and the whole system must run reliably under variable load.

The LLM inference landscape in 2026 looks very different from just three years ago. Python-based frameworks dominated the early days: HuggingFace's `transformers` library for prototyping, NVIDIA's TensorRT-LLM and FairSeq for production serving. Then came vLLM with PagedAttention and continuous batching, which fundamentally changed what was possible in throughput [15]. TGI (Text Generation Inference) from Hugging Face brought a Rust-based router to the Python inference server [28]. llama.cpp pioneered CPU inference with quantized models, enabling LLMs to run on laptops and edge devices [12].

Rust has emerged as the language of choice for the infrastructure layer of this ecosystem. The llama.cpp project, written in C and C++, spawned a wave of Rust implementations. Hugging Face open-sourced Candle, a minimalist Rust ML framework [5]. Burn provides a full deep learning framework in pure Rust with CubeCL for hardware-agnostic GPU kernels [34]. Mistral.rs delivers a high-performance inference engine that supports multimodal models [29]. TGI's router is itself a Rust binary [28]. The pattern is clear: where you need low-latency networking, safe concurrency, and fine-grained memory control, Rust is the answer.

The Inference Engine We Will Build: llm-rs

Throughout this book, we will build a complete inference engine called `llm-rs` from scratch. Each chapter adds a new component to the same project:

- **Chapter 1** defines our model configuration and tensor types.
- **Chapter 2** implements a tokenizer module that converts text to token IDs.
- **Chapter 3** builds attention layers with support for causal masking.
- **Chapter 4** assembles a complete transformer forward pass and the autoregressive generation loop.
- **Chapter 5** adds a KV cache system with PagedAttention-style block management.
- **Chapter 6** implements INT4 and INT8 quantization with dequantize-and-multiply kernels.
- **Chapter 7** designs a continuous batching scheduler.
- **Chapter 8** optimizes the CPU path with SIMD and parallelism.
- **Chapter 9** adds CUDA kernel integration for GPU acceleration.
- **Chapter 10** implements model loading from safetensors and GGUF formats.
- **Chapter 11** scales to multi-GPU tensor parallelism.
- **Chapter 12** wraps everything in a production-ready HTTP server with observability.

By the end, you will have a working inference engine that can load a decoder-only transformer, tokenize input, run attention and feed-forward layers, manage KV caches, and serve completions over HTTP. It is not intended to outperform vLLM or TensorRT-LLM in raw throughput (those systems have years of engineering investment), but it will teach you exactly how every piece works.

The LLM Serving Landscape: A Competitive Ecosystem

To understand why Rust matters for inference, consider the landscape:

vLLM remains the de facto standard for high-throughput serving [15]. It introduced PagedAttention, which maps KV cache allocations to fixed-size blocks in GPU memory, reducing fragmentation to under 4 percent. The original SOSP 2023 paper demonstrated that vLLM achieves 2–4× higher throughput

than leading systems like FasterTransformer and Orca at equivalent latency [15]. Later benchmarks showed up to 23x throughput improvement over naive static batching when combining PagedAttention, continuous batching, and prefix caching [21]. vLLM supports tensor parallelism, quantization (GPTQ, AWQ, FP8), and an OpenAI-compatible API. It is written in Python with CUDA kernels.

TGI (Text Generation Inference) from Hugging Face uses a three-tier architecture: a Rust launcher for process orchestration, a Rust router for HTTP/gRPC request handling, and a Python server for model inference [28]. TGI v3.0, released in December 2024, delivers significant speed improvements on long prompts (up to 13x over prior versions for prompts exceeding 200,000 tokens) by using an optimized prefix caching mechanism that can triple the token handling capacity while reducing memory footprint [4]. TGI supports tensor parallelism, FlashAttention, dynamic batching, and quantization (GPTQ, AWQ, NF4, FP8). As of late 2025, Hugging Face has placed TGI in maintenance mode while focusing on inference endpoints as a managed service, but its architectural innovations remain influential across the ecosystem [28].

llama.cpp pioneered CPU inference with quantized models and remains the dominant format for local/consumer inference. Its GGUF format supports a wide range of quantization schemes from FP16 down to 2-bit. llama.cpp now also provides comprehensive GPU support via CUDA, ROCm, Metal, and Vulkan [12]. NVIDIA's internal benchmarks show that on a GeForce RTX 4090, a Llama 3 8B model running through llama.cpp achieves approximately 150 tokens per second with 100-token input and output sequences [12].

