Spectro Cloud

AI unit economics: why token factories need more than metering

Token metering is the invoice mechanism, but profitability comes from understanding and optimizing the full economics of AI infrastructure: GPUs, models, tenants, energy, latency, and the outcomes customers pay for.

What exactly are we trying to measure?

A lot of the market is rushing toward the same headline capability right now: meter the tokens, expose a billing API, call the result a token factory.

Tokens matter, of course. They give customers a unit they understand and a bill they can reconcile against their own work product or staff activity.

But as an AI cloud provider, a token count tells you almost nothing about whether the infrastructure underneath made or lost money.

The backdrop is uncomfortable. GPU rental gross margins sit in the mid-teens once you account for power, labor, and depreciation. The same GPUs that cost five figures each routinely run at somewhere between 13% and 33% utilization. Put those two facts together and a massive share of the most expensive hardware in your building is earning less than it costs to own. A token meter will happily bill for the work that does happen, but stay silent about the idle GPU next to it.

We believe the operators who will win the next few years will be the ones who look beyond the billing endpoint and get control of questions like: what did these tokens cost us, what did we charge, and where is the money leaking? That's AI unit economics, and token metering is just one instrument on that dashboard.

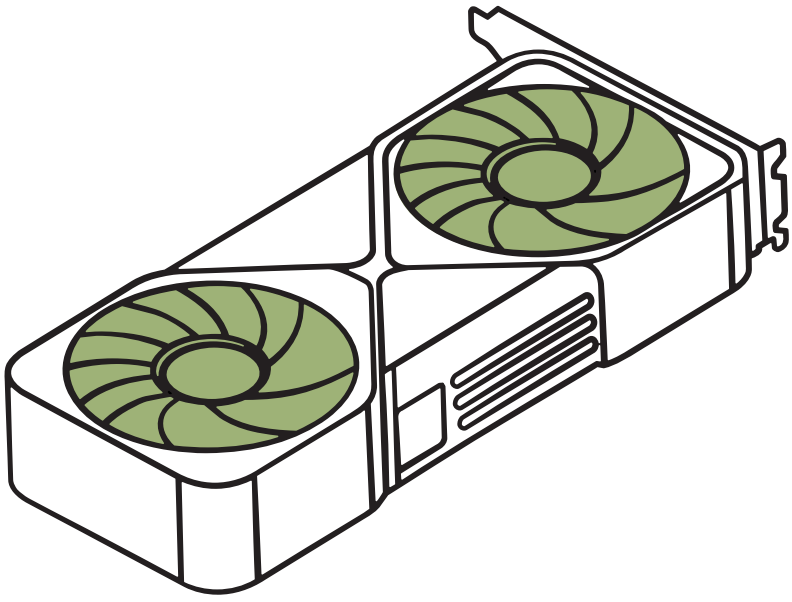
This paper makes the case for AI unit economics as the operating model for an AI cloud, sets out what a real meter reads once you look past tokens, walks through the four generations of AI billing and where the value is migrating, and maps where this capability exists in our PaletteAI platform.

Raw GPU capacity is a 14% business

Demand is the easy part (!!!) of running a neocloud. Every enterprise on the planet is asking about AI infrastructure, and the top-line numbers are enormous: Gartner expects worldwide AI spending to reach **$2.5 trillion in 2026**, up 44% year over year. McKinsey now **counts more than 100 neoclouds globally**, with only 10 to 15 operating at meaningful scale — so there's a market there for you to capture.

Turning that demand into profit is definitely not easy. The same McKinsey analysis puts GPU rental gross margins at 14–16% after labor, power, and depreciation, which is worse than a lot of grocery retail. Sell raw GPU-hour capacity in a bare-metal model and you've signed up for the economics of a high-capex hosting business with none of the pricing power.

The way out is to sell something more valuable than time on a card: governed, multi-tenant AI services with isolation, choice, and Day-2 operations included in the price. Token metering is part of how you bill for those services. It is not, on its own, how you make them profitable. A token count doesn't show you margin. It shows you revenue, only on the work that happened, and says nothing about the cost of the work that didn't.



