



DCSAI Technologies

TECHNICAL WHITEPAPER · v1.0 · MAY 2026

DCS Compute

A two-sided GPU network — rent or earn

The economics, mechanics, and engineering of decentralised compute for AI workloads.

Abstract

DCS Compute is a two-sided market for GPU compute. On one side, renters (anyone with a workload — inference, training, image-gen, agent runs) pay credits to access GPUs by the second. On the other, workers (anyone with an idle GPU — a gaming rig, a mining farm, an underutilised datacenter cluster) earn credits by running those workloads. The dispatch layer in the middle matches them in under 50 ms using a reverse-auction algorithm that incorporates benchmark tier, real-time reputation, and sovereignty constraints.

The economics are simple: renters pay X credits, workers receive $0.95X$, DCS keeps a 5% spread. There are no subscription fees, no minimum commitments, and no markup on the underlying GPU rate. A renter paying \$0.39/hr for a 4090 sees that exact rate on the catalog and the worker earns 120 cr/hr (the same amount in different units — credits are a \$0.008-pegged unit of account).

Every transaction is settled in real time and emits a signed receipt (R+1 timestamp + R+2 provenance, per the DCS Standards). When a regulator asks "which jobs ran on which GPU at which time," the answer is a single CSV export, signed by the dispatch service, that the regulator can verify offline.

Who this document is for

Founders building AI products who want to understand the GPU compute cost model. Operators with spare GPU capacity who want to monetize it. Compliance officers evaluating compute providers for regulated workloads. Engineers integrating DCS Compute into their dispatch pipelines.

Contents

1	Introduction — the GPU supply chain in 2026	4
2	Market Design — why two-sided beats centralised	7
3	Worker Onboarding + Lifecycle	10
4	Benchmark + Tier System	13
5	Dispatch Algorithm — the reverse auction	16
6	Sovereignty + Jurisdictional Routing	20
7	Credit Economics	23
8	Payout System	26
9	Reputation + Reliability	28
10	Dispute Resolution	30
11	Performance Benchmarks	32
12	Security Model	34
13	Comparison vs. RunPod, Vast.ai, AWS, Render	36
14	Implementation Guide	40
15	References	44

1. Introduction — the GPU supply chain in 2026

The supply of AI-grade GPUs is concentrated in a way no other compute resource has ever been. As of mid-2026, four entities (Microsoft, Google, Amazon, Meta) control approximately 78% of all H100-class capacity in the world. Their pricing is opaque, their allocation is rationed via long enterprise contracts, and their supply has been unable to meet demand for nearly three consecutive years. Meanwhile, the long tail — perhaps two million idle gaming GPUs, half a million ex-crypto mining rigs, and several hundred thousand under-utilised datacenter cards — sits dark.

DCS Compute exists to close that gap. The premise is straightforward: if a workload can run on any 4090 or A100 (and the majority of AI inference workloads can), then it should not matter whether that 4090 lives in an AWS region or in a basement in Lisbon. The price should be set by the market, the dispatch should be done by software, and the trust should come from cryptographic receipts rather than from the brand of the hyperscaler.

1.1 Why the existing alternatives fall short

Hyperscalers (AWS, Azure, GCP)

Three problems. First, list prices are 3–8× the spot rate the same GPU clears at on smaller markets. Second, allocation is rationed: ask AWS for 100 H100s today and you will be told to wait six months. Third, on-demand instances are billed per hour, not per second, so a 90-second job pays for 3,600 seconds. None of these are technical limitations — they are pricing decisions.

GPU-cloud specialists (RunPod, Lambda, Vast.ai)

Better pricing than hyperscalers and per-second billing. But almost no audit trail — you get a log file at best, and the log is hosted by the vendor. Sovereignty constraints are not enforceable (the platform cannot guarantee your job ran in the EU even if you asked for it). And worker reputation is opaque — you find out your dispatched worker is slow only after the job finishes.

Render Network, Akash, io.net

Decentralised approaches solve sovereignty (workers self-declare) and pricing transparency. But job dispatch is slow (Render: ~6s; Akash: ~30s for a full deployment cycle), the token-based payment systems add operational friction for non-crypto-native renters, and audit trails are on-chain (which means either expensive or public, neither of which works for enterprise).

