Author Name

Self-Hosted AI Inference

A Systems Engineer's Guide

December 13, 2025

Springer Nature

Contents

Chapter 1
Introduction to Self-Hosted Inference

Abstract This chapter introduces the concept of self-hosted AI inference, explaining why organizations and individuals choose to run their own models rather than relying on cloud APIs. We cover the fundamental differences between training and inference, the economic and strategic considerations for self-hosting, and provide a roadmap for what you'll build throughout this book. By the end of this chapter, you'll run your first local model and make your first inference request.

1.1 The Rise of Self-Hosted AI

The landscape of AI deployment shifted fundamentally in 2023 when Meta released Llama 2 under a permissive license [1]. For the first time, a model competitive with commercial APIs was available for local deployment. Other organizations followed: Mistral AI released their 7B model in September 2023 [2], Alibaba open-sourced the Qwen family [3], Microsoft published the Phi series optimized for efficiency [4], and Google contributed Gemma [5].

These models can run on consumer hardware. A 7 billion parameter model quantized to 4-bit precision requires approximately 4GB of VRAM—well within the capacity of a laptop GPU or even a MacBook's unified memory. The barrier to entry is no longer access to models or specialized hardware; it's the systems knowledge required to deploy and operate them reliably.

When you use a hosted API, the trade-offs are straightforward: you pay per token, accept the provider's latency (typically 50–200ms for first token), and send your data to external servers. When you self-host, those trade-offs invert. Your per-request costs drop to fractions of a cent after the initial hardware investment. Latency becomes a function of your local compute rather than network round-trips. Your data never leaves infrastructure you control.

The complexity trade-off is real. Self-hosting requires understanding GPU memory hierarchies, model quantization formats, inference engine architectures, and production

deployment patterns. This book provides that knowledge systematically, building from a single 7B model on a laptop to a distributed 400B deployment across multiple GPUs.

1.2 Training vs. Inference: Understanding the Difference

The terms ”training” and ”inference” describe two fundamentally different computational processes. Training creates a model by learning patterns from data. Inference uses that trained model to generate outputs for new inputs. The resource requirements differ by orders of magnitude, which is why you can run inference on a laptop but training typically requires a data center.

This book focuses on inference, not training. We won't derive backpropagation equations or explain attention mechanisms mathematically—other texts cover that well [6, 7]. Instead, we treat models as artifacts you download and deploy. The explanations below provide enough context to understand resource requirements and make informed infrastructure decisions. If terms like ”parameters” or ”gradients” are unfamiliar, the brief descriptions here should suffice; deeper understanding isn't required to build and operate inference systems.

1.2.1 What Happens During Training

Training a large language model involves adjusting billions of numerical parameters so the model can predict the next token in a sequence. The process works in three steps, repeated millions of times:

1. **Forward pass**: Input text flows through the network, producing a prediction for the next token.
2. **Loss calculation**: The prediction is compared to the actual next token, producing an error signal.
3. **Backward pass**: The error propagates backward through the network via gradient descent, slightly adjusting each parameter to reduce future errors.

The backward pass is the expensive part. It requires storing intermediate activations from the forward pass (consuming memory) and computing gradients for every parameter (consuming compute). For a 7B parameter model, this means computing and storing 7 billion gradients per training step.

Training Llama 2 70B required 1.7 million GPU hours on A100 80GB GPUs [1]. At current cloud prices of approximately $2/hour per A100, that represents over $3 million in compute costs—before accounting for failed experiments, hyperparameter tuning, and infrastructure overhead. Training also requires massive datasets: Llama 2 used 2 trillion tokens of text data.

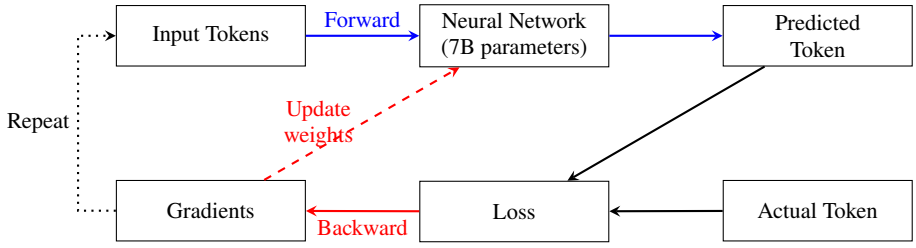

Fig. 1.1 The training loop. Each iteration performs a forward pass (blue) to generate predictions, computes the loss against actual targets, then propagates gradients backward (red) to update model weights. Training Llama 2 70B required approximately 1.7 million GPU hours of this loop.

