

Verifiable Home-GPU Compute Sharing with Duplicate-Dispatch Checks and Merkle-Anchored Receipts on a Sharded PoW Chain

Ying Willam Zhang (ORCID 0009-0002-1526-3332), SCDO Laboratory, 9Y9 PTY LTD

Preprint, October 2026

Abstract

Owners of home GPUs need a low-friction way to contribute compute and receive verifiable payment. This paper describes `scdo-gpu-share`, a worker and coordinator system in which each worker is identified only by a `secp256k1` key address and runs deterministic `llama.cpp` and `Blender` jobs. Results are compared by SHA-256 under duplicate dispatch: the first 20 jobs of every worker are always assigned twice, and 10% of later jobs are duplicated, never to a worker on the same public IP. Result batches are anchored as Merkle roots in the `AgentLedger` contract on SCDO Shard0 (chainId 5680), and only completed and anchored results are paid in the native unit SCDO0. A 10/10-minute schedule shares time between the owner and the platform, and idle platform time goes to public-good work (`Folding@home` team 1068523) or SCDO mining. A real-chain end-to-end run on 5 October 2026 produced four transactions in blocks 61707 to 61710 (registration, anchoring and two 0.01 SCDO0 payouts), all with receipt status 1, using 341,609 gas in total over a 38 s on-chain span. The end-to-end test also asserts that a second settlement pass sends no further payout.

Keywords: volunteer computing, home GPU, duplicate dispatch, Merkle anchoring, verifiable receipts, idempotent settlement, `Folding@home`, sharded proof-of-work

1. Introduction

Many individual GPU owners have hardware that sits idle for part of the day. Volunteer computing platforms such as BOINC [1] and `Folding@home` [2] showed that this capacity can support large public projects. `scdo-gpu-share` adds a direct and verifiable payment path for the individual owner, with a low-friction way to join.

Contributions:

1. A low-friction worker whose identity is a key address, persistent from worker release 1.2.4, with signed registration, benchmark and heartbeat messages.
2. A duplicate-dispatch policy for deterministic jobs (always for each worker’s first 20 jobs, then 10%), with an analytical detection bound.
3. Merkle-anchored receipts in the `AgentLedger` contract [5], [6] and a settlement rule that pays only completed and anchored results and is idempotent.
4. An owner-friendly time-sharing schedule with public-good use of idle platform time, and a real-chain end-to-end validation on SCDO Shard0.

2. Background and Related Work

Volunteer computing. BOINC [1] established public-resource computing on heterogeneous home machines, and `Folding@home` [2] brought volunteer computing to biomolecular simulation. `scdo-gpu-share` connects to

Folding@home through the official client and team 1068523, “SCDO Laboratory” [7].

Verifiable computation. TrueBit [4] verifies off-chain computation through an interactive challenge protocol. scdo-gpu-share follows a lighter path suited to deterministic jobs: redundant execution with hash comparison, plus non-interactive anchoring of results.

Merkle trees and AgentLedger. Merkle trees [3] let a single root commit to many leaves, with short inclusion proofs. AgentLedger [5], [6] is the laboratory’s receipt contract on SCDO Shard0, with functions `registerAgent(name, metadataURI)`, `anchorRoot(root, count, batchURI)` and `verifyProof`.

