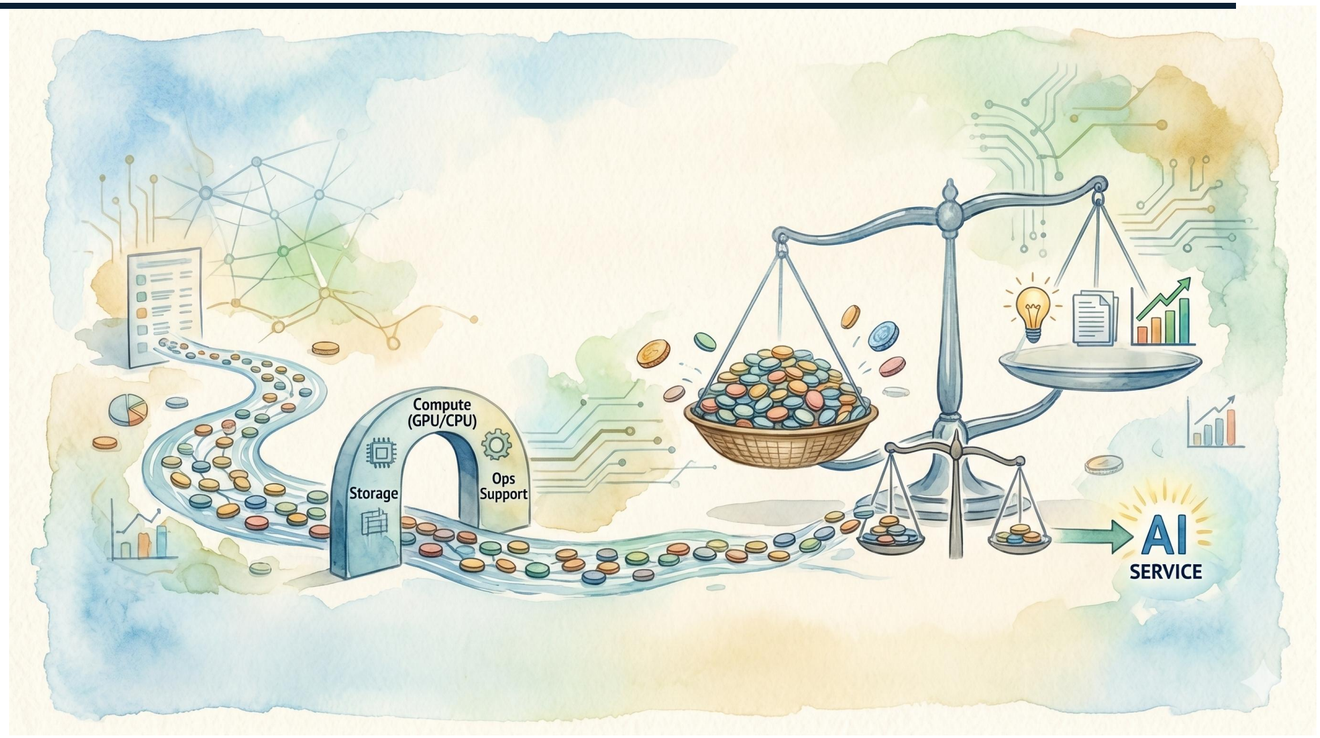
# The Economics of Tokens in AI




*From provider price lists to fully loaded service cost —*
*A practical framework for managing AI's new unit of consumption.*

**01 Measure**

Unit economics, cost per token, cost per business outcome

**02 Optimize**

Hardware, architecture, software, prompt and routing levers

**03 Govern**

Showback, chargeback, observability, policy guardrails

# Token economics is now an operating model problem — not just a model-pricing problem.

## 1 Tokens are the new unit of consumption

Weekly token volume on OpenRouter rose ~68× in 15 months, from 0.4T to 27.0T tokens. Tokens are now AI's currency, capacity unit, and monetization unit. *[Chen et al., 2026]*

