THE TOKEN AND GPU-CAPACITY BUSINESS

An End-to-End P&L Primer: Pricing, Metering, Billing, Settlement, and Revenue Assurance

---

Perspective: a GPU cloud / capacity provider (a "neocloud") that sells raw GPU capacity under committed and on-demand contracts, and a token-metered inference layer on top of the same fleet. Practitioner depth, with illustrative worked numbers drawn from public market benchmarks as of mid-2026. Prepared August 2026 for Ilan Gleiser.

Executive Summary

A GPU cloud is, financially, a leasing business wearing a software company's clothes. The provider raises capital, converts it into a rapidly depreciating asset (GPU servers plus the power, cooling, and network needed to run them), and then sells time on that asset in two very different wrappers: capacity, priced per GPU-hour and sold largely through multi-year take-or-pay contracts, and tokens, priced per million tokens of model input and output and sold through a usage-metered API. The first wrapper looks like commercial real estate or aircraft leasing; the second looks like a telecom selling minutes. The whole P&L lives or dies on three numbers: the price achieved per GPU-hour (or its token-denominated equivalent), the utilization of the fleet, and the depreciation life assumed for the hardware.

The middle of the business — metering, rating, billing, settlement — is where reported revenue is actually manufactured, and it is where money silently leaks. Every GPU-second and every token must be captured as an event, aggregated correctly, rated against the right price book, invoiced, collected, and recognized under the right accounting treatment. Telecom operators learned decades ago that 1–3% of revenue routinely disappears between the network switch and the invoice; usage-metered compute has exactly the same anatomy and exactly the same failure modes. Revenue assurance — the discipline of proving that everything used was billed and everything billed was correct — is therefore not a back-office nicety but a direct, high-margin lever on the bottom line: a recovered dollar of leakage carries no incremental COGS.

This primer walks the full chain: how the two product lines are priced (Section 2), what the cost stack underneath looks like (Section 3), two worked P&Ls — a 1,024-GPU capacity cluster and a token-serving endpoint (Sections 4–5), how metering pipelines are built and where they break (Section 6), how rating and billing turn meters into invoices (Section 7), how cash settlement and revenue recognition work, including take-or-pay backlog accounting (Section 8), and a practical revenue-assurance control framework with KPIs (Section 9). Section 10 assembles the full income statement and reconciles it to what public comparables like CoreWeave actually report.

1. The Business in One Picture

One asset base feeds two revenue engines. The asset base is a fleet of accelerated servers — today typically NVIDIA H100/H200 and Blackwell-class GB200/B200 systems — sitting in leased or owned data-center capacity with committed power. Above it sit two commercial wrappers:

- **The capacity business (GPU-hours) —** the provider sells dedicated GPU instances or whole clusters, metered in GPU-hours (usually rated per second or per minute). The dominant commercial form is the multi-year *take-or-pay* reservation: the customer commits to pay for a fixed quantity of capacity whether or not it is used, often 1–5 years, sometimes with prepayments. On-demand and spot capacity monetize whatever the committed book leaves stranded.
- **The token business (inference-as-a-service) —** the provider runs models (its own, open-weight, or the customer's fine-tunes) behind an API and charges per million input tokens and per million output tokens, with distinct rates for cached input and batch processing. Here the provider absorbs the utilization risk that the capacity customer would otherwise carry — and prices for it.

The strategic difference between the two is who owns the utilization problem. In the capacity business, a take-or-pay contract transfers utilization risk to the customer: revenue is contractual, visibility is measured in years (CoreWeave disclosed a $99.4 billion revenue backlog at March 31, 2026), and the provider's residual risks are counterparty credit, delivery milestones, and residual value at contract end. In the token business the provider retains utilization risk — tokens only bill when traffic arrives — but earns a substantially higher achieved price per GPU-hour when traffic is dense, because one GPU-hour can be resold many times over as batched, multiplexed token throughput. A well-run operator uses the token layer as a yield-management overlay on capacity the reserved book does not fully absorb.