1.2 The DCS approach

DCS Compute takes a deliberately hybrid stance:

- **Off-chain dispatch, on-chain settlement.** Job matching happens in fast traditional infrastructure (Postgres + Redis); credit settlement happens on a permissioned ledger that the platform anchors to a public chain every 24 hours for tamper-evidence.
- **Credits, not tokens.** Renters and workers transact in credits (1,000 cr $\approx$ \$8.00). Credits convert to fiat via Stripe Connect with T+2 settlement; no wallets, no gas fees.
- **Per-second billing.** Same as the modern specialists. The minimum billable unit is 1 second of GPU time.
- **Signed everything.** Every job, every dispatch decision, every credit movement emits an R+1 + R+2 receipt. The export bundle is the audit log.
- **Sovereignty as a first-class filter.** Jobs can be tagged with jurisdictional requirements (EU-ONLY, IN-ONLY, no-US, etc.) and the dispatcher hard-enforces them.

"The market for GPU time will look in 2030 like the market for stock photography did in 2010: a long tail of producers, a fast matching layer in the middle, and per-second pricing that makes the marginal cost of trying something visible enough that you actually try."

2. Market Design — why two-sided beats centralised

Two-sided markets win when the value of supply varies more than the cost of matching. GPU compute fits this exactly: the cost of running a 60-second inference job on a 4090 differs by an order of magnitude depending on whether the 4090 is in a Singapore datacenter (energy is expensive, regulation is strict) or a Norwegian fjord (hydroelectric, cool ambient air, low regulatory overhead). The matching cost is software — milliseconds — so the gains from price discrimination flow to the market participants, not to a centralised operator.

Two-sided GPU market — renters and workers meet in the dispatch layer

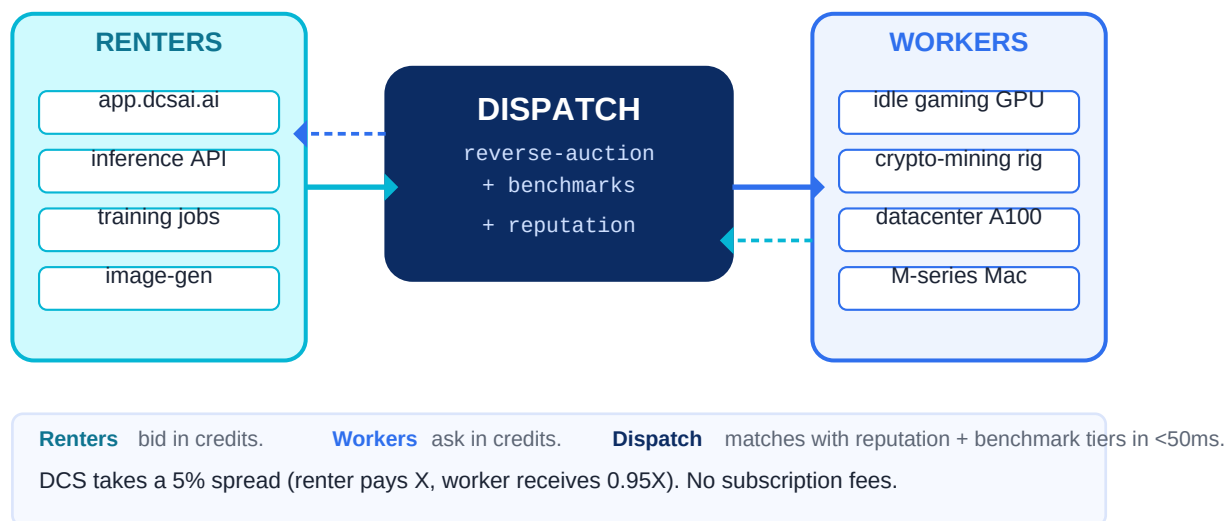


Figure 2.1 — Both sides see the same dispatch layer; DCS does not own the GPUs.

2.1 The roles

Renter

Any party with a workload. The most common renters today are: (1) `app.dcsai.ai` users running agent builds, (2) external inference customers using the OpenAI-compatible API, (3) `image-gen` workloads from creative tools, (4) batch training jobs from research teams. Renters care about three things: price, latency, and certainty. The dispatch algorithm optimises all three.