## 2 Cost per token is not a constant

The same token can cost very different amounts depending on hardware generation, attention kernel, batching state and KV-cache residency. *[Wu & Deng, 2026]*

## 3 Agentic AI is a token multiplier

Multi-agent workflows add input + communication + reasoning + tool tokens. Coordination cost scales Coordination cost scales $O(|V|^2)$ in dense topologies. *[Chen et al., 2026]*

## 4 Provider price ≠ fully loaded cost

In production RAG and agentic stacks, the harness — vector DB, embeddings, orchestration, orchestration, observability — is typically **40–60%** of feature spend. *[FinOps Foundation, 2026] Foundation, 2026]*

## 5 FinOps discipline transfers — with new primitives

Inform → Optimize → Operate still applies. New primitives: API-key governance, proxy/observability, model routing, prompt & KV caching, batch APIs. *[FinOps, Microsoft, 2025] Microsoft, 2025]*

So What — The Mandate

Lower the **fully loaded cost per useful token** — and the cost per **successful business outcome** — every quarter, the way you already do for compute, storage and SaaS.

Token economics is the cloud-unit-economics challenge of the AI era — and the practitioners who move first will compound the advantage.

# Tokens have become AI's consumption, capacity and monetization unit — all at once.

Factor of production

### Tokens consume GPU time, memory and bandwidth — the the raw inputs of AI output.

Cost ties directly to silicon, electricity and engineering capacity.

Medium of exchange

### Providers price, sell and meter meter capacity in tokens.

Input, cached input, output and thinking tokens are all separate price points.

Unit of account

### Budgets, forecasts and showback are increasingly denominated in tokens.

Cost per query, per user, per workflow — all roll up from a token base.

Driver of valuation

### Token usage links directly to enterprise value.

Monthly Active Users (MAU) and engagement explain ~86% of the variance in AI token valuation; pricing alone does not.

OpenRouter weekly volume

**~68×**

Growth in 15 months — from 0.4T to 27.0T tokens / week (Dec 2024 → Mar 2026).

FinOps practitioner survey

**#1**

Managing token cost & usage in SaaS-model AI is the top FinOps challenge today.

Output : Input ratio in agents

**150:1**

Input-heavy ratio in complex coding agents — context loaded repeatedly across turns.

AI token valuation drivers

**85.7%**

Variance in AI token valuation explained by MAU + visits; pricing only marginal (p≈0.06). *Network effects dominate.*

*Sources: Chen et al., "Token Economics for LLM Agents" (2026); Akpan, "Valuation of AI Tokens" (2026); FinOps Foundation, Token Economics for SaaS (2026).*

The Executive Formula

# Forecast AI demand the way NVIDIA does — users × sessions × tokens, then multiply for reality.

Token Demand Equation

$$\text{Tokens}_{\text{demand}} = \text{Users} \times \text{Sessions} \times \text{Tokens}_{\text{per session}} \times M_{\text{reasoning}} \times M_{\text{agentic}} \times (1 - \text{Cache-hit})$$

Multipliers — not additions — turn a benign capacity model into a five-to-ten-times forecast. *NVIDIA AI Podcast, 2025.*

Fully Loaded Cost per Token

$$\text{Cost / token} = \text{Provider price} + \text{Workload tax} + \text{Platform overhead} - \text{Optimization yield}$$

Microsoft FOCUS: *Unit cost = BilledCost ÷ ConsumedQuantity*. The four levers above are what bend that ratio over time.

Provider price

**List rate × workload mix**

Input vs cached input vs output vs thinking; tier (Standard / Batch / Flex / Priority).

Workload tax

**Context, turns, agents**

Long context, accumulated history, multi-agent calls, hidden reasoning tokens.

Platform overhead

**Harness around the model**

Vector DB, embeddings, reranker, orchestrator, observability — 40–60% of feature spend.

Optimization yield

**What the FinOps team claws back**

Caching, batch API, right-sizing, KV-cache reuse, quantization, routing.