Financially, the two lines also differ in revenue quality. Capacity revenue under committed contracts is recognized ratably as the capacity is made available, is backed by remaining performance obligations, and supports debt financing against contracted cash flows. Token revenue is pure usage revenue: recognized as consumed, re-priced frequently (token prices have deflated sharply every year as hardware and software efficiency compound), and far more exposed to competition. Rating agencies, lenders, and equity analysts will price the two streams very differently, which is why disclosure of backlog, RPO, and contract mix matters so much in this sector.

2. Pricing

2.1 Capacity pricing: reserved, on-demand, spot

Capacity pricing is a curve, not a number. The same physical H100-hour trades at wildly different prices depending on commitment length, cluster quality (interconnect, storage, support), and channel. Illustrative market benchmarks as of mid-2026:

Product form	Typical H100 price / GPU-hr	Notes
Spot / interruptible	$0.55 – $1.70 (avg ≈ $1.69)	Preemptible; monetizes stranded capacity
On-demand, neocloud	$1.89 – $2.99	UpCloud $1.89, RunPod $1.99–2.59, Hyperstack $2.50
On-demand, hyperscaler	$5.99 – $6.98	AWS p5 $6.88, Azure NC-H100 $6.98
1-yr reserved contract	$1.70 – $2.35	Late-2025/early-2026 market range

Product form	Typical H100 price / GPU-hr	Notes
Multi-yr take-or-pay cluster	negotiated (≈ $1.50 – $2.50 eff.)	Anchor deals; milestone delivery; prepayments
Market average (all forms)	≈ $3.65	Reserved avg ≈ $3.52 across listings

Sources: getdeploying.com H100 price index (48+ providers); zettabyte.space neocloud unit-economics analysis. Prices move quickly; treat as illustrative.

Three structural features of capacity pricing deserve emphasis. First, the commitment discount is really a utilization-risk transfer price: the gap between $6.98 on-demand at a hyperscaler and ~$2.00 on a one-year neocloud contract is what the market charges for holding idle-capacity risk, plus brand, ecosystem, and enterprise-compliance premia. Second, generation transitions reprice the whole curve: when Blackwell arrives at scale, H100 contract prices sag, which is why depreciation life and contract tenor must be jointly managed — you do not want a 6-year depreciation schedule against a fleet whose market rate halves in year 3. Third, large deals are increasingly structured, with delivery milestones (revenue starts when capacity is accepted), ramp schedules, prepayments that fund capex, and sometimes seller financing or vendor equity — all of which complicate both revenue recognition and cash settlement (Section 8).

2.2 Token pricing

Token pricing is a rate card per million tokens, always asymmetric between input and output. Output tokens are priced 3–5× input because generation is sequential (one forward pass per token) while input prefill parallelizes efficiently across the sequence. Illustrative rate-card points as of July–August 2026:

Model / tier	Input $/M tok	Output $/M tok	Notes
Frontier (GPT-5.6 Terra)	$2.50	$15.00	Premium closed model
Frontier (Claude Sonnet 5)	$2.00	$10.00	Cached input ≈ 10% of list
Fast tier (Gemini 2.5 Flash)	$0.30	$2.50	Flash-Lite input from $0.10
Value tier (DeepSeek-class)	$0.14 – $0.55	$2.19	Cached input as low as $0.0028
Open-weight small (Together/Groq)	$0.03 – $0.08	$0.08 – $0.12	Commodity floor

Sources: tldl.io verified LLM pricing tracker (July 2026); introl.com inference unit-economics guide. Cached-input discounts of ~90% and batch-API discounts of ~50% are now standard.

The rate card has grown real microstructure: cached-input pricing (typically ~10% of list) rewards prompt reuse and materially changes both revenue per request and metering requirements, since the biller must now distinguish cache hits from cache misses per request; batch pricing (typically 50% off) sells latency flexibility, which is to the token business what interruptible spot is to the capacity business; and long-context surcharges, image/audio token conversion rates, and per-request minimums round out the card. Every one of these distinctions is a metering obligation: a price you cannot meter is a discount you did not intend.

