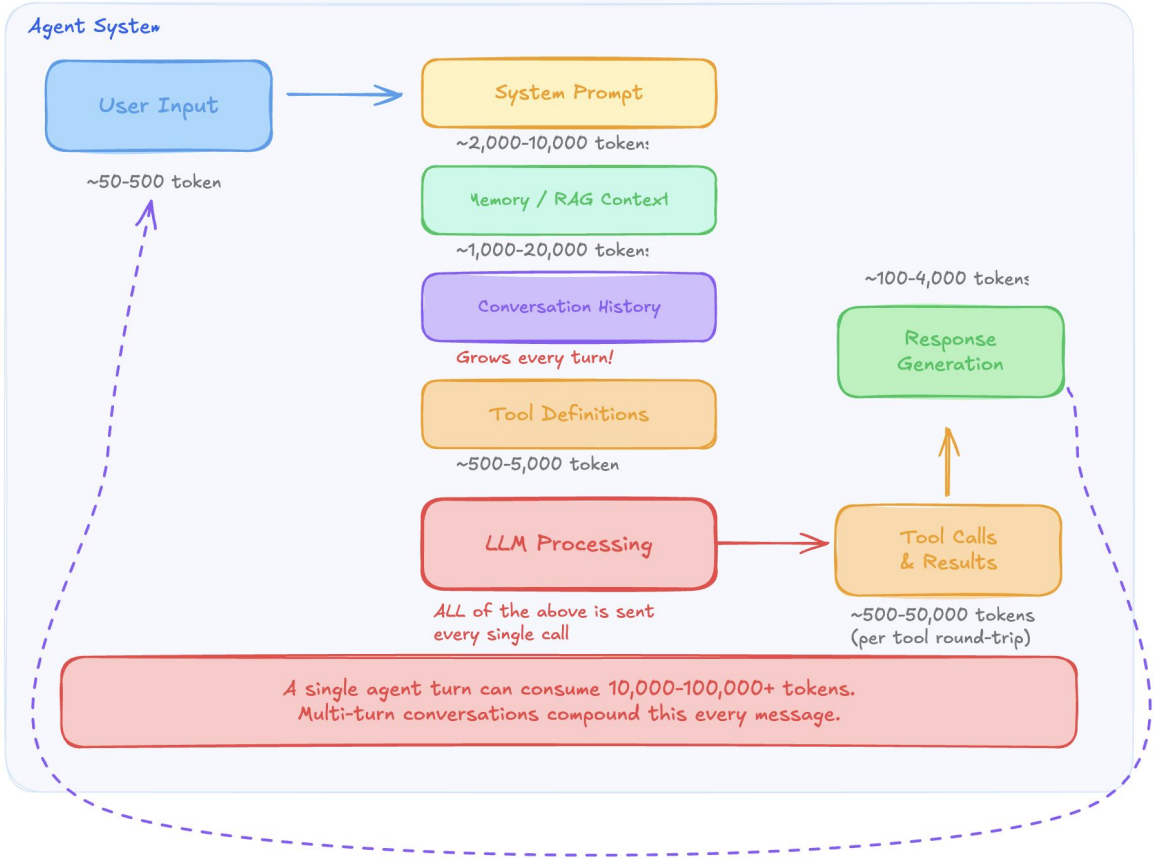
Token Usage Optimization with AI Agents

Being Good Stewards of Tokens

cirrius solutions

Token Flow Architecture

Where tokens accumulate as a request moves through an agent system



Being Good Stewards of Tokens

Why this matters regardless of your LLM provider

Subscription Model

Fixed monthly cost, but usage caps exist. Inefficient skills burn through your plan in days instead of weeks. Claude Pro, ChatGPT Plus, Copilot -- all have limits.

Usage-Based / API Mod

Pay per token. Every wasted token is wasted money.

The Core Problem

Most agent frameworks resend the ENTIRE context on every LLM call. System prompts, tool definitions, conversation history, memory -- all of it.

Who Needs This?

Everyone. Whether you're on Claude Max, OpenAI API, or running Gemma locally -- understanding token flow is the difference between a sustainable AI practice and one that gets shut down.

Inefficient Skills Are the Silent Killer

Good skill development isn't just about capability -- it's about efficiency. Every token you don't send is money you don't spend.

The Optimization Tiers

Layer these strategies for compounding savings

Tier 1: Prompt & Context Optimization

Cost: Free

Zero infrastructure changes. Better prompts, leaner context, smarter skill design.
The foundation everything else builds on. Saves 20-40% immediately.