3. System Design

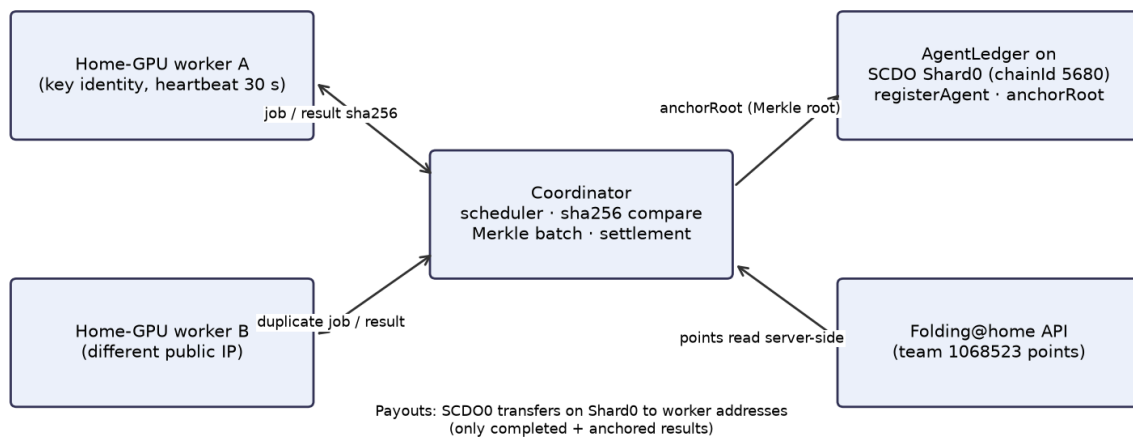


Fig. 1. Architecture of scdo-gpu-share: two workers on different public IPs, the coordinator, AgentLedger on SCDO Shard0, and the Folding@home statistics API.

3.1 Actors

Workers run on home GPUs. A worker has no account. It signs registration, benchmark and heartbeat messages with secp256k1 `personal_sign`, and addresses and amounts are Unicode NFKC-normalized. The *coordinator* schedules jobs, compares results, builds Merkle batches, anchors them, and sends payouts from its treasury address. *AgentLedger* on SCDO Shard0 holds agent registrations and anchored roots.

3.2 Deterministic job model

There are two job types: `llama_cpp` {prompt, seed, n_predict, optional model_path, minVramMB} and `blender_render` {blend_file, frame, optional minVramMB}. llama-cli runs with `--temp 0` and the job’s seed, and only stdout is kept. Blender runs headless, and the result is the SHA-256 of the rendered PNG. Because the output depends only on the inputs, two honest workers produce identical hashes.

3.3 Discovery, liveness and scheduling

The coordinator reads AgentRegistered logs about every 30 s. It assigns jobs when they are enqueued and again every 10 s. Workers send a heartbeat every 30 s, and a worker counts as available while its last heartbeat is less than 90 s old. Candidates are ranked by VRAM fit, reputation (completed jobs against failed or timed-out

jobs) and current load, and workers with fewer past assignments rank higher. A task whose worker's heartbeat goes stale, or that exceeds `task_ttl_seconds` (3600 in the example configuration), returns to the queue and is not offered to the same worker again. Each public IP may have up to 20 active workers, and workers inactive for more than 7 days are not counted.

3.4 Owner time-sharing

The worker alternates 10-minute platform slots with 10-minute owner slots, so the owner keeps the GPU half of the time. It polls for paid jobs only during platform slots.

4. Verification and Receipts

4.1 Duplicate dispatch

The first 20 original jobs of every worker are always assigned to a second worker. After that, `dual_dispatch_rate` is 0.1. The duplicate never goes to a worker on the same public IP (`dual_dispatch_distinct_ip: true`). The two results are compared by SHA-256.

4.2 Detection probability

Suppose a worker returns wrong results on k jobs and each job is independently duplicated with probability p . The probability that at least one of these jobs is checked is

$$P_{\text{detect}}(k) = 1 - (1 - p)^k.$$

During a worker's first 20 jobs $p = 1$, so any wrong result is compared. After that, with $p = 0.1$, the probability reaches about 0.65 at $k = 10$ and 0.88 at $k = 20$ (Fig. 2). This bound is analytical and is not a measurement.

