

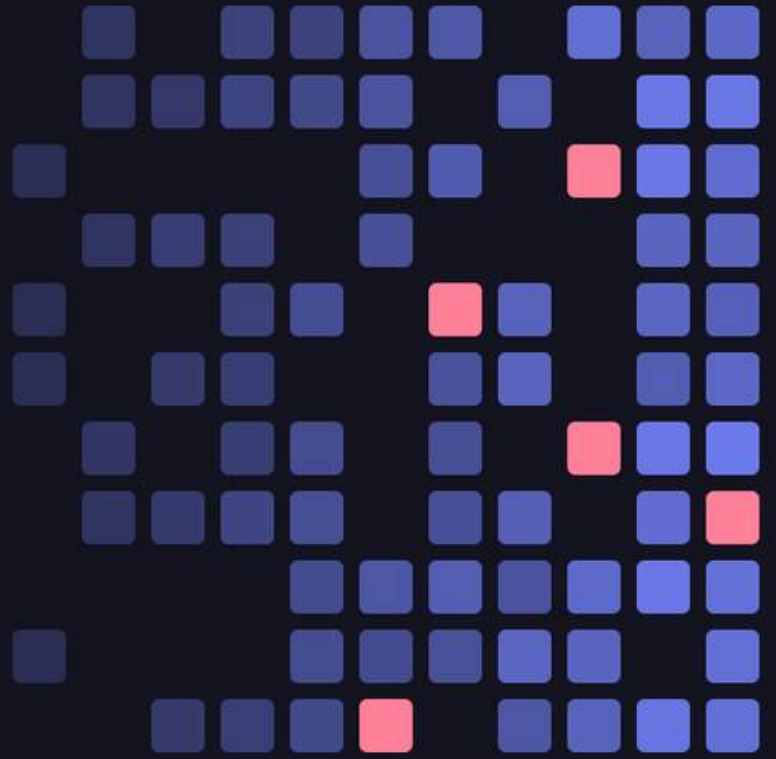
Gauntlet

ADVERSARIAL SYSTEMS IN AI · NOV 9, 2026

Inference is Crypto

Lessons on why not to trust inference providers

Tarun Chitra



History of Open Weight Models



FEB 2026

>50% of tokens

Open weights pass half of OpenRouter's tokens

JUL 2026

79% of tokens

The peak so far (72% in September)