Burn is a next-generation deep learning framework in pure Rust that does not compromise on flexibility, efficiency, or portability. Its CubeCL GPU compute language lets you write kernels once in Rust and run them on CUDA, ROCm, Metal, Vulkan, and WebGPU. Burn 0.19 introduced a CPU backend powered by MLIR and LLVM with JIT compilation and autotuning, bringing the same optimization capabilities from GPU backends to CPU execution [34].

Mistral.rs is a high-performance Rust-native inference engine supporting text, vision, video, audio, speech, and image generation models. It supports PagedAttention, ISQ (Integer Scalar Quantization), GGUF loading, continuous batching, FlashAttention V2, device offloading, and X-LoRA MoE non-granular scalings. Notably, it is written entirely in Rust using the Candle framework and requires no Python dependencies [29].

SGLang introduces RadixAttention for KV cache reuse in a radix tree structure, enabling automatic prefix caching across multiple generation requests and up to 5x faster inference for workloads with repetitive patterns [31]. It also uses compressed finite state machines for faster structured output decoding.

Why Rust? The Systems Argument

Three forces converge to make Rust the right choice for inference infrastructure:

Memory safety at scale. Inference servers run for weeks without restarts. A single use-after-free or data race in a C++ codebase can cause silent memory corruption that manifests hours later as incorrect outputs or crashes. Rust's ownership system guarantees that such bugs cannot exist in safe Rust. The llama.cpp ecosystem, while performant, has a history of memory safety issues in its more complex paths. Rust eliminates this class of bugs entirely.

Zero-cost concurrency. Serving thousands of concurrent requests requires fine-grained scheduling, lock-free data structures, and efficient thread pools. Rust's async runtime (Tokio) combined with its ownership model means you can write concurrent code that is both correct and fast. The TGI router, written in Rust, handles request scheduling and batching with sub-millisecond overhead [28]. Python's GIL makes similar concurrency patterns expensive or impossible without multiprocessing.

Predictable performance. Inference latency is user-facing. A 10ms delay in request processing compounds across a batch of requests. Rust has no garbage collector pauses, no GIL contention, and no interpreter overhead. Its predictable memory allocation patterns (pre-allocation, object pools, arena allocators) mean you can profile and optimize with confidence, knowing that a performance regression is a real optimization opportunity rather than a GC tuning problem.

How to Read This Book

This book assumes you are an experienced systems programmer or AI engineer. You should be comfortable with Python, have some exposure to C/C++, and understand the basics of neural network forward passes. Prior Rust experience is helpful but not required; we will explain idiomatic patterns as we go.

Each chapter builds on the last. Code examples are incremental: what you build in Chapter 3 is used in Chapter 4, which is used in Chapter 5, and so on. At the end of key chapters, you will find "Try It Yourself" exercises that extend the code and reinforce concepts.

The mathematical sections assume comfort with linear algebra (matrix multiplication, dot products) and basic calculus (derivatives for understanding softmax behavior). If you need a refresher, Appendix A provides a quick reference.

Framework Trade-offs: When Rust Makes Sense (and When It Does Not)

Before diving into implementation, it is worth being explicit about where Rust fits in the LLM stack and where it does not.

Where Rust excels. Inference serving is fundamentally a systems problem: massive tensor computations must move through memory hierarchies with minimal latency, dozens of concurrent requests must be scheduled and batched efficiently, and the whole system must run reliably under variable load. Rust's ownership model eliminates entire classes of bugs (use-after-free, data races) that plague C++ inference engines. Its zero-cost abstractions mean the high-level code compiles to performance competitive with hand-tuned CUDA kernels. The async runtime provides a clean path to concurrent request handling without thread pool complexity. For the infrastructure layer of LLM serving, Rust is hard to beat.