CFO insight: a ***more expensive*** model can be ***cheaper per task*** if it needs fewer tokens to succeed. Optimize per outcome, not per list price. — **Patel, Invest Like the Best, the Best, 2025.**

What you are actually paying for

# Five token types — and their price ratios — drive 80% of variance in your AI bill.

| Token type | What it is | Indicative price (per 1M) | Cost behavior | Lever |
|---|---|---|---|---|
| **Input**<br>prompt + history | Everything you send: system prompt, RAG RAG context, conversation history. | **GPT-5.4: $2.50 · Gemini 3 Pro: $1.50–$2.00** | Grows linearly with context; long history is the silent killer. | **Summarization, RAG, sliding window** |
| **Cached input**<br>repeat prefixes | Stable system prompts / docs reused across calls. | **Often 10× cheaper than input · GPT-5.4: $0.25** | 80–90% reduction on the cached portion of input tokens. | **Prompt caching, semantic cache** |
| **Output**<br>model response | Visible response generated by the model. | **Typically 3–5× input · GPT-5.4: $15.00** | Determined at runtime — hardest to control directly. | **Length limits, structured output** |
| **Reasoning / Thinking**<br>hidden chain-of-thought | Internal reasoning emitted by reasoning models — often invisible to user. | **Billed as output (Gemini explicitly states "incl. thinking tokens")** | Highly variable; can dominate cost on hard prompts. | **Thinking-budget caps, hybrid routing** |
| **Communication**<br>inter-agent + tool | Tokens spent agent→agent or agent→tool — invisible on the user prompt. | **Same as input/output but volume can be 10–100× higher** | Scales $O(\|V\|^2)$ in dense agent topologies. (see slide notes) | **Topology design, schema discipline** |

Practitioner rule

**An invoice line that says "tokens" hides at least four price points and three hidden multipliers.** ***Always decompose before you optimize.***

*Sources: OpenAI API pricing (2026); Google Gemini API pricing (2026); FinOps Foundation Token Economics for SaaS (2026); Chen et al. (2026).*

The Provider Price List

# Across vendors and tiers, list prices already span two orders of magnitude.

| Model | Tier | Input / 1M | Cached input / 1M | Output / 1M | Cost insight |
|---|---|---|---|---|---|
| **OpenAI · GPT-5.5**<br>Frontier | Standard | $5.00 | $0.50 (10× off) | $30.00 | 6× output premium |
| **OpenAI · GPT-5.4 mini**<br>Cost-efficient | Standard | $0.75 | $0.075 | $4.50 | ~7× cheaper than 5.5 |
| **Google · Gemini 3 Pro**<br>Frontier reasoning | Standard | $2.00 / $4.00* | $0.20 / $0.40* | $12 / $18* (incl. thinking) | *price doubles >200K context |
| **Google · Gemini 3 Flash**<br>High-volume agentic | Standard | $0.25 | $0.025 | $1.50 | 8× cheaper than Pro |
| **Same model · Batch tier**<br>Latency-tolerant | Batch | −50% | −50% | −50% | Free 50% if you can wait |
| **Same model · Priority tier**<br>SLA-guaranteed | Priority | +80% | +80% | +80% | Pay 1.8× for guaranteed throughput |

Frontier vs mini

### Within one provider, the smallest model is 50–100× 100× cheaper than the frontier model.

Most enterprise workloads do not need frontier — model right-sizing is the highest-leverage lever.

Cached vs raw input

### Cached input is consistently ~10× cheaper than fresh input.

A stable system prompt that is not cached is the single most common avoidable cost.

CFO read

### Tier and routing decisions can move unit cost by 20–80% with no quality change.

Architecture, not procurement, is now the primary lever.

# An issue tree of six driver families explains why the same task can cost 10× more in one stack than another.

**Effective cost / token**

**Why the same token costs costs different amounts**

### 1. Provider & model

**List rate × tier × tier modifier**

Input / cached / output split · Standard / Batch / Flex / Priority · Context-band step pricing · Batch API −50%, Priority +80%.

Frontier vs mini within one vendor: 50–100× per-token spread.