1.2.2 What Happens During Inference

Inference is simpler: it runs only the forward pass. There's no loss calculation, no gradient computation, and no parameter updates. The model weights are fixed; you're simply using them to transform inputs into outputs.

For large language models, inference works autoregressively—the model generates one token at a time, feeding each generated token back as input to produce the next. Generating a 100-token response requires 100 forward passes through the network. This sequential dependency is why LLM inference can feel slow even on fast hardware: you cannot parallelize the generation of a single response.

The resource profile differs substantially from training:

- **Memory**: You need to store the model weights and a *key-value cache* (KV cache) that grows with context length. No gradient storage required.
- **Compute**: Forward passes only, with operations dominated by matrix multiplications between inputs and weights.
- **Data**: A single prompt, not terabytes of training data.

The KV cache deserves attention because it's often the limiting factor for long-context inference. During generation, the model caches intermediate computations (keys and values from the attention mechanism) to avoid redundant work. For a 7B model with a 4096-token context, the KV cache consumes approximately 1GB of memory at FP16 precision. At 128K context length, that grows to roughly 32GB—potentially exceeding the model weights themselves. Chapter **??** covers memory calculations in detail.

A single inference request on a 7B model takes 10–100 milliseconds for the first token (depending on prompt length) and 20–50 milliseconds per subsequent token on a modern consumer GPU. This is the timescale you'll work with throughout this book: milliseconds per token, not months per training run.

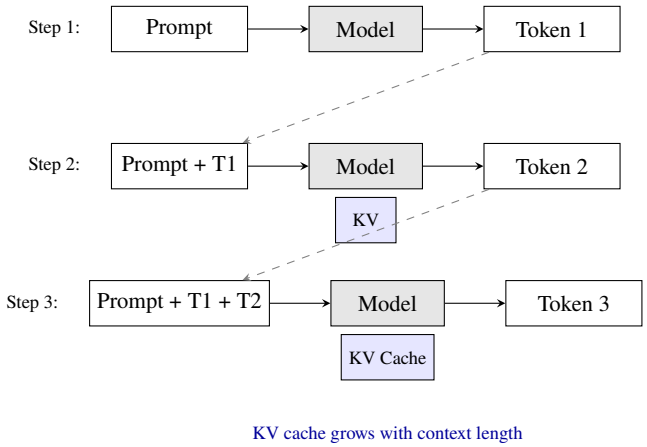

Fig. 1.2 Autoregressive inference. Each token is generated by a forward pass through the model, then appended to the context for the next iteration. The KV cache stores intermediate attention computations to avoid redundant work, growing with each generated token.

1.2.3 Resource Comparison

Table 1.1 summarizes the resource differences. The key insight: training is a one-time cost borne by model creators (Meta, Mistral, etc.), while inference is the ongoing cost you pay when deploying models. This book assumes someone else has already trained the model; your job is to run it efficiently.

Table 1.1 Resource requirements for training versus inference of a 70B parameter model.

Aspect	Training	Inference
Hardware	8+ A100 80GB GPUs	1–2 consumer GPUs
Time per run	Weeks to months	10–100 ms per token
Data required	Trillions of tokens	Single prompt
Memory usage	Weights + gradients + activations	Weights + KV cache
Compute cost	$1–10 million	$0.0001–0.01 per request
Who pays	Model creator	You (the deployer)

The implication is significant: you can leverage billions of dollars of training investment by downloading open-weight models and running inference on hardware you control. A $1,000 GPU can serve the same model that cost millions to train.

1.3 Why Self-Host?

Organizations and individuals choose self-hosted inference for four primary reasons: cost control, data privacy, latency requirements, and customization needs. The weight of each factor varies by use case. A hobbyist experimenting on weekends has different priorities than a healthcare company processing patient records.

1.3.1 Cost Control

API pricing follows a per-token model. As of Dec 2025, OpenAI's GPT-5 costs $1.75 per million input tokens and $14.00 per million output tokens [8]. Anthropic's Claude Sonnet 4.5 costs $3.00 input and $15.00 output per million tokens, while Claude Opus 4.5 runs $5.00 input and $25.00 output [9]. These costs accumulate quickly at scale.