Where Rust is still catching up. The Rust ML ecosystem is young compared to Python's PyTorch/JAX or C++'s TensorRT-LLM. Key limitations include:

- **Ecosystem maturity.** Python has a decade of model development tools: HuggingFace Transformers, PEFT, Accelerate, DeepSpeed. Rust equivalents exist (Candle, Burn) but cover a smaller fraction of the model zoo. If you need to run an obscure architecture or experiment with a brand-new research paper published yesterday, Python is still the faster path.
- **CUDA integration friction.** Writing custom CUDA kernels in Rust requires build-time compilation of C++ code, separate linking, and careful FFI management. In Python, you write `.cu` files and import them with minimal ceremony. The `cudarc` and `cuda-build` crates are improving but lack the polish of PyTorch's autograd integration.

- **ROCm stability.** AMD GPU support is notably less mature across all Rust ML frameworks. Candle's ROCm backend has limited testing, Burn's CubeCL supports ROCm but with fewer optimizations than CUDA, and llama.cpp (the de facto standard for AMD) is C/C++, not Rust. If your deployment targets AMD Instinct GPUs, you will likely use llama.cpp directly or wrap it.
- **Pre-trained model availability.** The vast majority of open-source models are distributed as PyTorch `.bin` or safetensors files with Python-based conversion tools. While Rust can load these formats, the surrounding ecosystem (converting LoRA adapters, applying quantization, extracting embeddings) is predominantly Python-first.

The pragmatic approach. Most production systems use a hybrid strategy: Python for model development, experimentation, and fine-tuning; Rust for the serving layer where latency and throughput matter. TGI itself follows this pattern, with a Rust router and Python inference server. Mistral.rs demonstrates that a pure-Rust stack is possible but requires accepting a narrower model coverage and fewer pre-built utilities. The choice depends on your constraints: if you need maximum model coverage and rapid iteration, lean toward Python. If you need low-latency serving with strict reliability guarantees, Rust is the right call for the serving path.

When to use Python instead. You should reach for Python when:

- Prototyping a new architecture or research idea. Rapid iteration matters more than throughput.
- Fine-tuning models with complex custom training loops. PyTorch's autograd ecosystem is unmatched.
- Deploying on hardware with poor Rust toolchain support (certain edge devices, specialized ASICs).
- Your team's expertise is Python-first and the latency requirements are modest.

When to use C++ instead. You should reach for C++ when:

- You need the absolute maximum performance and have an existing C++ codebase. llama.cpp and TensorRT-LLM represent years of hand-tuned optimization.

- Your deployment environment has strict binary size constraints (embedded systems). A well-optimized C++ binary can be smaller than a Rust binary with the same functionality due to fewer runtime dependencies.
- You are building a GPU kernel library where CUDA C++ is the native language.

What This Book Is Not

This is not a tutorial on how to call an API. You will not find instructions for using OpenAI's chat completions endpoint. This is a deep dive into the internals, written in Rust, designed for engineers who want to build the next generation of inference infrastructure.

This is not a training book. We focus exclusively on inference (forward pass, generation). Training requires gradient computation, optimizer state management, and distributed data parallelism, which are important but distinct problems.

This is not a comparative framework review. We mention vLLM, TGI, Candle, Burn, llama.cpp, and Mistral.rs throughout to provide context and benchmarks, but the focus is always on building your own engine.

By the end of this book, you will be able to:

- Explain the transformer architecture from first principles and implement a forward pass in Rust.
- Understand tokenization algorithms and build a tokenizer.
- Implement attention mechanisms including FlashAttention-style tiling.
- Design and optimize KV caches for autoregressive generation.
- Apply quantization schemes (INT4, INT8, FP8) to reduce memory and accelerate inference.
- Implement continuous batching and PagedAttention-like memory management.
- Optimize CPU inference with SIMD vectorization and parallelism via rayon.
- Write CUDA kernels for GPU acceleration and integrate them into a Rust project.
- Load models from safetensors and GGUF formats.

- Design a distributed inference system with tensor parallelism.
- Deploy an inference server using axum with proper observability.

This is a deep dive into the internals, written in Rust, designed for engineers who want to build the next generation of inference infrastructure.

Chapter 1: The Transformer Architecture

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rustforllminference.