SEP 2026

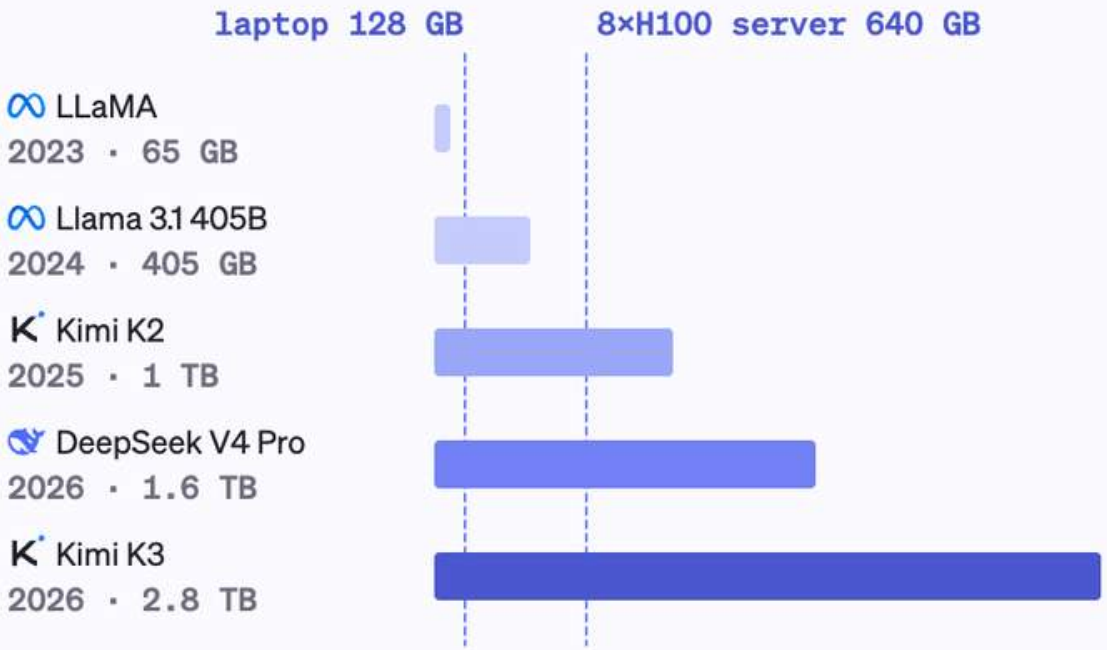
27% of spend

Up from 11% in January

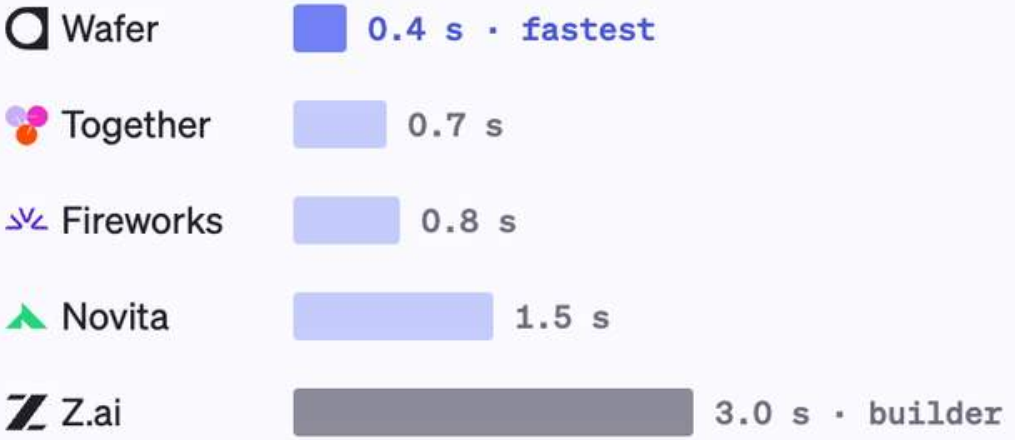
Sizes are total parameters · shares: OpenRouter Inferred (Sep 2026)

Nobody runs these locally anymore

WEIGHTS IN MEMORY, 8 BITS PER PARAMETER



TIME TO FIRST TOKEN, GLM-5.2



Specialists write custom kernels and serve the model faster than the lab that trained it

