Cloud Contraptions
TRAINING & CONSULTING

To discuss this course and customizations:
Call: +1 434-509-6890 or Email: sales@cloudcontraptions.com

Faster Responses from LLMs: Latency and Throughput Optimization

Class Duration

14 hours of live training delivered over 2 days.

Student Prerequisites

- Professional software development experience
- Familiarity with LLM API usage and running LLM-powered features in an application

Target Audience

Software engineers, platform engineers, and MLOps practitioners who need LLM-powered applications to feel fast and scale cost-effectively. Particularly relevant for teams whose response times are hurting user experience, whose costs climb under load, or who are moving from a prototype to a latency-sensitive production workload. Teams that also need deeper cost attribution and tracing can pair this with LLM Observability and Cost Engineering.

Description

LLM applications are slow and expensive by default, and the fixes live at three different layers. This two-day course teaches a practical, measure-first approach to making responses faster and throughput higher without sacrificing quality. Participants learn to define and measure the metrics that actually matter, time to first token, inter-token latency, and total response time, and then work through the optimization stack: application-level techniques (streaming, prompt and prefix caching, semantic caching, context compression, and model routing), system-level acceleration (continuous batching, paged attention, speculative decoding, and KV-cache quantization), and model-level options (smaller, distilled, and quantized models). The course closes with perceived-performance design, latency budgets, load testing, and regression monitoring so gains hold up in production.

To discuss this course and customizations:
Call: +1 434-509-6890 or Email: sales@cloudcontraptions.com

Learning Outcomes

- Define and measure the latency metrics that matter: time to first token, inter-token latency, and total response time.
- Apply application-level techniques: streaming, prompt and prefix caching, semantic caching, and context compression.
- Route requests across models and tiers to balance speed, cost, and quality.
- Use provider-native features such as prompt caching and batch APIs to cut latency and cost.
- Reason about system-level acceleration: continuous batching, paged attention, speculative decoding, and KV-cache quantization.
- Choose model-level options, including smaller, distilled, and quantized models, for latency-sensitive paths.
- Design for perceived performance with streaming UX, partial results, and graceful degradation.
- Establish latency budgets, load tests, and regression monitoring.

Training Materials

Comprehensive courseware is distributed online at the start of class. All students receive a downloadable MP4 recording of the training.

Software Requirements

Python 3.12+ or Node.js 22+, API keys for at least one frontier model, a load-testing tool (k6 or Locust), optional access to a self-hosted inference server (such as vLLM) for the system-level topics, and Git.

Training Topics

Latency Fundamentals

- Time to first token, inter-token latency, and total time
- Throughput versus latency, and why they trade off
- Where time actually goes in an LLM request
- Setting realistic targets per use case

Measuring and Budgeting Latency

- Instrumenting requests for latency and token metrics
- Percentiles that matter: p50, p95, and p99
- Establishing latency budgets per feature

- Baselining before optimizing

Streaming and Perceived Performance

- Streaming responses to cut time to first visible output
- Partial results, progress, and optimistic UI
- Cancellation and backpressure handling
- Graceful degradation under load

Prompt and Prefix Caching

- Provider-native prompt caching
- Prefix caching and reusing computed attention
- Cache key design for parameterized prompts
- Measuring cache hit rates and savings

Semantic Caching

- Caching responses by query meaning
- Embedding similarity and thresholds
- Freshness, invalidation, and correctness risks
- When semantic caching helps and when it misleads

Model Selection and Routing

- Matching model size and tier to the task
- Routing fast and cheap versus deep and slow
- Cascades and fallback chains
- Quality guardrails when downtiering

Model-Level Optimization

- Smaller and distilled models for latency-sensitive paths
- Quantization and its quality tradeoffs
- Task-specific and fine-tuned alternatives
- Benchmarking quality after optimization

System-Level Acceleration

- Continuous batching and paged attention
- Speculative and self-speculative decoding
- KV-cache quantization and concurrency
- Self-hosted inference servers and when to use them

To discuss this course and customizations:
Call: +1 434-509-6890 or Email: sales@cloudcontraptions.com

Context and Token Reduction

- Prompt compression and context trimming
- Retrieval to shrink prompts instead of stuffing them
- Output length control
- Structured outputs to reduce wasted tokens

Load Testing and Regression Monitoring

- Load testing LLM endpoints realistically
- Detecting latency regressions in CI and production
- Alerting on p95 and p99 drift
- Capacity planning and cost-versus-speed tradeoffs