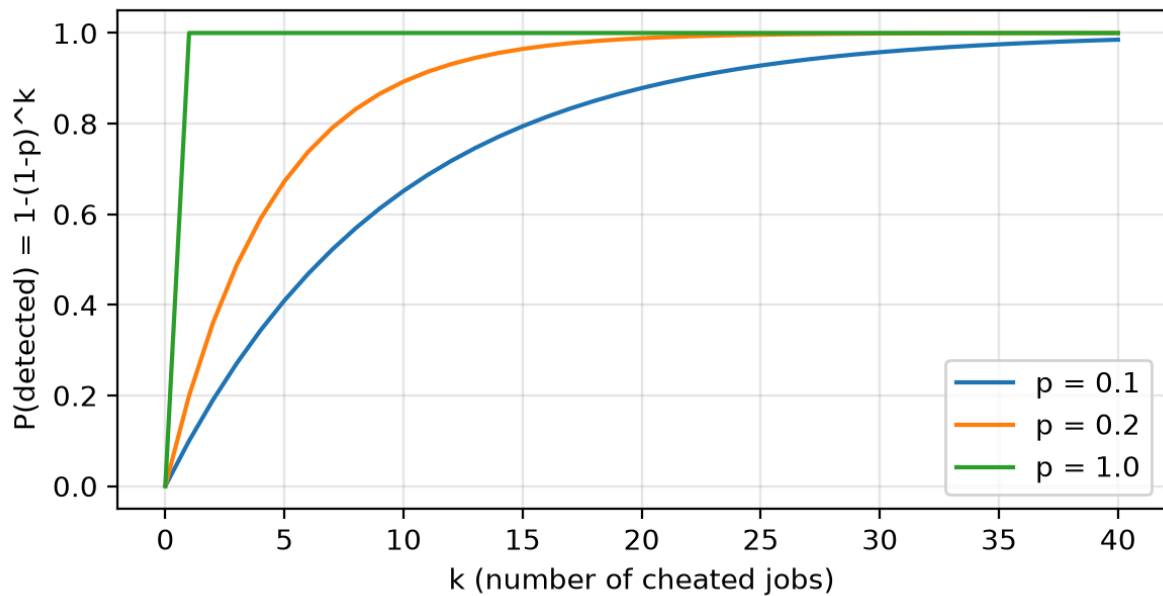


Fig. 2. Analytical detection probability $1 - (1 - p)^k$ for $p = 0.1, 0.2$ and 1.0 .

4.3 Merkle anchoring

The coordinator collects compared results into a batch and anchors the Merkle root with `anchorRoot(root, count, batchURI)`. Anyone can check a result against an anchored root through the contract's `verifyProof` function.

5. Settlement and Idempotency

Settlement pays only results that are both completed and anchored. The reward is a flat amount per task (default 0.2 SCDO), and the code has no size-dependent pricing. Payouts are made once the default minimum of 1.0 SCDO is reached. Each payout row stores the worker's completed, failed and timed-out counts. Once a transaction has a hash it is recorded and never re-sent, even if waiting for its receipt times out. It is reconciled later from its receipt (`internal/payout/reconcile_test.go`). The settlement step is idempotent: running `SettleOnce` a second time sends no new transfer.

Table 1. Protocol parameters (source: repository README and `config.yaml.example`).

Parameter	Value
Duplicate dispatch	first 20 jobs per worker always; then 10% (<code>dual_dispatch_rate: 0.1</code>), distinct IP
Sybil limit	20 active workers per public IP (inactive > 7 days not counted)
Heartbeat / offline	every 30 s / unavailable after 90 s (<code>heartbeat_ttl_seconds</code>)
Time-sharing	alternating 10-min platform and 10-min owner slots
Task TTL (example config)	3600 s
Reward per task	0.2 SCDO (<code>reward_per_task_scdo</code>)
Minimum payout	1.0 SCDO
Folding@home caps	5 SCDO per worker per UTC day; 50 SCDO global daily budget
Gas limits	300k <code>registerAgent</code> / 250k <code>anchorRoot</code> (estimates 168,871 / 134,338)

6. Implementation and End-to-End Validation

The coordinator and worker are written in Go. External binaries are pinned by SHA-256: `fah-client` 8.5.6 and `SCDOLAB/scdo-gpu-miner` v1.0.3. `AgentLedger` is deployed at `0x377aee324323237c11469c0673b39ab30753a2aa` (deployment block 41101). Its source is verified on `scdoscan.io`, and its Foundry suite passes 20 of 20 tests.

End-to-end run. On 5 October 2026 the acceptance test (`server/e2e/e2e_test.go`) ran against the real SCDO Shard0 chain with no mocks. The code was at commit `2c945a6` (PR #1). In the test, two workers `register` and `benchmark`, and one `llama_cpp` job is assigned to both. The outputs match, and `AnchorOnce` anchors the batch. The proof verifies, and `SettleOnce` pays both workers. All four transactions were sent by the treasury address `0x9036d91ed51be47bdb85999d5d96107b66a2ee0b` at a gas price of 10 gwei.

Table 2. On-chain transactions of the end-to-end run (SCDO Shard0, chainId 5680; times UTC+8 from block timestamps).

#	Operation	Transaction hash	Block	Time	Status	Gas used	Value
1	registerAgent	0x8bdf6badb4a89129671ef3c69b95dea50a7f66c17ee811af3c0464d93c954d8e	61707	15:47:02	1	166,383	0
2	anchorRoot	0x73808b87e80df3f4ca296615141597ce3f81db5592eb19ec38b093b014191f0c	61708	15:47:21	1	133,226	0
3	payout A	0xd0989cd136f148103f02b7f71df41004d49a3be7fd2c8c02f4035277eb5d4167	61709	15:47:24	1	21,000	0.01 SCDO0
4	payout B	0x841534aa41c7411520f2e2e4c923fda50c2ba2eb15a944aa808479606f3de0a3	61710	15:47:40	1	21,000	0.01 SCDO0

Payout A went to 0x306f8344321bc2b7d63e96a935223a4c631442e0 and payout B to 0x3ee3b578c2035a0248fd0b5be828b5dc5a396a1f.