Sizes: total params · TTFT figures illustrative

Open weights unbundled the AI stack

CLOSED AI

One lab sells the whole stack

App, model, inference and price come as one product

Harness

The app: holds context, calls tools

Hermes Agent · Kilo Code · OpenClaw

Router

Picks a model, then a provider

OpenRouter

Model builder

Trains the weights, sells its own API

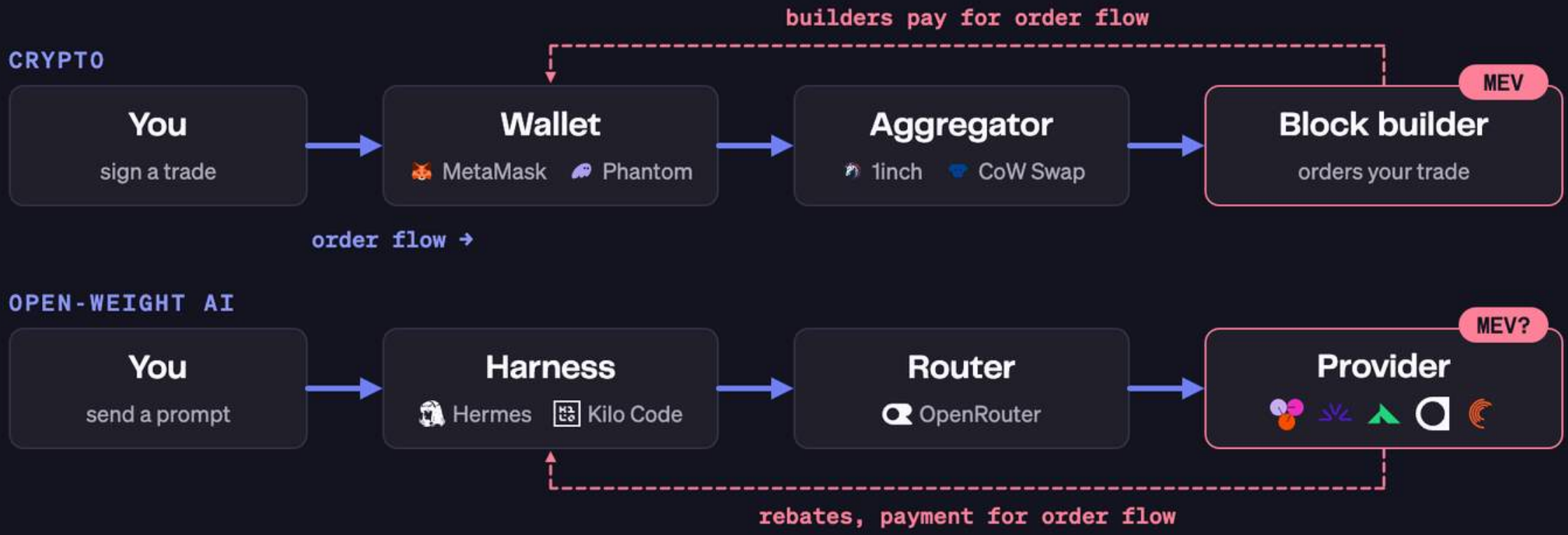
Z.ai · DeepSeek · Moonshot · MiniMax

Inference provider

Rents GPUs, sells tokens

Together · Fireworks · Novita · Wafer

Your prompts are order flow



MEV = what whoever executes your order can take from you (like a bot sandwiching your trade)

How OpenRouter picks a provider

HERMES AGENT, TURN 10

```
POST /api/v1/chat/completions
X-Title: Hermes Agent
{
  "model": "z-ai/glm-5.2",
  "messages": [
    system, user, assistant,
    tool, assistant, tool, …
  ],
  "tools": [ … ]
}
```

100K input tokens, 90K of them sent last turn

⚠ The price it weighs is the input price. Cache and speed don't count.

1 · CAPABILITY: TASK → MODEL

GLM-5.2 — Hermes names it (or asks openrouter/auto)

2 · EXECUTION: MODEL → PROVIDER

provider	input	TTFT	share of traffic
Z.ai	$1.00	3.0s	25%
Together	$1.00	0.7s	25%
Fireworks	$0.80	0.8s	39%
Wafer	$1.50	0.4s	11%
~~Provider D~~	$0.70	–	filtered out: too many errors

filter → weight by 1/price² → pick → retry on failure

Repricers game the Tullock contest



Gordon Tullock (1980): contests for rents

share ∝ (1 / price)²

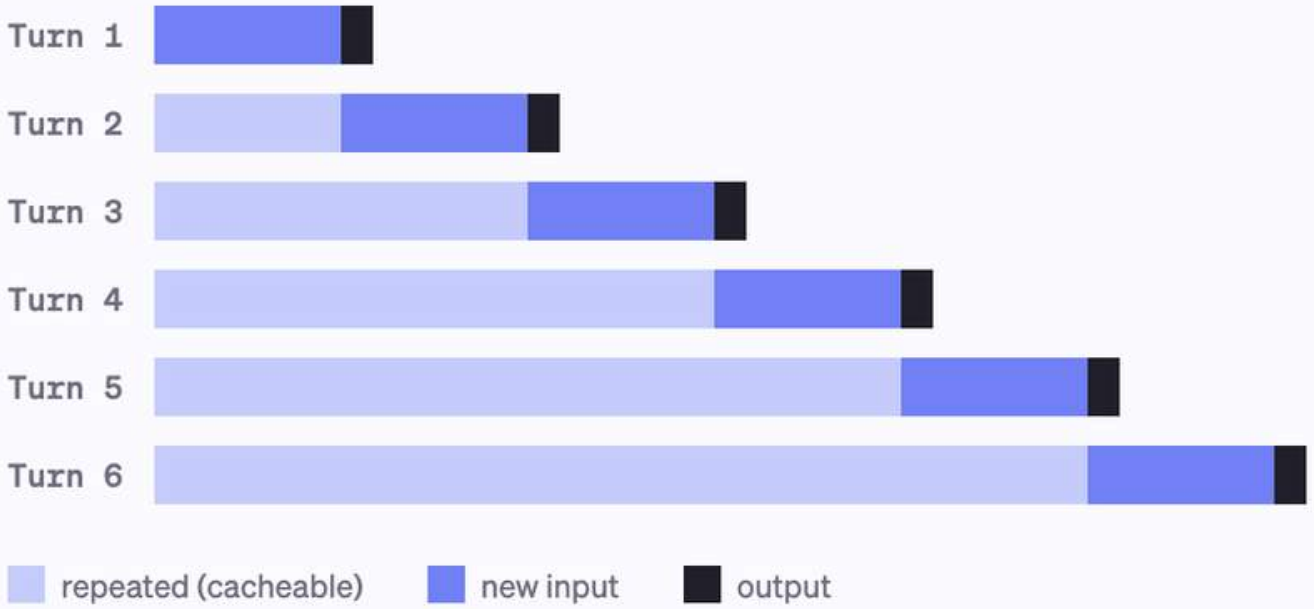
r = 2 in OpenRouter's docs, ≈ 1.97 in the data