Tier 2: Prompt Caching

Cost: Config only

Cache static context (system prompts, tool defs) so you're not re-processing them every call.
10x cheaper context within the 5-minute TTL window. Massive savings on repetitive calls.



Tier 3: Tiered LLM Routing

Cost: Architecture

Use small models for simple tasks, big models only when needed. Heartbeat agents on
Gemma/Haiku for monitoring, escalate to Sonnet/Opus for complex reasoning.



Tier 4: Nuclear -- Self-Hosted LLMs

Cost: Infrastructure

Host your own models. Shift costs from per-token API fees to infrastructure.
Maximum control, maximum responsibility. Best for high-volume or privacy-critical workloads.

Tier 1: Prompt & Context Optimization

Zero cost. Zero infrastructure. Immediate results.

Efficient Skill Design

Load only the context a skill actually needs. A skill that handles case resolution doesn't need your full org's knowledge base.

Progressive Disclosure

Don't front-load everything. Start with minimal context, let the agent request more only when needed.

Step 1: System prompt + user query
Step 2: Agent decides what tools/docs
Step 3: Load only what's relevant

Context Compaction

Summarize older conversation turns instead of sending full verbatim history. A 20-turn conversation can be compacted to key decisions and facts.

Sliding Window Strategy

Keep only the last N turns in full detail. Older turns get summarized or dropped.

Prevents quadratic cost growth.

Tool Definition Pruning

Every tool definition is sent with every call.
Only register the tools relevant to the current task. A calendar-checking agent doesn't need Salesforce deploy tools.

Combined Tier 1 savings: 20-50% reduction
with no code changes beyond prompt tuning

Tier 2: Prompt Caching

10x cheaper context calls within the 5-minute TTL window

How It Works

The LLM provider caches the static prefix of your prompt (system instructions, tool definitions, reference docs). On subsequent calls within the TTL, those cached tokens are read from memory instead of being re-processed.

First call: Full price (cache write)
Next 5 min: 90% discount on cached portion (cache read)
After 5 min: Cache expires, full price again

What to Cache (Static Prefix)

System prompt / instructions
Tool definitions (schemas)
Reference documentation
Framework context
Stable conversation summaries

What Changes (Dynamic Suffix)

User's current message
Latest tool results
Recent conversation turns
Real-time data lookups

The 5-Minute TTL Trap

Cache expires after 5 minutes of inactivity (5-10 for OpenAI). If your agent is called sporadically (once every 10 min), you pay full price every time. Design for cache warmth: batch operations, keep heartbeats alive, or accept the miss and optimize other tiers instead.

Tier 3: Tiered LLM Routing

Right-size the model to the task. Stop using a cannon to swat flies.

The Problem: Default Escalation

Without explicit model routing, most agent frameworks default to the most capable (and expensive) model for every task. Checking if your calendar has a meeting shouldn't cost the same as writing a technical architecture document.



Key insight: Explicitly set the model per task.
Don't let the framework default to the most expensive option.

Tier 4: Nuclear -- Self-Hosted LLMs

Shift costs from per-token API fees to owned infrastructure

What This Mean:

Run open-source models (Llama, Mistral, Gemma, DeepSeek) on your own hardware or private cloud.
Tokens become "free" at the API level -- you pay for compute, not per-token.

What You Gain

Zero per-token costs at scale
Tokens are effectively "unlimited"

Full data privacy and sovereignty
Nothing leaves your infrastructure

No vendor rate limits or outages
You control availability

Customizable / fine-tunable models
Train on your domain data

What You Take O

Extensions

GPU infrastructure costs
Hardware or cloud GPU rental

Model quality gap vs. frontier
Open-source trails Opus/GPT-4.5

Maintenance and operations burden
Updates, monitoring, scaling

No "magic updates" from providers
You own the upgrade cycle

When It Makes Sens

High volume: Your API bill exceeds what infrastructure would cost.
Privacy critical: Healthcare, finance, government -- data cannot leave your environment
Specialized tasks: Fine-tuned models outperform general-purpose for narrow domains

Bottom Line

Every team building with AI needs a token strategy



Rules of Thumb

1. Start with Tier 1. It's free and it works. Audit your prompts and skills first.
2. Add Tier 2 caching if your agent makes repeated calls with stable context.
3. Add Tier 3 routing when you have distinct task complexity levels.
4. Consider Tier 4 (nuclear) when API spend is high

Tokens are the new bandwidth. Just like we learned to optimize network calls, compress payloads, and cache responses -- we need to bring that same discipline to how we spend tokens. Be good stewards. Your budget will thank you.