Consider a customer support application handling 10,000 conversations per day, averaging 500 input tokens and 500 output tokens per conversation. At GPT-5 rates, that's approximately $56 per day ($8.75 input + $70 output), or $2,340 per month. For a coding assistant processing 100,000 requests daily with longer outputs, monthly API costs can reach $40,000–70,000 depending on the model tier.

Self-hosted costs work differently. The primary expenses are:

- **Hardware**: A single RTX 4090 ($3,000) can run a 7B model at 30–50 tokens/second, or a quantized 70B model at 5–10 tokens/second.
- **Electricity**: GPUs under load draw 300–400W. At $0.12/kWh, that's roughly $25–35/month for continuous operation.
- **Maintenance**: Minimal for small deployments; significant for multi-GPU clusters.

The break-even calculation depends on utilization. A $2,000 GPU investment pays for itself in 1–2 months if you're replacing $1,500/month in API costs. If your usage is sporadic—a few hundred requests per day—APIs may remain more economical. Chapter **??** provides detailed cost models for different deployment scales.

1.3.2 Privacy and Data Sovereignty

1.3.3 Latency and Performance

1.3.4 Customization and Control

1.3.5 When NOT to Self-Host

1.4 What You'll Build in This Book

1.4.1 The Progressive Journey

1.4.2 Control Plane Evolution

1.5 Hands-On: Your First Local Inference

1.5.1 Installing Ollama

1.5.2 Running Your First Model

1.5.3 Making API Requests

First Inference Request

```
# Pull a 7B model
ollama pull llama3.2:7b

# Make an inference request
curl http://localhost:11434/api/generate \
  -d '{
    "model": "llama3.2:7b",
    "prompt": "Explain what AI inference is in one paragraph.",
    "stream": false
  }'
```

1.6 Understanding the Response

1.7 Summary

Key Takeaways

- Inference is fundamentally different from training and accessible with consumer hardware
- Self-hosting offers cost control, privacy, and customization benefits
- Open models like Llama, Mistral, and Qwen make self-hosting viable
- You've run your first local inference with Ollama

Problems

1.1 Cost Comparison

Calculate the monthly cost of running 100,000 inference requests through OpenAI's GPT-4 API versus self-hosting a 7B model on an RTX 4060. Assume average request length of 500 tokens input and 200 tokens output.

1.2 Latency Measurement

Using Ollama, measure the time-to-first-token (TTFT) and tokens-per-second for three different prompts: (a) a simple question, (b) a code generation request, and (c) a creative writing prompt. What patterns do you observe?

1.3 Streaming vs Non-Streaming

Implement a simple Python script that makes both streaming and non-streaming requests to Ollama. Measure the perceived latency (time until user sees first output) for each approach.

References

1. Touvron, H., Martin, L., Stone, K., et al.: Llama 2: Open Foundation and Fine-Tuned Chat Models. arXiv preprint arXiv:2307.09288 (2023). https://arxiv.org/abs/2307.09288
2. Jiang, A.Q., Sablayrolles, A., Mensch, A., et al.: Mistral 7B. arXiv preprint arXiv:2310.06825 (2023). https://arxiv.org/abs/2310.06825
3. Bai, J., Bai, S., Chu, Y., et al.: Qwen Technical Report. arXiv preprint arXiv:2309.16609 (2023). https://arxiv.org/abs/2309.16609
4. Li, Y., Bubeck, S., Eldan, R., et al.: Textbooks Are All You Need II: phi-1.5 Technical Report. arXiv preprint arXiv:2309.05463 (2023). https://arxiv.org/abs/2309.05463

5. Gemma Team: Gemma: Open Models Based on Gemini Research and Technology. arXiv preprint arXiv:2403.08295 (2024). https://arxiv.org/abs/2403.08295
6. Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. MIT Press (2016). https://www.deeplearningbook.org/
7. Vaswani, A., Shazeer, N., Parmar, N., et al.: Attention Is All You Need. Advances in Neural Information Processing Systems 30 (2017). https://arxiv.org/abs/1706.03762
8. OpenAI: API Pricing. https://openai.com/api/pricing/ (accessed December 2025)
9. Anthropic: Claude API Pricing. https://www.anthropic.com/pricing (accessed December 2025)