### 2. Prompt & context

**What you send each call**

System prompt length · conversation history (re-sent each turn) · RAG payload · document attachments · context-window band.

10-turn chat ≈ 10× cost of single turn if no summarization.

### 3. Output & reasoning

**What the model emits**

Response length · hidden chain-of-thought · reasoning effort setting · format (prose vs JSON) · output priced 3–5× input.

Thinking tokens can dominate: Gemini explicitly bills "incl. thinking".

### 4. Agentic workflow

**Turns, tools, agents**

Number of agents in topology · inter-agent comms · tool calls · retries · reflection loops · redundant context loading.

Coordination cost scales $O(|V|^2)$ in dense topologies — the "communication tax."

### 5. Hardware & runtime

**Where the token actually runs**

GPU generation · memory bandwidth · KV-cache residency · batch composition · attention kernel · prefill / decode disaggregation.

"Token cost is hardware- and architecture-dependent." — Wu & Deng, 2026.

### 6. Software optimizations

**What the platform negates**

Continuous & in-flight batching · PagedAttention KV · prompt & semantic caching · quantization · speculative decoding · MQA / GQA · routing.

Each lever shifts the bottleneck — combined effect can be 5–10× throughput.

*Sources: Wu & Deng, "Computational Token Economics" (2026); Chen et al., "Token Economics for LLM Agents" (2026); NVIDIA "Mastering LLM Inference Optimization" (2024); FinOps Foundation (2026).*

Why a token's cost is hardware-dependent

# "Extreme co-design" — across silicon, memory, software and runtime — is what compounds cost-per-token down.

Layer 1 · Silicon

## Hardware generation generation

GPU generation, tensor cores, memory bandwidth (HBM), on-chip cache, NVLink / interconnect bandwidth, prefill vs decode placement, single-node vs disaggregated serving.

Same token costs different amounts on H100 vs B200 vs MI300 — bandwidth and KV residency dominate decode economics.

Business impact

Choice of inference provider locks 2–5× of unit cost — cost — re-evaluate annually.

Layer 2 · Model architecture

## Attention & layout

Multi-Query / Grouped-Query Attention (MQA/GQA), FlashAttention-2/3, MoE sparsity, speculative draft + verifier, distillation, quantization (INT8 / FP8 / INT4).

FlashAttention-3 reorders memory traffic; MQA/GQA shrinks KV-cache to free batch capacity; quantization halves memory halves memory at near-equal quality.

Business impact

2–4× throughput on the same hardware — pure cost-per-token reduction.

Layer 3 · Serving runtime

## Batching & KV mgmt

Continuous & in-flight batching, PagedAttention KV, prefill–decode disaggregation, tensor / pipeline parallelism, chunked prefill.

PagedAttention eliminates KV fragmentation → much larger batches; in-flight batching keeps GPUs busy across uneven across uneven request lengths.

Business impact

Higher tokens / GPU-hour → lower TCO per request, request, better SLA headroom.

Layer 4 · Application

## Caching & routing

Prompt caching, semantic cache, response memoization, RAG payload trimming, hybrid local/cloud routing, model right-sizing, thinking-budget caps.

TEA-UF retail case: caching + 40% local routing cut monthly token spend from $1.8M → $1.1M (~39%). [Nandagopal, 2025] [Nandagopal, 2025]

Business impact

30–90% effective cost reduction — the "free money" most money" most teams leave on the table.

**Lower cost / token raises demand — the Jevons effect.** Plan capacity for the next 10× of usage, not just the next budget.

# In multi-agent systems, every "specialization dividend" must clear a coordination tax.

**How tokens flow through an agentic workflow**

Single LLM call

**Baseline 1×**


Input : Output ≈ 3:1 to 5:1.

Multi-agent workflow

**Token multiplier — typically 5–50×**


**Communication tokens** — agent-to-agent messages, JSON schema validation, tool I/O. Add reflection loops, retries and you compound consumption.