The biggest line item is the idle GPU

Industry GPU utilization is dismal. An analysis of more than 4,000 Kubernetes clusters by Wesco put average utilization at 13%, with GPU memory rarely climbing above 20%. Even OpenAI, about as sophisticated an operator as exists, estimates its own infrastructure at roughly 33% (both figures are discussed in our earlier work on **MIG partitioning**). One-third utilization is the good end of the range.

The ironic detail for anyone building their strategy around token metering is this: an idle GPU produces zero tokens, so it appears nowhere on a token meter and nowhere on the customer's bill. The most expensive asset in the room is invisible.

The first economic question for an AI cloud, then, is about useful work rather than token pricing: how much of what we bought is doing something billable, and how do we raise that number?

The levers are scheduling and partitioning: MIG to slice a physical GPU into right-sized pieces, dynamic resource allocation to match workloads to silicon, quotas and fair-share scheduling for contended environments, and deliberate oversubscription where the workload mix allows it. When those work together, the gains are large. Combining MIG partitioning, dynamic resource allocation, and namespace-level quotas, we've seen customers push utilization on shared pools up by **as much as 70%**.

Not all tokens are created equal

A token is an excellent unit of customer pricing and a poor unit of infrastructure consumption, for four reasons:

First, models vary enormously. A million tokens served by an 8–billion-parameter model and a million served by a 70–billion-parameter reasoning model differ by an order of magnitude in GPU-seconds, memory residency, and KV-cache pressure. Same line on the invoice, wildly different cost to produce. Bill both at one rate and you're cross-subsidizing the expensive model with the cheap one, usually without knowing it.

Second, input and output tokens are not the same animal. Input tokens are processed in parallel during the prefill phase and are comparatively cheap. Output tokens are generated one at a time during decode, are bound by memory bandwidth, and dominate both latency and cost. This is why public model APIs charge more for output than input, often several times more. Reasoning and agentic workloads make the asymmetry worse, because they produce long chains of output tokens per request, the expensive kind.

Third, context length drives memory in a way token counts hide. The KV cache grows with context length across every layer of the model, so a long-context request occupies far more GPU memory, for far longer, than a short one with the same total token count. Two invoices can match to the token and represent completely different occupancy of your most constrained resource.

Fourth, batching changes everything. The same token is cheap when it batches efficiently with others and expensive when it runs nearly alone on a reserved GPU. Cost per token is partly a property of how busy the machine was at that moment, which the token count cannot express.

None of this means token metering is wrong. It means a token is a unit of value to the customer, not a unit of truth about your infrastructure. You need to meter both, really. Bill in the unit your customer understands, and measure in the units that tell you what happened.

What an AI unit economics dashboard shows

If a token is one signal, what's the rest of the set? A full meter reads GPU-seconds and GPU-memory occupancy, MIG-slice allocation, accelerator utilization and power draw in watts, KV-cache and prefix-cache hit rates, time-to-first-token and tokens per second, queue depth and wait time, input versus output tokens, request counts, cache hits, vector-database calls, agent tool invocations, and per-tenant spend. Token count is one data point in a wide table.

Render that table for a finance audience and the calculations change from infrastructure metrics into the questions a business owner cares about:

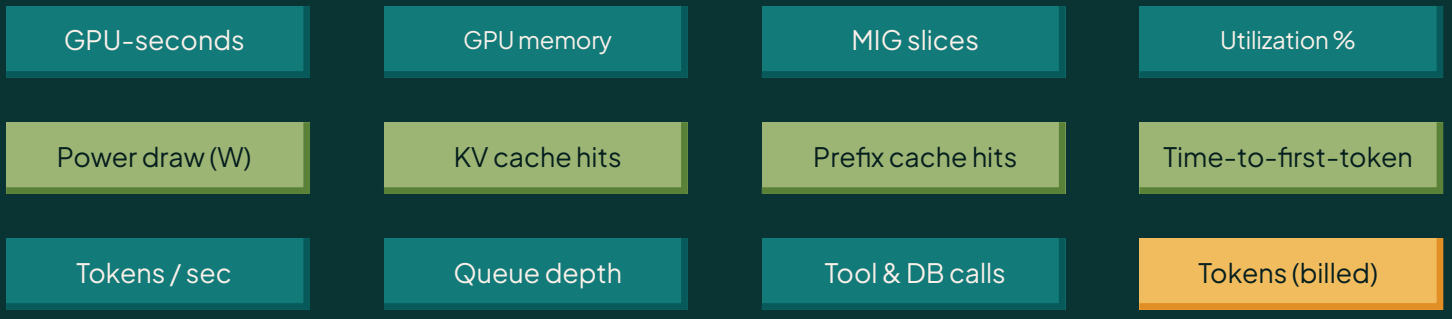
- Cost per successful inference, and cost per agent task
- Cost per conversation, per workflow, per completed outcome
- Gross margin by tenant, and which tenants are underwater
- Utilization and stranded capacity by pool
- Energy consumed per token, and per useful result
- Model efficiency and cache-hit ratio