2.3 Price architecture: the price book as a controlled object

Between list price and invoice sits the commercial machinery: enterprise discounts, committed-spend deals (customer commits to $X over a term, drawn down against usage at discounted rates), prepaid credit packs, free tiers, promotional credits, and partner/reseller margins. The operational requirement is that all of this lives in a versioned, dated, access-controlled price book that the rating engine consumes — not in spreadsheets, sales emails, or hand-edited invoice adjustments. Every revenue-assurance program that finds material leakage

finds some of it here: an expired discount still applied, a negotiated rate never loaded, a currency mis-set, a free tier without an enforcement cap. Treat rate-table changes with the same change-management rigor as production code deployments, because that is exactly what they are.

3. The Cost Stack

The cost side of the P&L is dominated by capital recovery. A useful mental model: roughly half to two-thirds of the fully-loaded cost of a GPU-hour is depreciation and financing on the hardware itself; power, facilities, network, and people share the remainder.

- **Capex —** an 8×H100 HGX server runs ≈ $300k; network fabric (InfiniBand/RoCE), storage, and integration add 15–25%, landing all-in cluster capex around $40–48k per GPU. Blackwell-class systems are substantially more per GPU but deliver more throughput per dollar.
- **Depreciation life —** the single most consequential accounting judgment in the sector. The same server generates ≈ $50k/yr of depreciation on a 6-year life and ≈ $75k/yr on a 4-year life. Long lives flatter EBIT today at the cost of overstating the asset's late-life earning power; the market debate over 5–6 year GPU lives versus 3–4 year competitive relevance is, in essence, a debate about whether reported margins in this sector are real.
- **Power and facilities —** an H100 draws ~700W (B200 up to ~1,000W); a dense rack draws 40–80kW, 3–8× a conventional enterprise rack. At a 1.3 PUE and $0.09/kWh, power is roughly $0.15 per GPU-hour; colocation space and cooling, priced per kW-month, add roughly $0.20–0.25.
- **Financing —** fleets are heavily debt-financed against contracted cash flows. CoreWeave carried $24.9B of debt at Q1 2026 with net interest expense of $536M in the quarter — over 25% of revenue — with recent facilities priced around SOFR+2.25% floating / ~5.9% fixed, against earlier GPU-backed facilities in the 8–11% range. Interest sits below EBITDA but is utterly real cash: a debt-financed fleet breaks even in the high-$1s per GPU-hour before a dollar of profit.
- **People and platform —** site reliability, datacenter ops, support, and the software layer (scheduler, storage, billing itself). Small per GPU-hour (≈ $0.10 in the worked example) but scales with product complexity — the token business carries meaningfully more engineering cost than raw capacity.

4. Worked Example A: 1,024-GPU Capacity Cluster P&L

Assumptions: 128 servers × 8 H100 = 1,024 GPUs; $300k/server plus 18% for fabric, storage, and integration → $45.3M capex ($44.3k/GPU); 5-year straight-line depreciation; 1.275 kW IT load per GPU, PUE 1.3, $0.09/kWh; colocation at $135/kW-month on IT load; ops and support at $0.10/GPU-hr; 70% debt at 8.0% interest-only for simplicity. The entire cluster is sold under a take-or-pay reservation at an effective $2.10/GPU-hr, so billable hours = 8,760 × 1,024 = 8.97M GPU-hours regardless of customer utilization.

Line item	$/GPU-hr	$M / year	% of revenue
Revenue (take-or-pay @ $2.10)	2.100	18.84	100.0%
Power (1.275 kW × 1.3 PUE × $0.09)	(0.149)	(1.34)	7.1%
Colocation ($135/kW-mo)	(0.236)	(2.12)	11.2%
Ops, support, platform	(0.100)	(0.90)	4.8%
EBITDA	1.615	14.49	76.9%

Line item	$/GPU-hr	$M / year	% of revenue
Depreciation (5-yr SL on $45.3M)	(1.010)	(9.06)	48.1%
EBIT	0.604	5.42	28.8%
Interest (70% LTV @ 8.0%)	(0.283)	(2.54)	13.5%
Pre-tax income	0.322	2.89	15.3%

Cluster-level illustration. EBITDA margin of ~77% at full contract coverage compares to CoreWeave's reported 56% adjusted EBITDA margin in Q1 2026 (company level, diluted by ramping capacity, on-demand mix, and opex).