Token flow now traverses **(1) Input** → **(2) Reasoning / CoT** → **(3) Communication (inter-agent)** → **(4) External (RAG / tools)** → **(5) Output**. In dense topologies, coordination cost scales **$O(|V|^2)$** — the "Coasian boundary" of agent specialization.

The multiplication effect — empirical

## A single agent-driven coding task can consume consume 150× more input tokens than a one-shot LLM call.

**68×** Growth in weekly tokens routed via OpenRouter — Dec 2024 to Mar 2026 — driven largely by agentic apps.

**$O(V^2)$** Communication scaling in dense agent topologies — every added agent inflates pairwise messaging.

**+×** Security / verification adds a "risk premium" shadow price on every token in trust-sensitive flows.

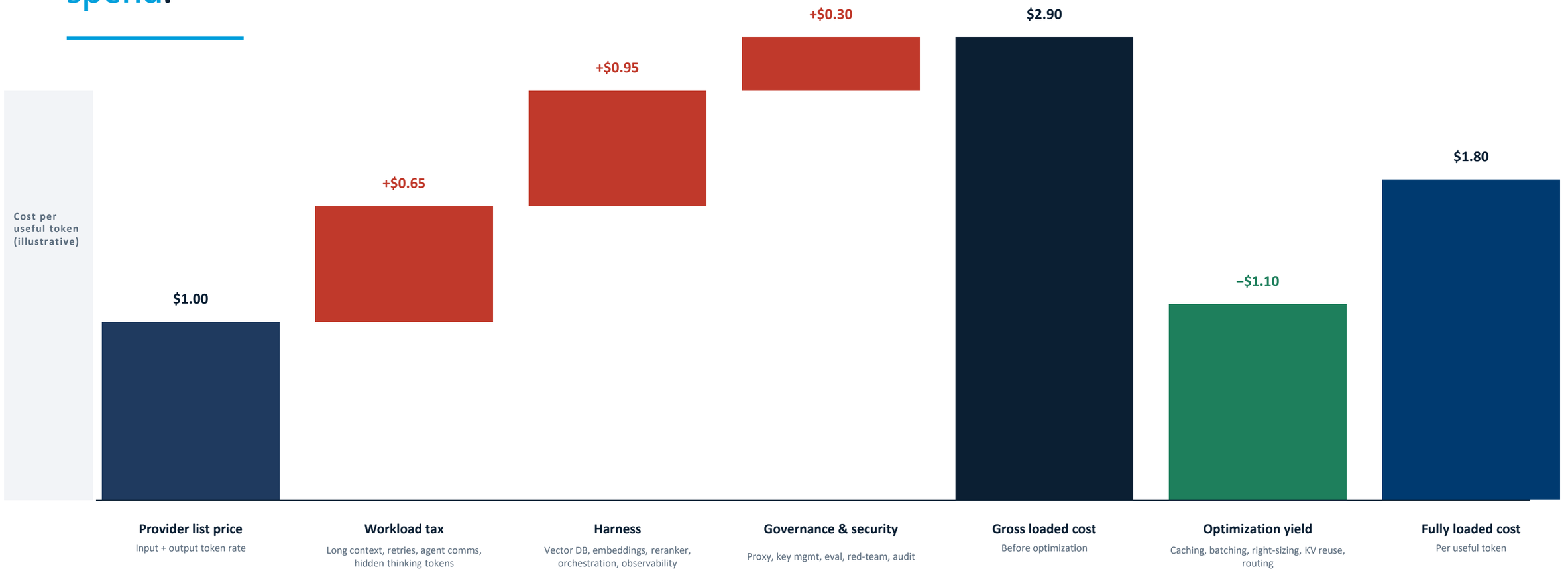
Executive read

**Specialization is real — but past a point, marginal coordination cost overtakes the productivity gain.**
**Govern agentic spawning; require architectural review & spend modeling before any autonomous workflow ships to production.**

*Sources: Chen et al., "Token Economics for LLM Agents" (2026); FinOps Foundation Token Economics for SaaS (2026).*

From price list to fully loaded service cost

# Provider price is just the start — the harness around the model is typically 40–60% of feature spend.




**Read:** illustrative — provider price normalised to $1.00 per useful token. The model bill is <35% of the gross loaded cost; mature FinOps reclaims ~40% via optimization. The exhibit is shaped from FinOps Foundation harness benchmarks (40–60% of feature spend) and Microsoft FinOps Toolkit unit-cost methodology.