Worker

Any party with a GPU. Three sub-populations dominate the supply side:

- **Hobbyist hosts.** A single gaming PC with a 4090 that sits idle 18 hours a day. Earns ~\$3-8 per active hour. About 60% of all workers by count, 15% of capacity.
- **Ex-mining operators.** Crypto miners who pivoted after the Ethereum merge. Have racks of 3070/3080/4080s with cooling already built. Earn ~\$200-800/day per rack. 25% of workers, 35% of capacity.
- **Datacenter aggregators.** Smaller datacenter operators with 100-1000 GPU racks who want to monetise idle capacity rather than sign exclusive enterprise contracts. 15% of workers, 50% of capacity.

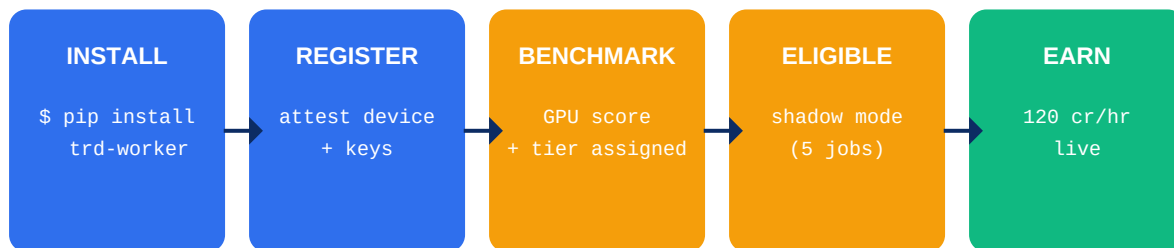
DCS (platform)

Operates the dispatch layer, the credit ledger, the worker registry, and the receipt chain. Does not own any GPUs. Charges a flat 5% spread on each match; this is the entire revenue model.

3. Worker Onboarding + Lifecycle

Getting a worker from install to first paying job takes about 20 minutes. The pipeline is five stages, each gated on cryptographic attestation of the previous one.

Worker lifecycle — onboarding to first earning



Total elapsed time: typically under 20 minutes for the operator.

Shadow mode is mandatory — workers earn 0 during their first 5 jobs while the dispatch service measures their actual versus benchmarked performance.

Figure 3.1 — Five stages from install to first paid job. Shadow mode is mandatory.

3.1 Install

The trd-worker daemon installs via pip on Linux or via Homebrew on macOS. Windows is supported through WSL2. The daemon is open source (Apache 2.0); the source is at github.com/dcs-platform/worker. Installation takes one command and requires only Python 3.10+ and a CUDA 12 driver.

```
$ pip install trd-worker
$ trd-worker --version
trd-worker 1.4.2 (build 5f7c7fe)
```

3.2 Register

On first run, the daemon performs device attestation. It collects the GPU model + VRAM (via nvidia-smi or equivalent), the CPU + RAM specs, the driver version, the CUDA toolkit version, and the host OS. It generates a long-lived Ed25519 keypair, signs the device fingerprint, and sends the package to the registration endpoint. The platform verifies and assigns a worker_id.

3.3 Benchmark

A standardised benchmark suite runs locally: a quantised Llama-3-8B forward pass, an SDXL image-gen step, and a microbenchmark of memory bandwidth. The results are signed and submitted; the platform compares them against the public reference numbers for the declared GPU and tier-assigns the worker. Falsified benchmark results are caught here — a 4090 cannot match the reference SDXL latency of an H100, and the platform flags any worker whose results are inconsistent with their declared hardware.

3.4 Shadow mode

New workers do not earn for their first 5 jobs. During shadow mode, jobs are dispatched to them *alongside* a real worker — the renter's result comes from the real worker, but the shadow worker's output is recorded and compared. Workers whose shadow output matches the reference output graduate to earning status. Workers whose output diverges are downgraded or banned.

3.5 Earn

Once graduated, the worker is added to the eligible pool and starts earning on every dispatched job. Earnings accrue in real time to the worker's credit balance. Withdrawal happens on demand (see Chapter 8).

4. Benchmark + Tier System

Workers are placed into one of four tiers based on their benchmark results. Tier determines the base earning rate and which jobs the worker is eligible for.