Read the sensitivity, because it is the whole business. At $2.10 with full take-or-pay coverage this cluster earns a 15% pre-tax margin and pays back its full capex in about 3.8 years of pre-principal cash flow — uncomfortably close to the useful life. Move the achieved price to $1.70 (the bottom of the 2026 contract range) and pre-tax income falls to roughly minus $0.7M: below breakeven after financing. Shorten depreciation to 4 years at $2.10 and EBIT drops from $5.4M to $3.2M. Leave 20% of the cluster unsold (no take-or-pay, merchant exposure) and EBITDA falls by ~$3.8M against a fixed cost base. The equity case is leveraged to price, tenor, and utilization simultaneously — which is precisely why the committed backlog, not the GPU count, is the asset that lenders underwrite.

5. Worked Example B: Token Unit Economics on the Same Fleet

Now run an inference endpoint on one 8×H100 node from the same fleet, charged internally at the $2.10/GPU-hr transfer price → $16.80/node-hr. Serving a 70B-class open-weight model with tensor parallelism across the node, a well-tuned stack (continuous batching, paged KV-cache, speculative decoding) sustains ≈ 10,000 output tokens/second aggregate at healthy batch depth, and prefill throughput roughly 25× that for input tokens. Utilization here means traffic density: the fraction of theoretical token throughput actually sold across the day's peaks and troughs.

Traffic utilization	Output tokens sold / node-hr	Cost / 1M output tok	Margin @ $2.19/M	Margin @ $10/M
30%	10.8M	$1.56	29%	84%
50%	18.0M	$0.93	57%	91%
60%	21.6M	$0.78	64%	92%
85%	30.6M	$0.55	75%	95%

Input tokens cost ≈ $0.03/M to serve at 60% utilization against list prices of $0.14–$2.50/M — richly profitable, which is why cached-input discounts of 90% are commercially survivable.

Two lessons fall out. First, utilization is the margin: the identical node produces a 29% or a 75% gross margin at the same rate card depending purely on traffic density, which is why serverless token providers obsess over routing, multiplexing, multi-tenancy, and autoscaling-to-zero. Second, the token wrapper can dramatically out-earn the capacity wrapper on the same silicon: at 60% utilization and even the value-tier $2.19/M output price, the node generates ≈ $47/hr of token revenue against a $16.80 internal capacity cost — an achieved price of ~$5.90 per GPU-hour, nearly 3× the take-or-pay rate. That spread is the reward for absorbing utilization risk and operating a much more complex metering, serving, and billing stack. It is also why token prices deflate relentlessly: every efficiency gain (quantization ~50%, speculative decoding 2–3× latency, better batching —

compounding to ~16× versus naive deployment per Introl's 2026 analysis) becomes price competition within quarters.

6. Metering: Turning Physics into Billable Events

Metering is the sensory system of the P&L. Everything downstream — invoice, revenue, cash, audit — is a transformation of metering events, so an unmetered event is unbillable revenue lost forever, and a double-metered event is a dispute, a credit memo, and reputational damage. The two product lines meter very different things:

Product	Billable meters	Primary source of truth
Capacity	GPU-seconds by SKU; instance-hours; storage GB-mo; egress GB; IP/API extras	Scheduler/control-plane allocation ledger, cross-checked vs node telemetry
Token API	Input tokens; output tokens; cached-input tokens; batch vs interactive; per-model rates; images/audio	API gateway request logs with tokenizer counts, cross-checked vs serving-engine counters
Both	SLA downtime (credit meters); committed-spend drawdown; free-tier consumption	Incident system; billing ledger

The reference pipeline has five stages, and modern usage-billing platforms (Lago, Metronome, Orb, and in-house equivalents at scale players) all converge on the same shape: (1) event capture at the source of truth, emitting one immutable usage event per billable action with customer ID, resource, quantity, and timestamp; (2) ingestion through a durable, idempotent stream — every event carries a unique transaction ID so retries, replays, and race conditions cannot double-bill (production systems ingest up to ~1M events/second); (3) aggregation into billable metrics per customer per period (SUM of GPU-seconds, SUM of output tokens, MAX of concurrent instances, COUNT of requests); (4) rating against the versioned price book (Section 7); and (5) storage of both raw events and rated results, because disputes are settled by replaying raw events, not by arguing about aggregates.