That telemetry is the difference between knowing you served 4 billion tokens last month and knowing that one tenant's reasoning workload ran at negative margin because it never batched and never hit cache.

Much of the raw material for this already exists in your management platform (or at least it does in ours). PaletteAI surfaces fleet overviews, utilization dashboards, and metering views with Alertmanager-driven alerting, alongside per-tenant cost visibility; the inference layer exposes vLLM metrics such as KV-cache usage, prefix-cache hits, time-to-first-token and throughput, and DCGM metrics for GPU power, memory and utilization; and the routing layer tracks spend per user, team, and API key.

The work ahead is therefore less about inventing new *sensors* and more about composing the ones we have into a margin view.

The full AI economics dashboard view



Four generations of AI billing

It helps to see token billing as a stage rather than a destination.

Generation 1 is GPU-hour billing. You rent time on a card. It's simple, it's where most providers started, and it is very hard to differentiate here on anything but price.

Generation 2 is token billing. You sell output instead of time, which is a genuine improvement and where your competitors are today. But as we have explored, tokens are a noisy proxy for cost, and even on the customer end of the equation, there's no automatic relationship between tokens and business results.

Generation 3 is outcome billing. Customers stop buying tokens and start buying completed work: a resolved support case, a reviewed contract, a passing code change, a finished design iteration. Nobody ever asked how many SQL queries powered their CRM.

In time, nobody will ask how many tokens powered their support bot.
They'll ask how many cases it closed and what each one cost.

Generation 4 is policy-driven AI economics. Cost, latency, sovereignty, and energy stop being things you report on after the fact and become policies the platform optimizes against in real time, placing, routing, batching, and selecting models to hit a margin and an SLA at once. Economics becomes a control input rather than a monthly post-mortem.

The value expands upward through these generations, and so does the defensibility. Operators stuck at generations 1 and 2 compete on price. Operators who reach 3 and 4 compete on profitability, performance, and customer experience, which are far harder to undercut.

Generations 1 and 2 are here today, and token metering is part of getting generation 2 right. Generations 3 and 4 are where the platform layer is heading.

Optimization beats cheaper hardware

There's a reflex, when margins are thin, to go hunting for cheaper GPUs. The bigger savings instead should come from running the hardware you already own better. Continuous batching, KV-cache reuse, prefix caching, speculative decoding, intelligent routing, and right-sizing to smaller models can each cut the cost of a useful result without changing a single piece of capex. A GPU that serves twice the tokens through better batching is worth more to you than one that costs 20% less and sits two-thirds idle. Why shop for a cheaper H100 while the one you have is mostly asleep?

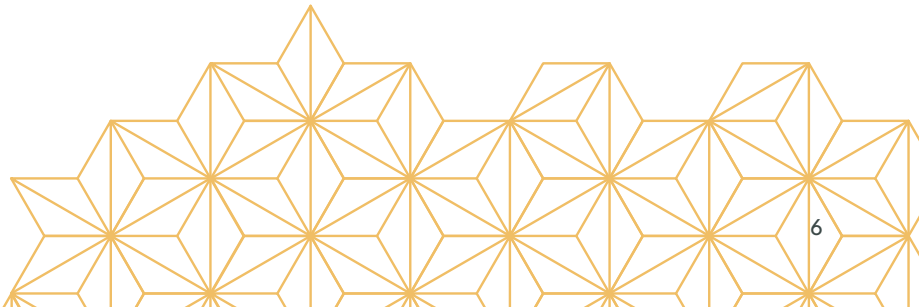
This has a direct consequence for what a platform should display. Most show tokens consumed. Perhaps the more useful number is cost avoided: what did prefix caching save this week, how much did routing overflow to cheaper capacity reduce the bill, what did right-sizing a workload to a smaller model do to its cost per task? In engineering, optimization is almost always a thankless, invisible task, but when so much money is tied up in GPUs, there's an opportunity here to turn engineering virtue into a dollar sign your CFO appreciates.