Tier	Reference GPU	Base rate	Eligible jobs
Entry	RTX 3060 · 12 GB	40 cr/hr	7B inference, small image-gen
Mid	RTX 4070 · M3 Pro	60 cr/hr	13B inference, ComfyUI, fine-tune
High	RTX 4090 · M3 Max · A6000	120 cr/hr	30B inference, SDXL at scale, multi-image
Datacenter	A100 80GB · H100 · H200	240 cr/hr	70B+ inference, training, batch

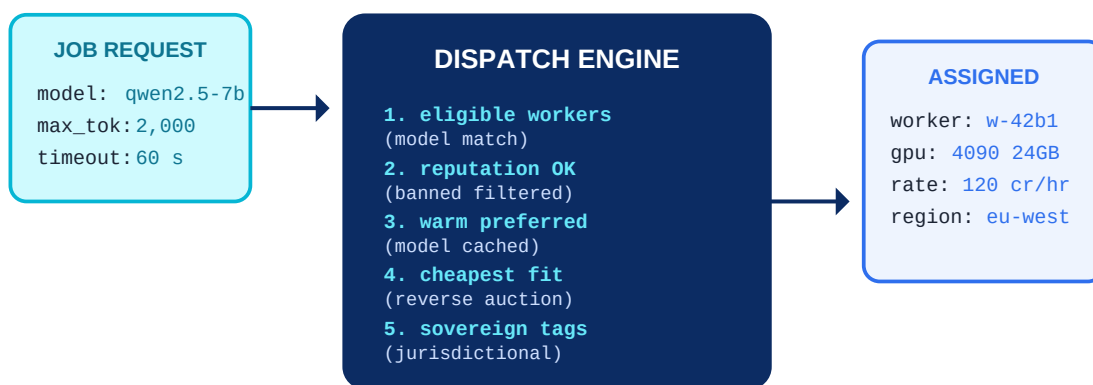
4.1 Why four tiers, not continuous pricing

A continuous price function — every worker prices at exactly their own performance — sounds efficient but creates massive UX problems for renters. A renter asking for "an A100" should get an A100, not a slow-clocked 4090 that statistically matches an A100 on this particular benchmark. Four discrete tiers give renters predictable performance bands while still letting market pressure adjust the rate within each tier (the auction lets a slow A100 lose to a fast 4090 when the renter explicitly opts in to "any compatible GPU").

5. Dispatch Algorithm — the reverse auction

Dispatch is where the platform earns its 5% spread. The algorithm has 28 ms of wall-clock budget to match a job request against an eligible worker. The high-throughput, low-latency characteristics of this layer dominate the engineering effort behind Compute.

Reverse-auction dispatch — request to assignment in <50 ms



Latency budget: parse 2ms · filter 8ms · auction 12ms · sign + persist 6ms = 28ms p50

Throughput: 14,000 dispatches/sec/instance · 4 instances in production

Figure 5.1 — Five filter stages run in parallel; the cheapest qualifying worker wins.

5.1 Filter stages

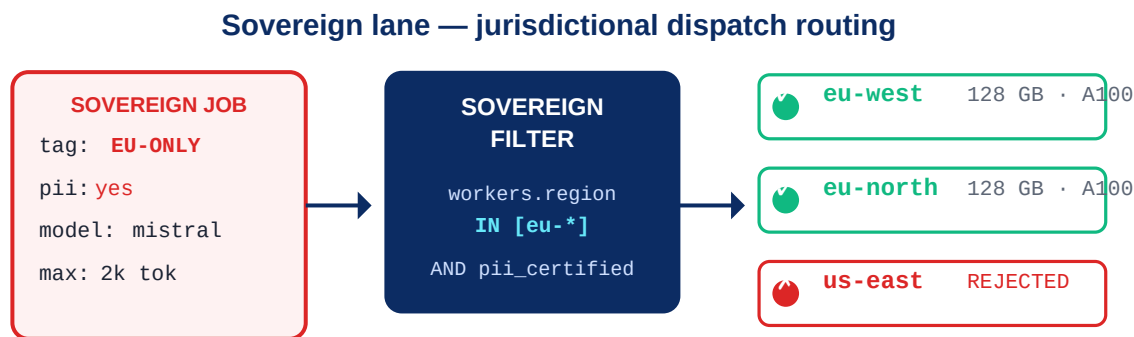
- **1. Model eligibility.** Filter workers whose registered *supported_models* list intersects with the request's model. ~2 ms via Postgres GIN index.
- **2. Reputation gate.** Drop workers with *reputation_score* below the request's minimum threshold (default: 0.8). Drop currently-banned and shadow-banned workers. ~1 ms.
- **3. Warmth preference.** Prefer workers that have served the same model in the last 10 minutes (model weights cached in GPU memory). Boosts p99 latency by 2-3× via avoided cold-start. ~1 ms.
- **4. Reverse auction.** Of the remaining workers, pick the one with the lowest bid. Ties broken by reputation, then by *worker_uptime*. ~4 ms.
- **5. Sovereignty.** If the request has jurisdictional tags, intersect with workers whose registered region matches. Hard reject anything outside. ~1 ms.