Where metering breaks, in practice: late events (a node agent buffers during a network partition and flushes after the billing period closed — you need a documented late-event policy and a re-rating window); duplicate events (idempotency keys must be enforced at ingestion, not trusted from emitters); clock skew between nodes assigning events to the wrong period; lost events when agents crash — which is why capacity metering should be derived from the allocation ledger (intent) and reconciled against node heartbeats (reality), never from a single fragile source; orphaned resources that keep metering after the control plane thinks they are gone (or worse, stop metering while still consuming power); and tokenizer drift in the token business — the tokenizer version is billing-critical configuration, because a silent tokenizer change re-prices every request by a few percent in one direction or the other. Every one of these failure modes has a matching revenue-assurance control in Section 9.

7. Billing: Rating, Invoicing, and the Contract Layer

Rating converts aggregated usage into money: quantity × applicable rate, walked through the discount waterfall (list → volume/tier → negotiated rate → committed-spend discount → promotional credits) in a deterministic, versioned, replayable order. The non-negotiable engineering property is reproducibility: given the same events and the same price-book version, the rating engine must produce the same invoice, byte for byte, months later. Without that, disputes and audits become archaeology.

Invoicing then assembles rated usage into the customer-facing document: usage lines by SKU and model, committed-fee lines, prepaid-credit drawdowns, SLA credits, proration for mid-period starts and stops, and

taxes (which, for compute sold across borders, are a genuine sub-specialty — place-of-supply rules, VAT reverse charges, and US state cloud-tax variation all attach at invoicing). Dunning — the retry-and-escalate machinery for failed payments — matters enormously in the self-serve token business where cards fail constantly, and barely at all in the enterprise capacity business where the risk is concentrated slow-pay instead.

The contract layer adds structures that pure usage billing does not have. Take-or-pay invoices bill the committed quantity monthly or quarterly regardless of consumption, with usage above commitment billed as overage and usage below simply forfeited (or, in some contracts, banked into a limited rollover — a term with direct revenue-recognition consequences). Committed-spend deals invert this: the customer prepays or commits to $X, and every month's rated usage draws the balance down, with true-up invoicing if the term ends short. Milestone billing governs new-build clusters: invoicing begins per contract when capacity passes acceptance tests, so delivery-date slippage is directly a revenue event. SLA credits flow the other way — downtime beyond the committed availability generates credits against the next invoice, making incident metering a billing input. Each of these structures is a place where the invoice can drift from the contract, and each therefore appears again in the revenue-assurance control set.

8. Settlement and Revenue Recognition

8.1 Cash settlement

Settlement is the unglamorous conversion of invoices into cash, and its texture differs completely across the two lines. The token/self-serve business settles like consumer SaaS: card-on-file or prepaid credits, processor fees of 2–3%, chargebacks, and fraud — of which the sector's signature is stolen-card GPU consumption (often for crypto mining or resale), where the provider eats both the chargeback and the burned capacity. Prepaid credits deserve special respect: they are customer liabilities (deferred revenue) until consumed, they age, and their breakage (expiry unconsumed) is recognizable revenue only under defined conditions. The enterprise capacity business settles like industrial leasing: invoices on net-30/60 terms, wires, occasional letters of credit or parent guarantees on large take-or-pay books, and DSO management. Concentration is the defining risk: when a handful of AI labs and hyperscalers are most of your backlog — CoreWeave's disclosed counterparties include Meta (a $21B commitment), Anthropic, Cohere, Jane Street, and Mistral — receivables risk is really single-name credit risk, and the finance function should treat it with credit-committee discipline, not accounts-receivable routine.

Two further settlement webs surround the core. Channel settlement: capacity and tokens sold through hyperscaler marketplaces, brokers, aggregators, and resellers arrive net of channel fees (marketplace listing fees, reseller margins), requiring reconciliation of the platform's settlement report against your own metering — an inter-company version of the meter-to-bill problem. Supplier settlement: on the cost side sit take-or-pay obligations of your own — colocation and power contracts with minimum commitments, and revenue-share arrangements with datacenter partners — so the provider is simultaneously a seller and a buyer of committed capacity, and a mismatch in tenor between the two books (long supplier commitments against short customer contracts, or vice versa) is a structural P&L risk that belongs on the CFO's dashboard.