# Chargeback works only when you separate cost to generate a token from the value of one.

### From token unit cost to internal transfer price

**Layer 1 · Token utility**

**What does this token DO for the business?**

Resolved ticket, generated lead, drafted contract clause, code-review comment. Defines the *value density* of a token — and varies sharply by LOB.

**Layer 2 · Cost to generate a token**

**BilledCost ÷ ConsumedQuantity (FOCUS)**

Microsoft FOCUS unit-cost formula. Capture by SKU meter category, application tag and cost center.

**Layer 3 · Internal transfer / recovery price**

**Raw model cost + Harness + Platform overhead + Risk premium**

Recovery price is set on *fully loaded* cost — not the provider invoice — so the platform team is sustainably funded.

### Different LOBs consume different token mixes

| LOB | Token mix | Unit metric | Value density |
|---|---|---|---|
| **Customer Care** | Cached input-heavy<br>Mini model | $ / resolved ticket | **High** |
| **Engineering / Code** | Frontier reasoning<br>+ agentic loops | $ / merged PR | **Very High** |
| **Marketing** | Output-heavy<br>1:1 in:out | $ / accepted asset | **Medium** |
| **Risk / Legal** | Long context<br>+ verification | $ / reviewed clause | **High** |
| **Internal productivity** | Mixed; high session count | $ / active user / mo | **Variable** |

### Recommended unit metrics — surface these to the LOB, not raw token counts

**$ / query · $ / API call**
Useful for capacity planning

**$ / user / month**
Compare against seat-priced alternatives

**$ / workflow completion**
For multi-agent & agentic apps

**$ / business transaction**
Embedded in customer outcomes

*"Our AI-assisted ticket resolution costs $0.18 per ticket, down from $0.31 last quarter" enables real ROI conversations. — FinOps Foundation, 2026*

*Sources: Microsoft FOCUS / Azure FinOps Toolkit (2026); FinOps Foundation Token Economics for SaaS (2026); Akpan, "Valuation of AI Tokens" (2026).*

# Without a proxy / observability layer, every tag, budget and policy is just an aspiration.

01 · API key governance

### Foundational control layer — without this, nothing else works.

Keys mapped to a named owner, application and cost center. No self-service creation. Scoped to specific models. Rotated on schedule. Documented revocation runbook for runaway spend or compromise.

02 · Proxy & observability layer

### A gateway that injects what providers don't: tags, don't: tags, policy, attribution.

LiteLLM / Portkey / Helicone / Bedrock / Azure OpenAI proxies. Inject user, app, feature, environment, experiment metadata into every call. Multi-provider attribution & policy enforcement.

03 · FOCUS tagging & allocation

### Cost center, application, environment, feature, experiment.

Map providers to FOCUS columns: ServiceName, ConsumedQuantity, BilledCost, Tags, ResourceId. Showback to LOB owners; chargeback once attribution is >95% complete.

04 · Model & routing policy

### Approved-model list, max context, agentic review review gate.

Frontier-only flags. Data-classification guardrails (what can leave). Architectural review before any autonomous agent ships. Dynamic routing escalates only ambiguous queries to frontier models.

05 · Budgets & anomaly detection

### Tokens-per-minute limits, not just monthly $ $ budgets.

Account-level alerts as backstop; workload-level alerts to assign root cause. Detect: runaway agent loops, prompt-injection abuse, accidental frontier-model swaps, dev/test bleed in production.

06 · Accountability & ownership

### A named individual owns AI cost — with authority authority to enforce.

Three viable models: **FinOps-led**, **Platform-Engineering-led**, or **AI Center of Excellence**. Without clear ownership, every function defers — and nothing ships.