Shares from (1/price)² · r fit: The Souq, 1,648 routing choices across 6 models and 35 providers

Agents make the cache the whole bill

TOKENS SENT ON EACH TURN OF AN AGENT LOOP



INPUT · read in parallel

$1.00

CACHED · reuse the KV cache

$0.10

OUTPUT · one token at a time

$4.00

Illustrative prices per million tokens · agents passed humans on OpenRouter on Feb 6, 2026

The repricer playbook: just don’t cache

01

Cut the input price



Win the contest and the order flow

02

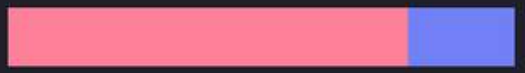
Don’t recognize the cache

```
"cached_tokens": 0
← nobody can check this
```

Whether or not you kept it

03

Bill the repeats at full price



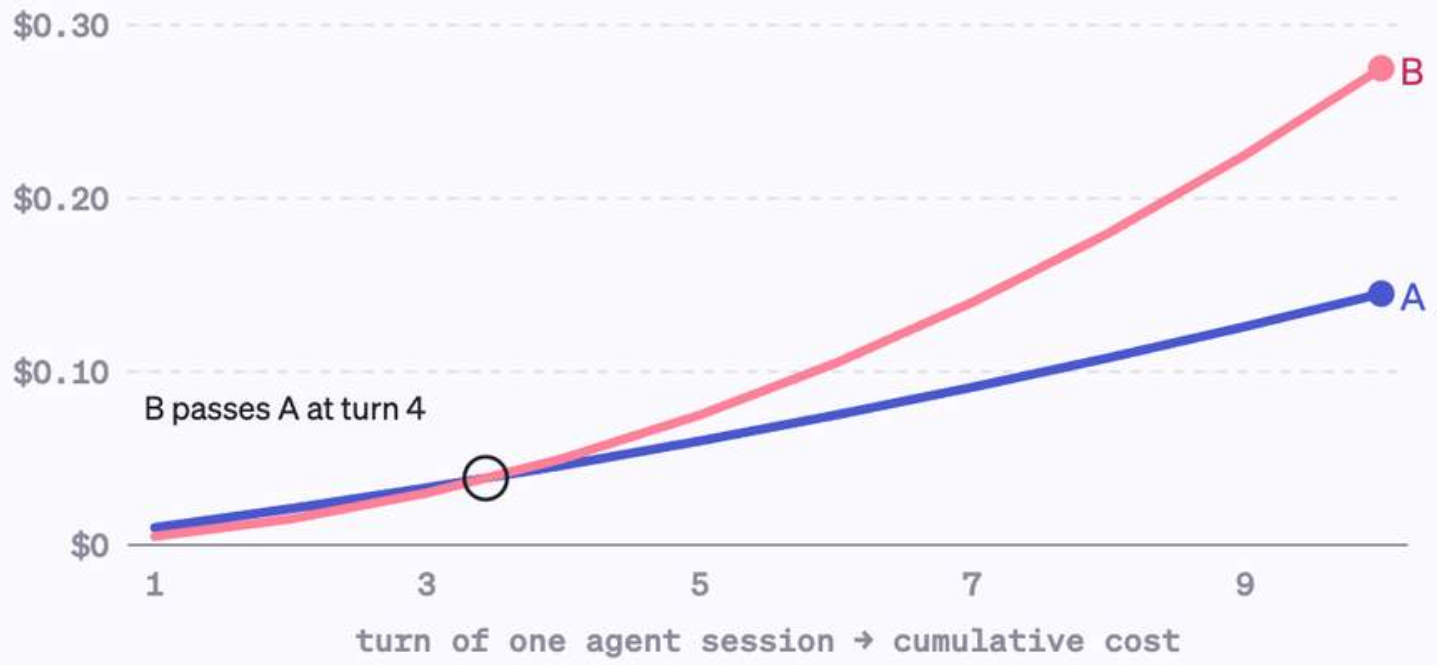
history at full input price

Every agent turn resends it all

Cheap to the router, expensive to the user.

