JULY 1, 2026

Why your AI token costs keep rising even as prices fall

Per-token prices are collapsing, but the bill climbs anyway. The reason is that modern agentic models spend far more tokens to finish the same job.

By **Dave Taylor**

Your AI token costs keep going up because you are paying per token but the models now spend far more tokens to finish the same job. The unit price is genuinely falling — inference prices have dropped a median of roughly 50x per year, and as fast as 200x in a single year for some models, per Epoch AI's tracking of LLM inference price trends. But a 2026 agentic model reads context, plans, calls tools, retries, and checks its own work before it answers. It burns tokens a 2024 chatbot never touched. Cheaper tokens, more of them per task — and the bill climbs.

Why does the bill climb when token prices are falling?

The two numbers move in opposite directions, and the one you feel on the invoice is total spend, not unit price. That's the whole confusion in one sentence: the price you read about is falling while the amount you pay is going up, and both are true at once. The fall in unit price is genuine and steep — Epoch AI puts it at "rapidly but unequally," a median near 50x per year across models, with the fastest dropping about 900x and the slowest around 9x, per Epoch AI's inference price data. If a task cost the same number of tokens in 2026 as it did in 2024, your bill would be a rounding error against two years ago.

It isn't, because the number of tokens per task has gone the other way — and it has gone further than the price has fallen. The shift is in what a model does before it replies. In 2024 you sent a prompt and got an answer in one pass. In 2026 an agent plans the task, pulls documents into context, calls tools, reads the results, corrects itself, and often runs the loop several times over. Each of those steps is tokens, input and output, metered and billed. You asked it to do a whole job, not answer a question — so a single request now carries a token trail many times longer than the equivalent request two years ago. Multiply that longer trail by many more requests as agents take on more work, and cheaper tokens still add up to a larger invoice at the end of the month.

Priced per token, paid per result

You are billed per token, but the business gets value per accepted result. Those are not the same unit, and the gap between them is where cost decisions go wrong. A token is an input to the work. An accepted result — a drafted email that actually goes out, a support reply that resolves the ticket, a lead qualified correctly — is the output the business pays people to produce. When a finance lead asks "what does our AI cost," the honest answer is cost per finished, usable piece of work, not cost per million tokens on a pricing page.

Look at what a pricing page actually shows and the trap gets clearer. Anthropic lists Claude Opus 4.8 at $5 input and $25 output per million tokens, Claude Sonnet 5 at a $2/$10 introductory rate through 31 August 2026 (standard $3/$15), and Claude Haiku 4.5 at $1/$5, per Anthropic's Claude Sonnet 5 pricing announcement. Read that as a shopping list and the cheapest model wins every time — Haiku is a fifth the output price of Opus, so route everything to Haiku and pocket the difference. That logic breaks the moment you count tokens instead of price. The sticker only tells you what a token costs, not how many the model spends to reach a result you can use. A model that costs a fifth per token but takes four passes to land the answer is not cheaper — it's the same job billed four times at a lower rate. It only looks cheap on the page, and the page is the one place the real cost never shows up.

Where the money actually leaks

The leak is almost never the headline price. It's the tokens a cheaper model spends getting to an answer you'll actually accept, plus the human time it takes to fix what the model got wrong. A smaller or lower-tier model often reasons longer, retries more, and needs more correction to reach the same finished output as a stronger one. Each retry is a fresh round of input and output tokens, billed in full. Each human fix is a person's time — the most expensive token in the stack, and the one that never appears on the API bill at all. So the model that looked cheapest on the pricing page can quietly become the most expensive per finished job. Route work to it by default and you can pay more per accepted result than a pricier model would have cost, because you paid for the same answer three or four times over before anyone shipped it.

Two levers move real money here, and both are underused. The first is model routing — sending simple work to a small model and hard work to a large one, instead of running everything through a single model. The second is caching — storing prompts and responses so the model doesn't re-read the same context on every call. Model routing can cut cost 30 to 60% and caching 50 to 90% on the repeated portion, per CloudZero's analysis of inference cost; those are a vendor's figures, so treat them as directional, not neutral research. The reason both help is structural, not clever: most token spend is repetition and mis-routing, not the one call that genuinely needed the frontier model. A support agent re-reading the same 4,000-token policy document on every ticket is paying for that context hundreds of times a day. Fix the routing and the caching and the token trail shrinks before you ever touch the model choice itself.

How to run AI spend like your cloud bill

Treat AI spend the way a disciplined team treats its cloud bill — as something you instrument, attribute, and review, not a flat line you notice at renewal. Nobody runs cloud infrastructure without knowing which service costs what and why. AI inference is the same class of problem: a metered utility that quietly grows with usage until someone measures it.

This is the part we do for clients. CxD ships and runs AI agents for SMBs — lead generation, sales operations, custom internal agents — and we instrument each one for cost from the first day it goes live, not after the invoice surprises someone. The number we track is cost per accepted result, not cost per token, because that's the number that tells you whether the agent is earning its keep. Cost per token tells you almost nothing on its own: a low number can hide a model that retries endlessly, and a high one can belong to the model that finishes the job first time. The teams that keep AI spend under control measure the

unit of work the business actually cares about, catch the runaway loops early, and route deliberately. The teams that don't get a surprise invoice and a model nobody trusts enough to use. Track the right unit and the operating decisions get simple:

→ Route each task to the smallest model that clears the quality bar → cache the context that repeats → measure cost per accepted result, and let a runaway retry loop show up as a cost spike, not a mystery.

The payoff is real once the instrumentation is in place. On a recent recruitment-and-staffing engagement, an agent we built and ran lifted inbound calls 37% over an 8-week pilot — and because it was instrumented for cost per accepted result from day one, we could prove the spend was buying qualified conversations, not burned tokens. That's the difference between an AI line item a finance lead defends at budget time and one they quietly kill because nobody can say what it bought. When you can point at a per-result cost and the outcome it produced, the spend defends itself. Most AI projects that stall do so for reasons upstream of the model, which is the same pattern behind why AI reliability is set by the workspace, not the model: the work that decides the outcome happens before and around the model, not inside it. The readiness and build work we run puts that instrumentation in from the start, so the cost question has an answer the day the agent goes live.

What it adds up to

Falling token prices are real, and they're not the number that decides your AI bill. The bill is set by how many tokens a model spends to reach a result you'll accept — and agentic models spend a lot more of them than the chatbots they replaced. A cheaper model that retries, reasons longer, or needs a human to fix its output can cost more per finished job than the pricier one you skipped. The unit that matters is cost per accepted result, and the discipline that keeps it honest is the same one you already apply to your cloud bill: instrument it, route deliberately, and review it on a cadence. Get that right and cheaper tokens finally show up where you want them — on the invoice, not just the pricing page.

If your AI spend is a flat line you only look at when it jumps — with no way to say which agent, which task, or which retry loop is driving it — the fix is instrumentation, not a cheaper model. Book a 30-minute working session and we'll look at what your agents actually cost per result, and where the token trail is longer than the job needs it to be.