**FinOps lifecycle applied to tokens**

**Inform** Inventory accounts, deploy proxy, build dashboards. *Visibility first.*

**Optimize** Right-size models, enable caching, migrate to batch, prune context.

**Operate** Chargeback, policy enforcement, commitment negotiation, CI/CD negotiation, CI/CD integration.

*Sources: FinOps Foundation Token Economics for SaaS (2026); Microsoft FinOps Toolkit + FOCUS (2026).*

Optimization levers — ranked by business impact

# Most enterprises can cut fully loaded cost per token by 40–70% in two quarters — without — without touching a model contract.

| Layer | Lever | Mechanism | Typical savings | Effort | Owner |
|---|---|---|---|---|---|
| App | **Model right-sizing** | Dynamic routing — escalate only ambiguous requests to frontier | **60–90%** | Medium | Platform Eng + LOB |
| App | **Prompt caching** | Provider-side cache for stable prefixes (system prompt, RAG) | **50–90% on cached** | Low | App teams |
| App | **Batch API** | Move latency-tolerant workloads to batch tier | **~50% flat** | Low | Data & ML eng |
| App | **Context management** | Summarization, sliding window, RAG payload trimming | **20–60%** | Med–High | App teams |
| Runtime | **PagedAttention + in-flight batching** | vLLM / TensorRT-LLM serving — KV mgmt, continuous batching | **2–4× throughput** | Med | Inference platform |
| Arch | **Quantization + speculative decoding** | FP8 / INT4 weights; small draft model + verifier | **2–3× throughput** | High | ML platform |
| Procure | **Commitments & provisioned tput** | PTUs / EDP / volume agreements with spillover | **10–30%** | Low | FinOps + Procurement |
| Hybrid | **Local / open-source routing** | Run Llama / Qwen on owned GPUs for non-frontier work | **30–50% on covered** | High | ML / Infra eng |

Sequence matters

**Right-sizing & caching first (low effort, highest yield) → batching & context next → runtime / arch optimizations once you understand workloads → commitments only after 60–90 days of clean after 60–90 days of clean baselines.**

Operating model & maturity roadmap

# From experimentation to industrialized token FinOps — three stages, ~12 months. months.

| | Stage 1 · Months 1–3<br>CRAWL — Foundational visibility | Stage 2 · Months 3–9<br>WALK — Allocation & optimization | Stage 3 · Month 9+<br>RUN — Active governance |
|---|---|---|---|
| **Visibility** | Account-level provider spend; ad-hoc dashboards. | **Walk** Workload-level attribution; unit metrics published. | **Run** Request-level attribution + AI cost in board KPIs. |
| **Allocation** | None — costs land in IT or engineering. | **Walk** Showback to teams & LOBs. | **Run** Chargeback with team budgets and recovery prices. |
| **Optimization** | Ad-hoc; engineer-driven; no shared playbooks. | **Walk** Right-sizing review & prompt caching across top-spend apps. | **Run** Dynamic routing, cost in CI/CD, modeled commitments. |
| **Governance** | No policy; developer-led credit-card sprawl. | **Walk** Policy defined; API-key governance enforced. | **Run** Policy enforced via proxy controls; agentic gate live. |
| **Tooling** | Provider dashboards only. | **Walk** Proxy + basic FinOps platform integration. | **Run** Integrated observability + warehouse + FOCUS pipelines. |
| **Cultural outcome** | Engineering aware of cost. | **Walk** Shared visibility; LOBs trust the numbers. | **Run** Cost-conscious AI development by default. |

**Underlying lifecycle:** Assessment → Design → Implementation → Evaluation · TEA-UF (Nandagopal, 2025) — repeats every quarter as models, prices and workloads evolve.

# A 90-day action agenda — visibility before optimization, optimization before chargeback.

Days 0 – 30

## Inventory & instrument

"You can't optimize what you can't see."