The receipt in step 02 is written by the party that gets paid more when the number is small.

Half the price, twice the bill



A · $1.00 input · $0.10 cached
$0.145

B · $0.50 input · no cache
$0.275

...and the router sends B 4× A's traffic

Illustrative: 10K new tokens per turn, output left out

The same session, turn by turn

Turn	Input sent	Repeated	A pays	B pays	A total	B total
1	10K	0	$0.010	$0.005	$0.010	$0.005
2	20K	10K	$0.011	$0.010	$0.021	$0.015
3	30K	20K	$0.012	$0.015	$0.033	$0.030
4	40K	30K	$0.013	$0.020	$0.046	$0.050
5	50K	40K	$0.014	$0.025	$0.060	$0.075
6	60K	50K	$0.015	$0.030	$0.075	$0.105
7	70K	60K	$0.016	$0.035	$0.091	$0.140
8	80K	70K	$0.017	$0.040	$0.108	$0.180
9	90K	80K	$0.018	$0.045	$0.126	$0.225
10	100K	90K	$0.019	$0.050	$0.145	$0.275

A: $1.00/M input, $0.10/M cached · B: $0.50/M input, no cache · 10K new tokens per turn

It's already happening on OpenRouter

AT A PRICE CUT

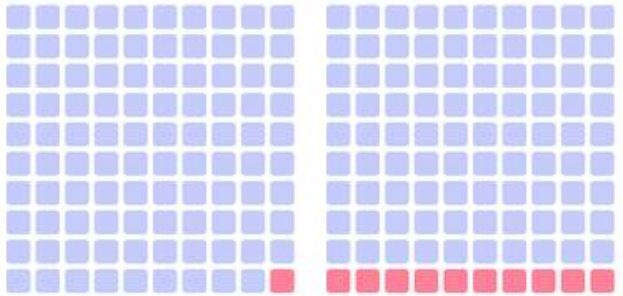
−71pp

less repeated input recognized as cached than at Z.ai

PINNED SESSIONS

Z.ai 99%

Repricers ~90%



10× more repeats billed at full price

BREAK-EVEN

~5

repeated queries before Novita costs more than the model builder's own API

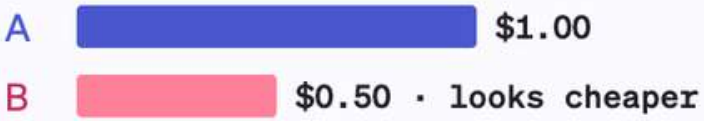
OPENROUTER REVENUE, ~$140M A YEAR

$3.5M/yr users overpay today (2%) · $14M/yr if nobody caches (10%)

The new router prices in the cache

1 · SCORE WHAT USERS ACTUALLY PAY

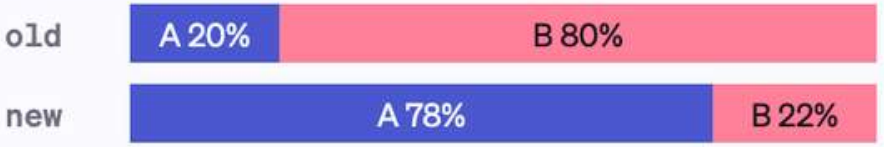
Quoted input price per M tokens



Effective price per request (55K tokens, 45K repeated)



Share of traffic



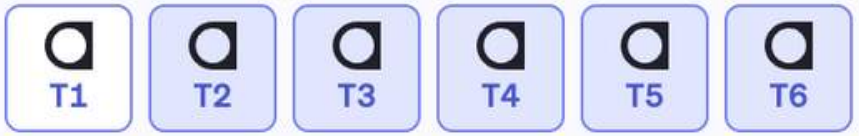
2 · KEEP EACH SESSION ON ONE PROVIDER

Before: re-routed every turn



new provider each turn → cache lost on every hop

After: pinned after turn 1



same provider → cache stays warm

⚠ Still open: turn 1 is the only auction, and the cache count is still self-reported

The new router looks like an ad auction

RANKED ON A PREDICTION

SEARCH ADS

rank = bid × predicted click rate

OPENROUTER, OCT 28

score = unit prices × predicted token mix
÷ predicted success rate

In both, you're ranked on the platform's prediction and paid on what actually happens