What the chain proves. The run produced four transactions in four consecutive blocks. The first and last are 38 s apart (15:47:02 to 15:47:40 UTC+8). All four receipts have status 1, and each AgentLedger transaction emitted one log. Total gas was 341,609, which at 10 gwei is 0.00341609 SCDO0 in fees. Each of the two distinct worker addresses received 0.01 SCDO0.

What the test asserts. After the payouts, the test calls `SettleOnce` a second time and asserts that the number of `payout_records` rows does not change (`server/e2e/e2e_test.go`, L197–207). This is a test assertion and not a separately logged measurement.

7. Idle-Time Public-Good Integration

When there is no paid job in a platform slot, the worker runs public-good providers in the default order `scdo-mining`, then `fah` (modes: `priority` or `rotation`, rotating every 60 min). A paid task pauses the `Folding@home` client through its local API. `Folding@home` runs as the official client, with the worker’s SCDO address as donor and team 1068523 by default. The server reads points from `api.foldingathome.org` and does not rely on totals reported by the worker. The first score is taken as a baseline. Payouts are made in SCDO0 per UTC day, capped at 5 SCDO per worker within a 50 SCDO global budget, and are anchored with leaves `fah:<address>:<day>:<points>:<txhash>`. Mining uses `scdo-gpu-miner` on a public PPLNS pool, and the pool pays mining rewards, not the treasury.

8. Design and Future Work

The following extensions build on the protocol above:

- **Third-worker arbitration and strike rule.** When two hashes differ, a third worker can settle the job by majority, and a strike rule can manage worker eligibility. Both are part of the design and are the next scheduler features.
- **Folding@home rate.** The conversion `fah.scdo_per_million_points` defaults to “0” and is set when production `Folding@home` rewards begin.

- **Live deployment metrics.** The public endpoints `/api/gpu/workers` and `/api/gpu/fah/stats` are ready to report workers online, tasks done and team points as owners join.
- **Cross-GPU reproducibility.** Bit-exact agreement of deterministic outputs across GPU models will be measured and reported.
- **BOINC.** BOINC is listed as a provider and will be enabled as an additional public-good option.

9. Conclusion

scdo-gpu-share gives individual home-GPU owners a low-friction way to contribute compute. Results are checked by duplicate dispatch, anchored as Merkle roots in AgentLedger, and paid in SCDO0 with idempotent settlement. Owners keep their GPU half of the time. A real-chain run on SCDO Shard0 completed registration, anchoring and two payouts, all with status 1, in a 38 s on-chain span.

References

- [1] D. P. Anderson, “BOINC: A system for public-resource computing and storage,” in *Proc. 5th IEEE/ACM Int. Workshop on Grid Computing (GRID)*, 2004. doi:10.1109/GRID.2004.14
- [2] M. Shirts and V. S. Pande, “Screen savers of the world unite!,” *Science*, vol. 290, no. 5498, pp. 1903–1904, 2000. doi:10.1126/science.290.5498.1903
- [3] R. C. Merkle, “A digital signature based on a conventional encryption function,” in *Proc. CRYPTO ’87*, LNCS 293, 1988. doi:10.1007/3-540-48184-2_32
- [4] J. Teutsch and C. Reitwießner, “A scalable verification solution for blockchains,” arXiv:1908.04756, 2019.
- [5] Y. W. Zhang, “SCDO Agent Ledger: On-Chain Identity, Work Attestation and Settlement for Autonomous AI Agents on a Sharded PoW Blockchain,” preprint, Zenodo, 2026. doi:10.5281/zenodo.23119809.
- [6] SCDOLAB, AgentLedger contract 0x377aee324323237c11469c0673b39ab30753a2aa on SCDO Shard0. <https://scdoscan.io/#/address?address=0x377aee324323237c11469c0673b39ab30753a2aa> ; source <https://github.com/SCDOLAB/agent-ledger>
- [7] Folding@home, team 1068523 “SCDO Laboratory.” <https://stats.foldingathome.org/team/1068523>