8.2 Revenue recognition (ASC 606 view)

Usage revenue — on-demand GPU-hours and tokens — is the easy case: recognized as consumed, in the period metered. Take-or-pay capacity is recognized ratably as the stand-ready obligation is satisfied, i.e., as capacity is made available, whether or not the customer uses it; unbilled committed amounts sit in remaining performance obligations (RPO), and amounts billed or prepaid ahead of availability sit in deferred revenue. This

is why backlog is the sector's favorite disclosure — CoreWeave's $99.4B revenue backlog is contracted future revenue awaiting delivery — and also why it deserves scrutiny: backlog converts to revenue only as fast as data centers are energized and clusters pass acceptance, and it can be repriced, renegotiated, or (in credit stress) defaulted. Variable consideration cuts revenue at recognition, not just at invoicing: expected SLA credits and expected credit-memo rates should be estimated and netted, so a provider with chronic incidents recognizes less revenue than its meters would suggest. Prepayment breakage, milestone acceptance timing, and rollover rights in take-or-pay deals are the recurring judgment areas an auditor will probe.

9. Revenue Assurance

Revenue assurance answers two questions continuously: is everything we delivered being billed (completeness), and is everything we billed correct (accuracy)? Telecom, which invented the discipline, consistently found 1–3% of revenue leaking before controls; usage-metered compute has the same anatomy — high-volume events, complex rating, negotiated exceptions — and no reason to expect a better baseline. At a 56% EBITDA margin, recovering one point of revenue leakage is worth roughly two points of EBITDA growth for zero incremental COGS.

9.1 Where the money leaks

Leakage mode	Mechanism	Typical detection
Unmetered usage	Agent crash, orphaned allocation, event loss in pipeline	Capacity ledger vs billed-hours reconciliation
Meter-to-bill gap	Events captured but dropped/late in aggregation or rating	Event-count completeness checks per stage
Rating errors	Wrong/stale price book, expired discount still live, FX error	Invoice re-rating; rate-table change audit
Entitlement drift	Free tier uncapped, internal/test usage on prod SKUs	Entitlement audit; margin-by-customer outliers
SLA over-crediting	Credits granted beyond contractual formula	Credit memo vs incident-record match
Fraud	Stolen cards, abuse of free tiers, resale of discounted capacity	Velocity checks, KYC, anomaly detection
Contract leakage	Overage never billed; rollover mis-tracked; milestone billing missed	Contract-to-invoice audit per account
Tokenizer/meter drift	Counting change silently re-prices all requests	Golden-request regression tests on billing

9.2 The control framework

The backbone control is the three-way match, run daily and automated: (1) what the control plane says was allocated (the intent ledger), against (2) what telemetry says actually ran (node heartbeats, serving-engine counters), against (3) what the billing system rated and invoiced. Discrepancies land in a workflow queue with materiality thresholds, root-cause categories, and closure SLAs — exactly like a trading firm's breaks process. Around that backbone: completeness checks that event counts reconcile across every pipeline stage; re-rating audits that replay a sample of invoices from raw events against the price book; change control on rate tables and tokenizer versions with four-eyes approval and effective-dating; entitlement reviews that catch uncapped

free tiers and forgotten discounts; margin-by-customer monitoring, because a customer whose gross margin quietly went negative is either mis-rated or mis-contracted; and credit-memo governance, since discretionary credits are the softest channel through which revenue leaks. The token business adds golden-request tests: a fixed corpus of requests run through the full meter-rate-bill path on every deploy, with billed amounts asserted to the cent.