KNOWN EXPLOIT: SKEWED BIDDING

Bid low where the platform over-predicts, high where it under-predicts, and suppress what you control (here, cache hits)

CLOSEST RELATIVES

Pay-per-action ad auctions closest

Per-action bids × the platform's estimated action rates, and advertisers control whether actions get reported

Agarwal, Athey & Yang · AER P&P 2009

Search ads with quality scores

Rank by bid × predicted click rate

Varian · IJIO 2007
Edelman, Ostrovsky & Schwarz · AER 2007

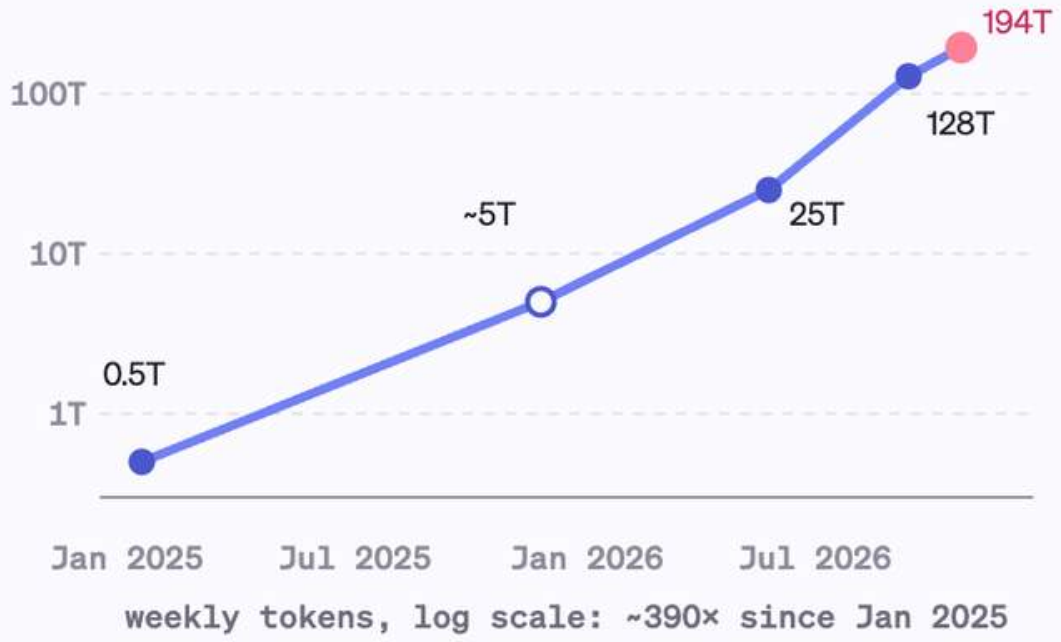
Scoring auctions

Price and quality folded into one score

Che · RAND 1993 · Asker & Cantillon · RAND 2008

Skewed bidding was first measured in timber auctions (Athey & Levin 2001)

More tokens, more ways to cheat



ALREADY IN THE LITERATURE

ICLR 2025 · 11 of 31 endpoints off-spec

Model Equality Testing: Which Model Is This API Serving?

Irena Gao, Percy Liang, Carlos Guestrin

NeurIPS 2025 workshop · quantized swaps

Are You Getting What You Pay For? Auditing Model Substitution in LLM APIs

Will Cai, Tianneng Shi, Xuandong Zhao, Dawn Song

ICML 2026 oral · token overcounting

Is Your LLM Overcharging You? Tokenization, Transparency, and Incentives

Ander Artola Velasco, Stratis Tsirtsis, Nastaran Okati, Manuel Gomez-Rodriguez

Token data: OpenRouter Inferred, OpenRouter Series B release, DiarioBitcoin

THANK YOU

Don’t trust, verify

- Score what users actually pay
- Publish every provider’s cache hit rate
- Signed receipts for cached tokens

The full post is here

Gauntlet

Tarun Chitra

Five ad attacks on the new router

Each one borrows a known exploit from online ads and aims it at one piece of the new router