What a Transformer Is and Why It Works

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rustforllminference.

Encoder-Decoder vs Decoder-Only

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rustforllminference.

Layer-by-Layer Anatomy

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rustforllminference.

The Forward Pass as Computation Graph

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rustforllminference.

Model Configuration in Rust

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rustforllminference.

Try It Yourself: Building a Model Config

Chapter 2: Tokenization and the Vocabulary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rustforllminference.

Byte-Pair Encoding and Its Variants

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rustforllminference.

Building and Using Tokenizers in Rust

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rustforllminference.

Prompt Assembly and Token Counting

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rustforllminference.

Try It Yourself: Implementing a Simple BPE Tokenizer

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rustforllminference.

Try It Yourself: Building a Prompt Assembler

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rustforllminference.

Chapter 3: Attention Mechanisms Deep Dive

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rustforllminference.

Scaled Dot-Product Attention Derivation

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rustforllminference.

Multi-Head Attention and Head Pruning

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rustforllminference.

FlashAttention and I/O-Aware Computation

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rustforllminference.

Causal (Masked) Attention for Autoregressive Generation

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rustforllminference.

Rotary Position Embeddings (RoPE) Detail

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rustforllminference.

Try It Yourself: Implementing Scaled Dot-Product Attention

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rustforllminference.

Try It Yourself: Implementing RoPE

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rustforllminference.

Chapter 4: Building the Core Inference Loop

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rustforllminference.

Setting Up a Minimal Rust Project

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rustforllminference.

Implementing Linear Layers, RMSNorm, and Activations

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rustforllminference.

Composing a Complete Transformer Layer

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rustforllminference.

The Autoregressive Generation Loop

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rustforllminference.

Benchmarking a Single-Layer Forward Pass

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rustforllminference.

Try It Yourself: Building a Complete Inference Loop

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rustforllminference.

Try It Yourself: Sampling Strategies

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rustforllminference.

Chapter 5: KV Caching and Autoregressive Optimization

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rustforllminference.

Why Naive Autoregressive Inference Is $O(n^2)$ in Memory and Compute

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rustforllminference.

Key-Value Cache Design and Tensor Shapes

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rustforllminference.

PagedAttention: Memory Management as an Operating System Problem

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rustforllminference.

Implementing PagedAttention in Rust

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rustforllminference.

Cache Eviction Strategies

Try It Yourself: Implementing a Contiguous KV Cache

Try It Yourself: Implementing PagedAttention

Chapter 6: Quantization for Memory and Speed

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rustforllminference.

Floating-Point Formats: FP32, FP16, BF16, FP8

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rustforllminference.

INT8 and INT4 Quantization Schemes

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rustforllminference.

GGUF/GGML Format and llama.cpp's Approach

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rustforllminference.

Implementing Dequantize-and-Multiply in Rust

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rustforllminference.

Quantization Schemes Compared

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rustforllminference.

Try It Yourself: Implementing INT4 Quantization

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rustforllminference.

Try It Yourself: Parsing GGUF Files

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rustforllminference.

Chapter 7: Batching Strategies

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rustforllminference.

Static vs Dynamic Batching

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rustforllminference.

The Continuous Batching Scheduler

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rustforllminference.

Continuous Batching and PagedAttention

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rustforllminference.

Request Scheduling and Priority

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rustforllminference.

Priority Queue Scheduling

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rustforllminference.

Work-Stealing Scheduling for Multi-Threaded Prefill

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rustforllminference.

Scheduling Simulation and Benchmarking

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rustforllminference.

Prefix Caching and RadixTree Scheduling

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rustforllminference.

Throughput-Latency Tradeoffs

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rustforllminference.

Try It Yourself: Implementing a Scheduler

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rustforllminference.

Try It Yourself: Prefix Caching with RadixAttention

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rustforllminference.

Chapter 8: CPU Optimization Techniques

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rustforllminference.

SIMD Vectorization with Rust

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rustforllminference.

Cache-Conscious Memory Layout

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rustforllminference.

Parallelism with Rayon and Work-Stealing

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rustforllminference.

BLAS Integration for Matmul

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rustforllminference.