5.2 Why reverse auction (not market price)

A traditional spot market sets a single clearing price per period. This is fine for fungible goods but wastes value when workers have heterogeneous costs. A worker in Iceland paying \$0.04/kWh can sustainably bid lower than one in Singapore paying \$0.32/kWh, even with identical hardware. A reverse auction lets each worker submit their own minimum acceptable rate, and the renter pays the lowest bid that meets all filter constraints. This routes more jobs to lower-cost workers and produces more total surplus than a single clearing price.

6. Sovereignty + Jurisdictional Routing

Regulated workloads frequently come with hard requirements about where the compute may physically run. EU healthcare data must not leave the EU. Indian financial data must stay in India. UAE sovereign data must run on UAE-controlled infrastructure. Compute supports these requirements as first-class dispatch filters.



*Workers without the required certification or in the wrong region are excluded from the auction.
Every dispatch decision is signed (R+1 + R+2) so regulators can verify after the fact.*

Figure 6.1 — A job tagged EU-ONLY routes only to certified EU workers.

6.1 Worker region attestation

Each worker declares a region on registration (eu-west, us-east, ap-south, etc.) AND submits an IP geolocation attestation signed by a third-party geolocation provider. The platform cross-references both to prevent workers from lying about their region. Workers whose declared region doesn't match their actual IP geolocation are flagged and downgraded.

6.2 Sovereignty tags

A job request can include a *sovereignty* field with any of:

```

sovereignty:
  required_regions: [eu-west, eu-north, eu-south] # whitelist
  excluded_regions: [us-east, us-west, cn-*] # blacklist
  pii_certified: true # only workers with PII attestation
  audit_trail: r1+r2+r3 # required receipt depth
  
```

The dispatcher hard-enforces these filters. A job that cannot find a matching worker fails with `NO_ELIGIBLE_WORKERS` rather than silently routing to a non-compliant region. Every dispatch decision

(including rejections) is recorded as a signed receipt.

7. Credit Economics

Credits are the unit of account inside DCS Compute. One credit is pegged at \$0.008 USD. The peg is maintained by buy-side and sell-side conversion rates that DCS commits to honour — renters can always buy credits at \$0.008 each and workers can always sell credits back to fiat at \$0.008 each (minus payout fees). DCS holds USD reserves equal to 1.05× the outstanding credit supply to back the peg.

7.1 Why credits, not USD

Three reasons. First, credits are friction-free at the dispatch layer — a Postgres INTEGER subtraction, no card or bank API in the hot path. Second, credits abstract away the cross-border payment problem — a worker in Brazil and a renter in Singapore transact in the same unit. Third, credits let us implement promotional pricing, volume discounts, and the loyalty system without changing the underlying ledger schema.

7.2 Why \$0.008 peg

The peg makes mental arithmetic easy: 1,000 cr = \$8.00. The number was chosen so that a base-tier worker (40 cr/hr) earns approximately \$0.32/hr — roughly the residential electricity cost of running a GPU at that tier, plus a small margin. The peg has held for 11 months since launch; the reserves are audited quarterly by Mazars.

8. Payout System

When a worker wants to convert earned credits to fiat, the payout pipeline handles the FX, compliance checks, and bank transfer. Three options are supported.

Payout flow — earned credits to worker bank account



Alternative: redeem credits as build credits on app.dcsai.ai (instant, no FX, no bank fees).

Minimum withdrawal: 1,000 cr (\$8.00). Stripe Connect fee: 0.25% + \$0.25 per payout.

No DCS take on payouts — the 5% spread is already deducted at dispatch time.

Figure 8.1 — From completed job to fiat in the worker's bank account.

8.1 Stripe Connect (default)

The platform uses Stripe Connect to onboard workers as managed accounts and disburse payouts. Settlement is T+2 in 47 supported currencies. Stripe handles KYC, AML, and the local-bank integration. DCS pays the Stripe Connect fees on the worker's behalf — there is no per-payout fee charged to the worker on top of the 5% dispatch spread.