Attack	In ads	What it exploits in the router	What would catch it
Skewed bidding A2	Advertisers shift bids onto actions the platform over-predicts	One averaged cache hit rate per provider	Bill at the scored price, or score each session on its own hits
Token inflation A3	Click fraud: whoever counts the clicks gets paid per click	The bill uses the provider's own token counts	Count tokens on the router's side and audit hidden reasoning
Cloaking A4	A clean page for the reviewer, something else for users	Quality is only checked with test requests	Tests that replay real sessions, and model equality tests
Sawtooth pricing A5	Bid cycles on Overture, the first search ad auction	Only turn 1 is auctioned, then the session is pinned	Lock the turn-1 terms for the whole session
Whitewashing A6	Banned accounts come back under new names	New endpoints start at the model's typical numbers	Reputation per operator, pessimistic starts, and bonds

Skewed bidding

IN ADS Advertisers who know the real action rates better than the platform shift their bids onto what it over-predicts

IN THE ROUTER

1. The router scores each provider on one cache hit rate, averaged over its past traffic
2. S keeps that average high on short sessions and posts a cached price of a penny
3. On long agent sessions S lets the cache miss, so most tokens bill at the full input price

WHAT WOULD CATCH IT

Bill at the scored price so the provider eats its own misses, or score each session on its own hits

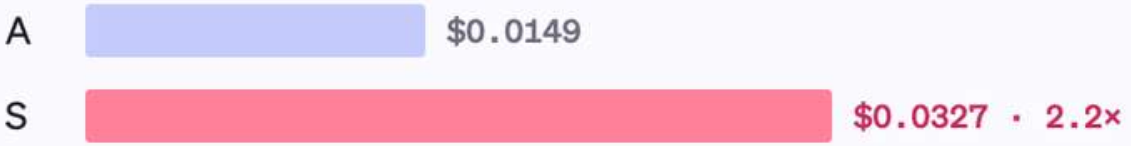
ONE TURN · 55K TOKENS, 45K REPEATED

Price per 1M tokens	Input	Cached
A · honest	$1.00	$0.10
S · skewed	$1.00	$0.01

WHAT THE ROUTER SCORES · 99% HITS



WHAT A LONG SESSION PAYS · S HITS 50%



Token inflation

IN ADS Click fraud: whoever counts the clicks gets paid per click, so they can bill clicks that never happened

IN THE ROUTER

1. The score predicts output tokens from the model's typical traffic, not from this provider's counts
2. The bill uses the provider's own counts, including hidden reasoning and how the text was split into tokens
3. So padding the reasoning, or splitting text into more tokens, raises the bill and leaves the score alone

WHAT WOULD CATCH IT

Count visible tokens on the router's side with the model's public tokenizer, audit hidden reasoning, or price per character instead of per token

ONE TURN, AS BILLED



WHAT THE ROUTER SCORES

The same score for both, because the extra tokens only show up on the bill

SAME TEXT, MORE TOKENS

`inference` 1 token

`in` `fer` `ence` 3 tokens, billed 3×

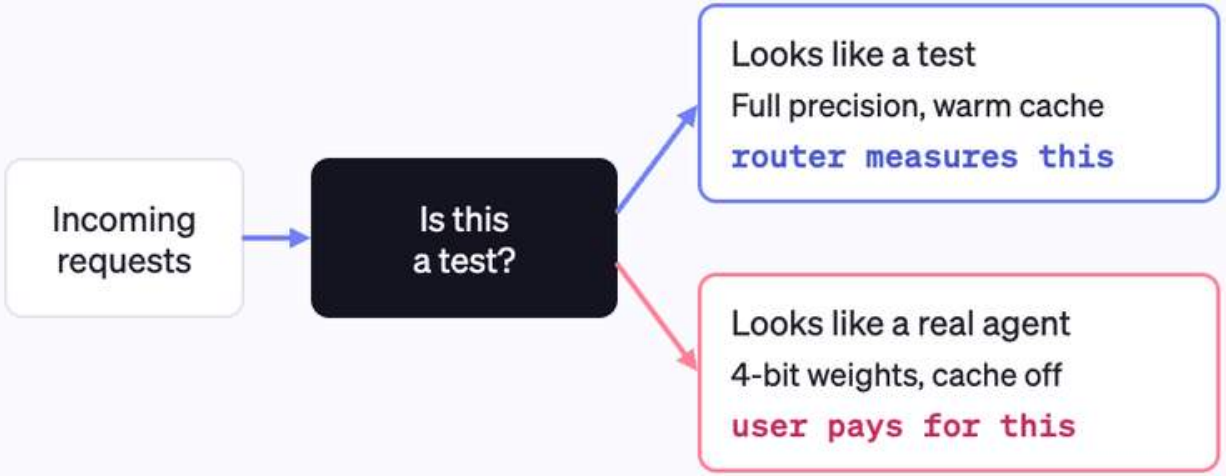
Cloaking

IN ADS Scam advertisers show the platform's reviewers a clean page and show real users something else