Benchmarking Methodology and Results

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rustforllminference.

Try It Yourself: SIMD Matrix Multiplication

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rustforllminference.

Try It Yourself: Rayon Parallel Matmul

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rustforllminference.

Try It Yourself: CPU Profiling with perf

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rustforllminference.

Chapter 9: GPU Acceleration with CUDA and ROCm

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rustforllminference.

Rust CUDA Bindings: Burn, Candle, llama.cpp

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rustforllminference.

Building a CUDA Backend for llm-rs

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rustforllminference.

Writing Custom CUDA Kernels

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rustforllminference.

Dequantize-and-Matmul Kernel

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rustforllminference.

FlashAttention Kernel Structure

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rustforllminference.

RMSNorm + Activation Fusion Kernel

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rustforllminference.

Memory Allocation Patterns for GPU Tensors

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rustforllminference.

Profiling with Nsight

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rustforllminference.

torch.compile-Style Kernel Fusion

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rustforllminference.

Try It Yourself: Writing a CUDA Kernel

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rustforllminference.

Try It Yourself: Implementing a Dequantize-and-Matmul Kernel

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rustforllminference.

Try It Yourself: GPU Memory Pool

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rustforllminference.

Chapter 10: Model Formats and Serialization

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rustforllminference.

PyTorch .bin and Safetensors Formats

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rustforllminference.

GGUF/GGML, ONNX, MLX, and TensorRT

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rustforllminference.

Serialization in Rust

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rustforllminference.

Weight Loading and Layout Transformation

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rustforllminference.

Loading GGUF Models in Rust

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rustforllminference.

Try It Yourself: Building a Model Loader

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rustforllminference.

Try It Yourself: GGUF Parser

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rustforllminference.

Chapter 11: Distributed Inference and Serving

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rustforllminference.

Tensor Parallelism vs Pipeline Parallelism

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rustforllminference.

Communication Patterns: All-Reduce and All-Gather

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rustforllminference.

Serving Frameworks: vLLM, TGI, Ollama, Mistral.rs

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rustforllminference.

Building a Production Rust Inference Server

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rustforllminference.

Try It Yourself: Building a Multi-GPU Engine

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rustforllminference.

Try It Yourself: Streaming API

Chapter 12: Production Deployment and Monitoring

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rustforllminference.

Containerization and Orchestration

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rustforllminference.

Docker Image Optimization

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rustforllminference.

Kubernetes Deployment

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rustforllminference.

Model Loading and Startup Optimization

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rustforllminference.

Observability: Metrics, Tracing, Logging

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rustforllminference.

Prometheus Metrics Implementation

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rustforllminference.

Grafana Dashboard Configuration

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rustforllminference.

Distributed Tracing with OpenTelemetry

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rustforllminference.

Structured Logging with Correlation IDs

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rustforllminference.

Log Aggregation and Alerting

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rustforllminference.

Fault Tolerance and Recovery

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rustforllminference.

A/B Testing New Models and Quantizations

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rustforllminference.

Cost Analysis: Dollars per Million Tokens

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rustforllminference.

Try It Yourself: Deploying with Docker

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rustforllminference.

Try It Yourself: Setting Up Prometheus Metrics

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rustforllminference.

Try It Yourself: A/B Testing Pipeline

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rustforllminference.

Appendix A: Benchmarking Methodology

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rustforllminference.

Hardware Specifications

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rustforllminference.

Software Versions

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rustforllminference.

Benchmarking Procedure

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rustforllminference.

Single-Layer Attention (Chapter 4)

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rustforllminference.

Full Inference (Chapter 8)

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rustforllminference.

Reproducing the Results

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rustforllminference.

Understanding Benchmark Numbers

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rustforllminference.

Statistical Rigor

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rustforllminference.

Conclusion: Where Rust Fits in the LLM Ecosystem

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rustforllminference.

The Trade-offs We Made

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rustforllminference.

Where Rust Shines (and Where It Does Not)

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rustforllminference.

The Pragmatic Stack

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rustforllminference.

The Future of Rust in LLM Inference

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rustforllminference.

What Comes Next

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rustforllminference.

References

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rustforllminference.