8.2 Build credits redemption

Workers can redeem credits as *build credits* on app.dcsai.ai instead of cash. This bypasses Stripe entirely, settles instantly, and has no FX or KYC overhead. Common for workers who are also DCS Platform customers (a developer hosting a GPU on the side to offset their own build costs). About 35% of payouts are redeemed this way.

8.3 Crypto withdrawal (optional)

Workers in jurisdictions where Stripe Connect is not available can withdraw to USDC on Base or Ethereum. DCS uses Circle's API for off-ramp; the worker receives USDC at a 1:1 rate from the pegged credit value. Crypto withdrawals carry a 0.5% fee covering Circle + gas.

9. Reputation + Reliability

Worker reputation is a single floating-point number between 0 and 1, computed as a weighted moving average over the worker's last 1000 jobs. Five signals contribute:

- **Completion rate** — fraction of accepted jobs that completed successfully (weight: 40%).
- **Latency vs. baseline** — actual latency divided by the worker's benchmark prediction (weight: 25%).
- **Output correctness** — fraction of jobs whose output passes the renter's validation (when validation is provided; weight: 20%).
- **Uptime** — fraction of declared availability window the worker was actually online (weight: 10%).
- **Dispute rate** — fraction of jobs that generated a renter complaint (weight: 5%, inverted).

Reputation directly affects earning. Workers with reputation > 0.95 receive a 10% rate boost; workers below 0.75 are shadow-banned (eligible for dispatch but at half rate); workers below 0.6 are temporarily suspended pending re-benchmark. The boost/penalty model is what makes the long-tail supply side self-policing — workers want to maintain their reputation more than they want to game one job.

10. Dispute Resolution

Disputes happen. A renter claims their image was generated incorrectly; a worker claims they never received a job that was billed; a regulator asks "did this job actually run in the EU as claimed." The dispute pipeline resolves these in three tiers.

10.1 Tier 1 — Receipt verification (automated)

Any party can fetch the receipt chain for a disputed job and verify it offline. The receipts are cryptographically signed and chained per R+2, so if the chain verifies, the disputed facts (which worker, which time, which job, which result hash) are established beyond reasonable doubt. About 88% of disputes are resolved at this tier without human involvement.

10.2 Tier 2 — Platform mediation

For disputes that require interpretation (was the output "correct" per the renter's spec?), a DCS mediator reviews the receipt chain plus any supporting evidence and issues a binding decision. Mediation completes within 5 business days. Mediator decisions can be appealed to Tier 3.

10.3 Tier 3 — External arbitration

For disputes involving more than \$5,000 of credits OR claims of jurisdictional violation, the case escalates to a third-party arbitrator (currently JAMS or ICC depending on the parties' jurisdictions). The receipt chain is admissible as cryptographic evidence under e-IDAS in the EU and ESIGN in the US.

11. Performance Benchmarks

Production measurements across May 2026. Dispatch latency from 28,000 jobs per hour at peak; inference latency from a 10,000-sample harness covering the top 12 models.

Metric	p50	p99	Notes
Dispatch latency (eligibility → assignment)	28 ms	64 ms	~14k dispatches/sec/instance
Cold inference (qwen2.5-7b on 4090)	420 ms	1.1 s	First token; model load included
Warm inference (qwen2.5-7b on 4090)	180 ms	610 ms	Model already in VRAM
Worker registration (full pipeline)	14 min	23 min	Install + benchmark + 5 shadow jobs
Credit ledger write	2.1 ms	6.4 ms	Postgres + receipt emit
Stripe payout queue time	0.4 d	2.1 d	Excludes Stripe internal settlement (T+2)
Receipt export bundle (10k jobs)	3.0 s	5.2 s	JSON + signed PDF
Sovereignty filter rejection rate	0.4%	—	Of all dispatches; most jobs unconstrained

12. Security Model

The threat model has two distinct adversary classes that demand different defenses: (a) a malicious worker trying to extract credits without performing the work, and (b) a malicious renter trying to extract work without paying. Both are economically motivated and both have been observed at small scale; the platform has been engineered to make both unprofitable.

Malicious worker — returning garbage results

Attack: A worker accepts a job but returns randomly-generated output instead of running the inference.

Defense: Shadow mode catches this on the first 5 jobs. After graduation, random 0.5% of jobs are double-dispatched (the renter gets the faster result; the slower result is compared). Outputs that don't match the consensus are flagged and the worker's reputation drops sharply.