IN THE ROUTER

1. The router can only check what a provider really serves by sending its own test requests
2. A provider can learn what tests look like: short prompts, benchmark questions, no tools or history
3. Tests get full-precision weights and a warm cache. Long agent sessions get 4-bit weights and no cache

ONE PROVIDER, TWO PATHS



WHAT WOULD CATCH IT

Tests that replay real agent sessions, statistical tests that the outputs come from the same model, or attested hardware

ALREADY SEEN ON RESOLD APIS

Matched the official model on everyday tasks, then fell up to 40% on AIME

Sawtooth pricing

IN ADS On Overture, rival advertisers raised and cut their bids in repeating cycles, leaving a sawtooth in the price

IN THE ROUTER

1. Only turn 1 is auctioned. After that a session stays pinned to its provider unless it goes down
2. Capture: undercut everyone and win a pile of first turns, each one pinning a long session
3. Harvest: raise the price, or let the cache go cold, on sessions that can't leave. Then cut again

WHAT WOULD CATCH IT

Lock the turn-1 price and cache terms for the session, or re-auction any session whose terms change



Whitewashing

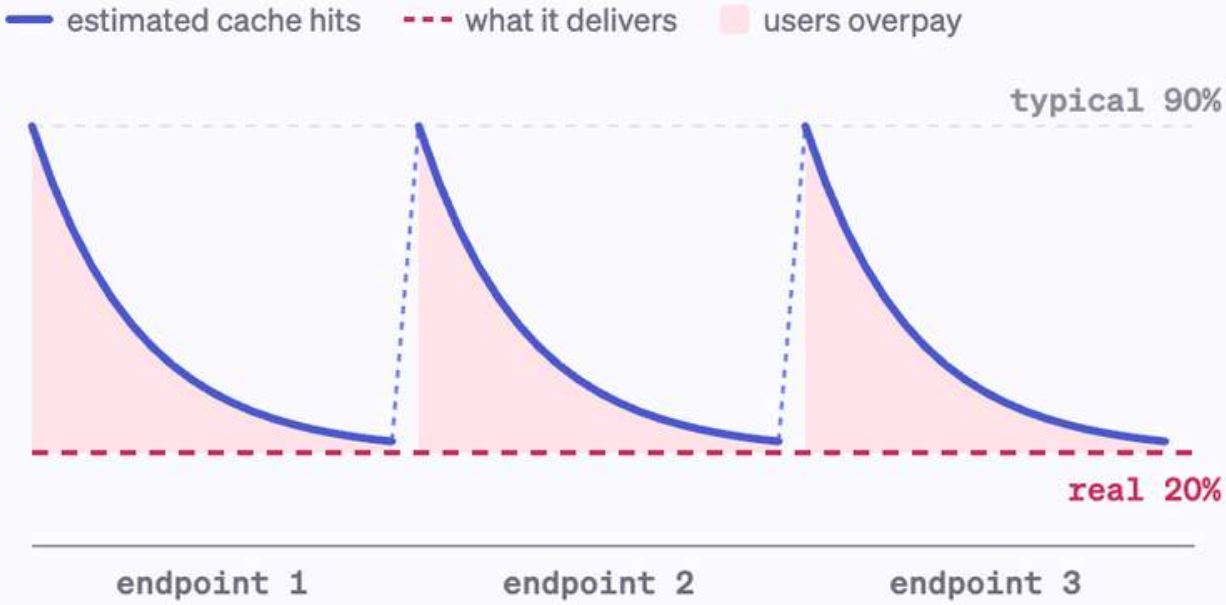
IN ADS A banned advertiser or a badly rated seller opens a new account and starts over with a clean record

IN THE ROUTER

1. A new endpoint has no history, so the router starts it at the model's typical cache hit and success rates
2. The provider skips the cache and serves cheap weights until the router's measurements catch up
3. Then it retires that endpoint and lists a fresh one, say in a new region, back at typical numbers

WHAT WOULD CATCH IT

Keep reputation per operator, not per endpoint, start new endpoints from pessimistic numbers, or require a bond that gets slashed



Every relist resets the estimate to typical, so the pink gap never closes