The platform already exposes the signals this depends on: prefix-cache hit rates, KV-cache usage and throughput from the inference layer, and token-aware routing that caps an expensive cluster at a token-per-minute ceiling and spills overflow to cheaper capacity rather than failing the request. Turning those into a standing cost-avoided view is the natural next step.

Billing asks what to charge; operations asks why it cost that much

Two questions sit at the heart of an AI cloud, and they belong to different systems. Billing answers "what should I charge?" It wants a clean, customer-legible unit — tokens, requests, outcomes — and not much else. Operations answers "why did this inference cost what it did, and how do I make it cost less?" It wants the full telemetry: utilization, batching, cache behavior, model choice, placement.

These systems serve different people. Billing serves finance and the customer. Operations serves platform engineering and, ultimately, the margin. Token metering does a fine job of the first. AI unit economics does the second, and feeds the first with the cost data that makes pricing a decision rather than a guess.



The foundations are already here

We've been building the foundations of AI unit economics into our PaletteAI platform from day one.

These include multi-tenancy with tenant and project isolation, RBAC tied into existing identity, GPU quota at tenant and project scope, admission control, MIG partitioning, dynamic resource allocation, and fair-share scheduling that give you control over who consumes what.

Fleet overviews, utilization dashboards, and metering views with Alertmanager alerting, combined with per-tenant cost visibility, give you the operational data to run a P&L against.

The inference layer adds the vLLM and DCGM signals, and the routing layer tracks spend per user, team, and key. Customers combining partitioning, dynamic allocation, and quotas have pushed shared-pool utilization up by as much as 70%.

We're not done yet. For example, we're building token-level metering and quota in Launchpad for AI, our new standalone AI appliance designed to solve the tokenomics problem for enterprise dev teams.

The model is three quota types working together — a token quota (total tokens per user or application), a rate limit (requests per minute or hour), and an access quota (total requests or tokens over a billing period) — with operator-chosen enforcement at the limit (throttle, block, or alert) and usage broken down by user, application, and model. This is generation 2 done properly, sitting on top of the existing governance.

Where this goes next is the composition work: stitching utilization, per-tenant cost, inference telemetry, and spend into a single margin-by-tenant and cost-per-outcome view, and then letting policy — cost, latency, sovereignty, energy — become an input the platform optimizes against rather than a report it produces. That's generations 3 and 4, and we'd ask you to stay tuned!

From flying blind to policy-driven

When it comes to AI unit economics and token metering, we can place AI cloud operators on a short maturity model.

- **Stage 0: flying blind.** You bill GPU-hours, can't attribute cost per tenant, and treat utilization as best-effort.
- **Stage 1: metered.** Tokens and GPU usage are counted per tenant. You can produce an invoice and defend it.
- **Stage 2: visible.** Utilization, per-tenant cost, and inference telemetry are on one dashboard. You can join the dots from activity to your margin.
- **Stage 3: optimized.** Routing, batching, caching, and partitioning actively reduce cost per useful result, and you can show cost avoided as a KPI.
- **Stage 4: policy-driven.** Economics, latency, sovereignty, and energy are policies the platform optimizes against in real time. You price and place by outcome.

Most of the market sits at stage 0 or 1. Getting to stage 2 or 3 is best practice. Stage 4 is the horizon we're all building toward. Where would you place yourself?

The bottom line

In our opinion, the token economy is real, and metering tokens is part of competing in it. But tokens are the invoice. The operators who treat AI infrastructure as a business to be optimized, rather than capacity to be rented, are the ones who'll still be here in 2028, and they'll be the ones setting prices, while everyone still counting tokens argues about them.

We believe you should treat utilization as your first economic metric; it's the largest, most fixable cost you have. Keep your billing unit simple for the customer and your operational telemetry rich for yourself, and don't mix the two.

Talk to us

If you're an AI cloud operator and you're keen to optimize your utilization and explore token metering, let's have a conversation.

Reach out to book a meeting with an AI expert at **spectrocloud.com/get-started**.