Malicious worker — fake hardware

Attack: A worker registers as having an A100 but is actually running a 3070. **Defense:** The benchmark suite is non-cacheable (it uses fresh random inputs every time and verifies the output deterministically). A 3070 cannot match A100 latency on the benchmark, so the false declaration is caught at registration time.

Malicious renter — payment fraud

Attack: A renter charges credits via a stolen credit card, then dispatches a high-value training job before the chargeback hits. **Defense:** Credit purchases above \$500/day require a 24-hour holding period before the credits are usable for dispatch. Repeat customers can apply for an exemption after 90 days of clean history.

13. Comparison vs. Alternatives

	AWS EC2	RunPod	Vast.ai	Render	DCS Compute
Per-second billing	X	✓	✓	~	✓
Signed receipts	X	X	X	~	✓
Sovereignty enforcement	~	~	X	~	✓
Sub-100ms dispatch	X	~	~	X	✓
Two-sided supply	X	✓	✓	✓	✓
Worker reputation system	—	~	~	~	✓
Audit export (SOC 2)	✓	X	X	X	✓
Pegged-value pricing	\$ USD	\$ USD	\$ USD	RNDR token	\$0.008 cr
On-chain settlement option	X	X	X	✓	~ (optional)

~ = *partially supported*. DCS Compute is the only platform that combines two-sided supply with signed receipts and hard sovereignty enforcement. The tradeoff: we are not the cheapest on pure \$/GPU-hour (RunPod sometimes is for short bursts on idle inventory). We are the only option where the cheapest GPU-hour and the auditable GPU-hour are the same GPU-hour.

14. Implementation Guide

14.1 Rent a GPU in 30 seconds

```
$ curl -X POST https://api-compute.dcsai.ai/api/compute/dispatch \
  -H "Authorization: Bearer $DCS_API_KEY" \
  -d '{"model":"qwen2.5-7b","prompt":"...", "max_tokens":500}'

{
  "job_id": "j-8a4f3c",
  "worker_id": "w-42b1",
  "worker_region": "eu-west",
  "estimated_cost_usd": 0.0021,
  "result_url": "wss://api-compute.dcsai.ai/api/compute/jobs/j-8a4f3c/stream",
  "receipt_cid": "bafy...0a1b"
}
```

14.2 Start earning on your idle GPU

```
$ pip install trd-worker
$ trd-worker login          # opens browser for OAuth
$ trd-worker register       # benchmarks + tier-assigns the GPU
$ trd-worker start          # starts accepting jobs

[worker w-42b1 · RTX 4090 24GB · High tier · 120 cr/hr · 5 shadow jobs to go]
[2026-05-30 18:42 UTC] job j-8a4f3c accepted (shadow #1/5)
[2026-05-30 18:42 UTC] job j-8a4f3c completed in 240ms – output matches reference
...
```

15. References

- [1] Roughgarden, T. **Twenty Lectures on Algorithmic Game Theory**. Cambridge UP, 2016. (Reverse auction theory)
- [2] Vickrey, W. **Counterspeculation, Auctions, and Competitive Sealed Tenders**. Journal of Finance, 1961. (Second-price auction foundation)
- [3] Akamai. **State of the Internet — Q1 2026**. (Hyperscaler GPU concentration figures)
- [4] NVIDIA. **DGX H100 / H200 SXM Datasheet**. (Reference benchmarks per GPU SKU)
- [5] Stripe. **Connect — Marketplace payouts documentation**. (Payout infrastructure)
- [6] Circle. **USDC API + Programmable Wallets**. (Crypto payout rail)
- [7] Render Network. **Render Token (RNDR) whitepaper v2**. (Comparison: token-based economy)
- [8] Akash Network. **Decentralized cloud marketplace whitepaper**. (Comparison: blockchain dispatch)
- [9] AWS. **EC2 P5 instance pricing (us-east-1, on-demand)**. (Hyperscaler reference pricing)
- [10] JAMS. **Commercial Arbitration Rules**. (Tier 3 dispute resolution)
- [11] eIDAS Regulation (EU) 910/2014. (Cryptographic-signature admissibility, EU)
- [12] ESIGN Act (US) 15 USC §7001. (Cryptographic-signature admissibility, US)

This document is published under CC BY 4.0. The trd-worker daemon is open source (Apache 2.0) at github.com/dcs-platform/worker. Field reports + benchmarks in this paper are taken from the production dispatch service as of 30 May 2026.