LLM Cost Engineering: Scaling Production Systems Profitably

LLM-powered products are powerful but are expensive by default. This course teaches you how to measure, optimize, route, cache, and monitor LLM workloads so you can scale production systems profitably.-----Course Overview

This course is designed for developers and product teams and focuses on designing and building cost-efficient LLM systems from the ground up. Cost engineering is presented as the foundation of sustainable AI products, enabling teams to control token usage, match tasks to the right model tier, avoid repeated payments for similar requests, and maintain quality under budget constraints.

The curriculum covers practical patterns used in production, which you will implement step-by-step in Python and Jupyter. Topics covered include:

- Understanding the LLM cost landscape, including pricing for input, output, cached, and batch requests across providers.
- Diving into token economics to uncover hidden overhead in system prompts, tool schemas, and conversation history.
- Applying prompt compression techniques to reduce waste without sacrificing answer quality.
- Implementing capability-based model routing to send simple tasks to low-cost models and complex reasoning to premium models.
- Adding caching and batch processing to capture major provider discounts.
- Designing fallback and resilience patterns for outages and budget pressure.
- Evaluating fine-tuning ROI through break-even analysis and distillation logic.
- Wiring everything into a production monitoring stack with telemetry, dashboards, alerts, and optimization loops.

Prerequisites

This course is designed for developers.

- **Required:** Basic Python knowledge (comfortable with classes, functions, list and dictionary operations, and simple data analysis/basic pandas usage).
- **Recommended:** Curiosity about LLM systems, cost optimization, and production engineering trade-offs. Familiarity with LLM APIs is helpful but optional.
- **Not Required:** No prior cost engineering experience is needed.

Learning Outcomes

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

- Understand LLM pricing mechanics and calculate the cost of LLM calls to estimate spend for real workloads.
- Measure token consumption and identify hidden overhead to track Token Efficiency Ratio (TER) across features.
- Implement prompt compression strategies (e.g., manual distillation, rolling summaries) to reduce token spend by 40–70% while preserving response quality.
- Design and implement model routing pipelines across model tiers that align task complexity with model cost.
- Apply exact-match, semantic, and provider-native caching to avoid repeated spend and capture 50–90% discounts.
- Use batch APIs and hybrid queues for non-urgent workloads at discounted rates.
- Implement fallback chains, circuit breakers, and budget-aware degradation logic for reliability.
- Evaluate fine-tuning economics with break-even and total cost of ownership analysis (TCO/ROI).
- Build production telemetry, dashboards, and alerts for continuous cost optimization.

Detailed Course Syllabus

Chapter	Content	Hands-On
Chapter 1	The LLM Cost Landscape - provider pricing anatomy (input/output/cached/batch); model pricing spread; real-world cost drivers; unit economics basics.	Build a multi-provider cost calculator, compare 10+ models, estimate monthly spend for 3 workloads, and visualize pricing sensitivity.
Chapter 2	Token Economics - tokenization differences across models; hidden token overhead; system prompt inflation; reasoning token behavior; TER baseline.	Implement a token meter, audit prompt overhead, build a TokenBudget utility, and benchmark TER across multiple feature types.
Chapter 3	Prompt Compression - manual distillation, few-shot pruning, structured output constraints, rolling summaries, and cheap-model pre-compression.	Compress a real prompt by 40–60%, run before/after quality checks, and produce a cost-quality trade-off matrix with savings projections.
Chapter 4	Model Routing - tiered capability framework; rule-based, embedding-based, and LLM-based routing; escalation and fallback economics.	Classify 20 tasks by complexity, implement a ModelRouter with 3 strategies, simulate routing savings, and test cascade behavior.
Chapter 5	Caching Strategies - provider-native cache usage, exact-match cache, semantic cache, cache hit-rate optimization, and cache ROI modeling.	Build exact and semantic caches with TTL, simulate repeated traffic, measure hit/miss rates, and quantify monthly savings.
Chapter 6	Batch Processing - async batch APIs, JSONL job preparation, hybrid real-time vs. delayed queues, and batch monitoring patterns.	Create batch payloads, route eligible traffic to batch, simulate 50K-query split, and calculate cost impact versus fully real-time.

Chapter 7	Fallback & Resilience - multi-model fallback chains, retry policies, circuit breaker design, and budget-threshold degradation.	Implement a FallbackChain and CircuitBreaker, simulate outages and budget pressure, and log failover behavior with cost impact.
Chapter 8	Fine-Tuning Economics - break-even math, distillation pipelines, 1-year TCO modeling, and ROI decision criteria.	Compute break-even curves for multiple scenarios, design teacher-student distillation flow, and compare alternatives to fine-tuning.
Chapter 9	Production Monitoring - telemetry schema design, cost decorators, SQLite logging, dashboards, alerting thresholds, and optimization loops.	Log 10K+ simulated calls, build feature/model cost dashboards, trigger anomaly alerts, and run detect → fix → measure optimization cycles.