1. Inventory every model-provider account, API key & payment method.
2. Name a single accountable owner — FinOps, Platform Eng, or AI CoE.
3. Stand up an LLM proxy / observability layer; tag every call with app + cost center.
4. Publish a v1 spend dashboard with FOCUS unit-cost (BilledCost ÷ Quantity).
5. Set account-level $ budgets and TPM rate limits on every key.

Days 31 – 60

## Quick-win optimization

Demonstrate value while attribution matures.

6. Right-size the top 5 highest-spend apps — frontier → standard / standard / mini.
7. Enable prompt caching on every app with a stable system system prompt.
8. Migrate latency-tolerant workloads to Batch API (~50% off).
9. Audit agentic workflows for runaway loops, retries, schema friction.
10. Publish per-LOB unit metrics ($/ticket, $/PR, $/asset, $/user/mo).

Days 61 – 90

## Govern & commit

Lock the operating model.

11. Approve policy: model allowlist, max context, agentic review review gate.
12. Move from showback → chargeback for LOBs >X% of AI spend. spend.
13. Negotiate enterprise commitments based on 60-day baseline; demand spillover.
14. Stand up anomaly detection on tokens-per-minute and cost-per-outcome.
15. Set the next quarter's target: –X% on fully loaded cost / outcome.

Target outcome by Day 90

**Every dollar of AI spend is tied to a named owner, an application, and a unit metric.**
**A measurable downward trend in fully loaded cost per useful token — and per business outcome.**

Appendix

# References & sources

**Framework attribution**

The "Cost / token = Provider price + Workload tax + Platform overhead − Optimization yield" framework and the 90-day agenda are an **integrated synthesis** of the sources below. TEA-UF (Nandagopal), the Dual-View (Chen et al.) (Nandagopal), the Dual-View (Chen et al.) and the FinOps Inform-Optimize-Operate model are **sourced** verbatim where cited.


### Academic & research papers

**Nandagopal, K. (2025).**

Tokens as Currency: A Novel Framework to Sustain AI Adoption and Profitability. (Link)

→ TEA-UF lifecycle, retail / healthcare cost cases (~50% savings).

**Akpan, E. (2026).**
Attention Is All You Need: An Analysis of the Valuation of AI Tokens. (Link)
→ MAU + visits explain 85.7% of variance; pricing only marginal.

**Chen, et al. (2026).**

Token Economics for LLM Agents: A Dual-View Study from Computing and Economics. (Link)

→ Multi-agent token flow, communication tax $O(|V|^2)$, shadow pricing.

**Wu, Y. & Deng, Z. (2026).**
Computational Challenges in Token Economics. (Link)
→ Hardware/architecture dependence of token cost; PagedAttention, FlashAttention.

### Provider price lists (June 2026)

OpenAI API Pricing — *openai.com/api/pricing/*
Google Gemini API Pricing — *ai.google.dev/gemini-api/docs/pricing*

### Web sources & expert interviews

**FinOps Foundation (2026).**
Token Economics for SaaS-Model AI APIs. *finops.org/wg/token-economics-saas/*
→ Inform-Optimize-Operate, API key governance, proxy + observability, harness 40–60%.

**Microsoft TechCommunity (2025).**
Managing Azure OpenAI Costs with the FinOps Toolkit and FOCUS — Turning Tokens Into Unit Economics. (Link)
→ FOCUS columns; Unit cost = BilledCost / ConsumedQuantity; tagging.

**NVIDIA Developer Blog (2024).**
Mastering LLM Techniques: Inference Optimization. *developer.nvidia.com/blog/mastering-llm-techniques-inference-optimization*
→ KV cache, PagedAttention, MQA/GQA, FlashAttention, quantization, speculative inference.

**NVIDIA AI Podcast (2025).**
Tokenomics, Cost per Token, and Extreme Co-Design. *youtube.com/watch?v=zNuOOMM20Tk*
→ Demand = users × sessions × tokens × multipliers; Jevons effect; co-design.

**Patel, D. — Invest Like the Best (2025).**
The Supply & Demand of AI Tokens. *youtube.com/watch?v=LF3aUIM57uw*
→ A more expensive model can be cheaper per task; rate limits, supply constraints.