9.3 KPIs for the revenue-assurance function

KPI	Definition	Healthy zone
Leakage rate	Identified leakage / total revenue, annualized	< 0.5% mature; 1–3% typical at start
Meter-to-cash yield	Cash collected / value of metered usage at contract rates	> 98%
Billing accuracy	1 − (credit memos from billing error / billed revenue)	> 99.5%
Recovery rate	Leakage recovered / leakage identified	> 70%
Dispute rate & aging	Disputed invoice value / billed; days to resolve	< 1%; < 30 days
DSO by segment	Receivables days, split enterprise vs self-serve	Enterprise < 60; self-serve < 5
Unbilled usage aging	Metered-but-uninvoiced value by age bucket	No bucket > 1 cycle
SLA credit ratio	SLA credits / revenue	< 0.5% and matched to incidents

10. The Assembled P&L

Putting the pieces together, the income statement of a combined capacity-and-token provider reads as follows, with the operational system that feeds each line in parentheses: Revenue splits into committed capacity (ratable recognition off the contract and delivery ledger), on-demand/spot (metered GPU-seconds, rated), and token/inference (metered tokens, rated), net of estimated SLA credits and credit memos (variable consideration). Cost of revenue carries power, colocation and datacenter opex, cluster support, and — under the dominant convention — depreciation of revenue-generating hardware, making reported gross margin a direct function of the depreciation-life judgment. Operating expense holds platform engineering, sales, G&A, and the billing/assurance function itself. Below EBIT, interest expense on the fleet financing is the line that separates the sector's cheerful adjusted-EBITDA story from its GAAP reality.

The public reference point makes the shape concrete. In Q1 2026 CoreWeave reported revenue of $2.08B (+112% YoY), adjusted EBITDA of $1.16B (56% margin, down from 62% a year earlier), an operating loss of $144M (−7% margin), and a net loss of $740M (−36% margin), with D&A of $1.15B and net interest expense of $536M in the quarter against $24.9B of debt and $7.7B of quarterly capex. That is the sector's margin bridge in one line: a 56-point EBITDA margin becomes a negative operating margin once the quarter's depreciation is charged, and a deeply negative net margin once the financing of the fleet is paid — while a $99.4B backlog argues that today's losses are the funded construction phase of tomorrow's contracted revenue. Whether that argument closes depends on exactly the variables this primer has priced: achieved $/GPU-hour versus the cost stack, real hardware life versus depreciation schedules, utilization of the uncommitted fleet, counterparty credit in a concentrated backlog — and a metering-to-settlement chain tight enough that the revenue actually contracted is the revenue actually collected.

P&L line	Fed by	Key risk / control
Committed capacity revenue	Contract ledger + delivery milestones	Acceptance timing; counterparty credit
On-demand & spot revenue	GPU-second metering → rating	Completeness; orphaned allocations
Token revenue	Token metering → rating	Tokenizer drift; cache-hit classification
(SLA credits & memos)	Incident system; credit governance	Over-crediting; estimation of variable consideration
Power & facilities	Telemetry + supplier invoices	Supplier take-or-pay mismatch
Depreciation	Fixed-asset register	Life assumption vs competitive life
Interest expense	Debt schedule	Rate, covenants, contracted-cashflow coverage

A closing heuristic for the operator: the capacity business is won at contract signature (price, tenor, credit), the token business is won in the serving stack (utilization, throughput per dollar), and both are kept in the metering-to-settlement chain. Pricing sets the ceiling on the P&L; the cost stack sets the floor; metering, billing, settlement, and revenue assurance determine how much of the space between them you actually keep.

Sources and Further Reading

- CoreWeave Q1 2026 earnings press release (SEC EDGAR): revenue, adjusted EBITDA, backlog, debt, capex, interest — sec.gov/Archives/edgar/data/1769628/000176962826000220/coreweave1q26earningspress.htm
- GetDeploying H100 cloud price index across 48+ providers — getdeploying.com/gpus/nvidia-h100
- Zettabyte, "The unit economics of debt-financed GPU clouds" — zettabyte.space/blog/neocloud-unit-economics-gpu-cloud
- TLDL verified LLM API pricing tracker (July 2026) — tldl.io/resources/llm-api-pricing
- Introl, "Inference Unit Economics: The True Cost Per Million Tokens" — introl.com/blog/inference-unit-economics-true-cost-per-million-tokens-guide
- Lago, "How to architect billing systems to power usage-based pricing" — getlago.com/blog/architect-billing-systems

Worked examples are illustrative constructions from the benchmarks above, not any specific company's economics. This document is general information, not investment, accounting, or legal advice.