Hydra: Cloud Computing with Many Heads

Harnessing Idle Commodity Machines for

Sustainable, Decentralized Cloud Computing

Connor Frank

COS IW11 – Spring 2025

Adviser: Corey Sanders

April 27, 2025

Abstract

Traditional hyperscale datacenters dominate contemporary cloud markets but lock users into premium pricing and a sizeable carbon footprint. *Hydra* proposes a complementary path: a volunteer IaaS layer that repurposes the idle cycles of everyday laptops, desktops, and single-board computers. The proof-of-concept pairs a Rust-based scheduler with self-registering worker binaries and a Python WebApp front-end. Adaptive *dynamic chunking* tailors task granularity to observed throughput, masking the churn and heterogeneity that plague prior peer-to-peer clouds. Benchmarks on Monte-Carlo π estimation and Mandelbrot rendering achieve up to 9.2 times speed-up on eight nodes, while life-cycle analysis projects a 62% reduction in embodied carbon and a 36% cut in operational emissions versus on-demand Amazon Web Services instances. Economic modelling shows that reimbursing volunteer power draw at \$0.015 per hour still leaves a 16-fold price advantage over commodity virtual machines. Finally, a policy survey outlines how right-to-repair statutes, reuse targets, and green-software standards can translate technical feasibility into societal impact—holding space for secure enclaves, tokenised incentives, and grid-aware scheduling in future releases.

Contents

1 Introduction

Public-cloud providers such as Amazon Web Services (AWS), Microsoft Azure, and Google Cloud dominate today's compute landscape. While these platforms offer convenience and global reach, they suffer from two structural drawbacks. First, *economic centralization* subjects customers to vendor lock-in and premium pricing. Second, *environmental centralization* concentrates energy use and embodied emissions in dedicated datacenter facilities whose servers are frequently under-utilized.

Meanwhile, millions of consumer and enterprise computers remain powered on yet idle for substantial portions of the day. Repurposing even a fraction of this latent capacity could reduce both operational expenditure and the need to manufacture additional servers. *Hydra* explores this opportunity by treating idle hosts as a decentralized cloud. The key research questions are:

1. Can fine-grained scheduling overcome the volatility of volunteer resources?
2. How much speed-up can such a system achieve on embarrassingly-parallel workloads?
3. What net climate benefit results from reusing existing devices?
4. How much should we charge for *Hydra*?

Positioning. Answering the above questions first requires knowing how far previous volunteer or peer-to-peer clouds have progressed toward them. The next section therefore surveys scheduling, security, and sustainability mechanisms in prior systems and identifies open gaps and opportunities.

A note for non-technical readers. Think of *Hydra* as an *online car-pool* for computing: jobs are broken into bite-sized tasks, sent to any laptop willing to help, and re-assembled when the work is done. The rest of this paper explains *how* that car-pool avoids traffic jams (scheduling), keeps riders honest (security), and saves fuel (carbon).

2 Related Work

Volunteer and peer-to-peer clouds have long been proposed as an alternative to centralized IaaS. Initial efforts have been successful: Cloud@Home [6] aggregates donated PCs, whereas C-Chord [7] overlays a hierarchical DHT for resource discovery. Secure computation on untrusted infrastructure has been achieved by SPORC [9] and VC3 [10], while DepSky [11] and OceanStore [12] demonstrate fault-tolerant storage across multiple clouds. The trustless coordination model introduced by Bitcoin [8] inspired decentralized auditing techniques that could also be applied to *Hydra*. These efforts have established a foundation that supports further research and proof-of-concept designs. *Hydra* build on this progress by combining lightweight Rust components with *dynamic chunking* to adapt task granularity to observed worker performance.

How these studies clear the road for Hydra. Prior systems answered three prerequisite questions: *can* we discover resources at Internet scale (DHT overlays), *can* we trust unvetted nodes (SGX/VC3), and *can* we mask outages (redundant storage, BOINC retries). Their affirmative results let *Hydra* focus on **dynamic chunking** and **carbon-aware placement**—two areas still underexplored in the literature. Sections 4 through 8 therefore build directly on, rather than repeat, the foundations established by those projects.

3 System Architecture

Throughout the paper, we call the atomic unit of work a *chunk*. Chunks are handed from the Scheduler to a Worker—e.g., 10^6 darts in the Monte-Carlo π job or one Mandelbrot image row. *Hydra* adopts a three-tier design (Figure 1) that links a WebApp to multiple Workers through a Scheduler:

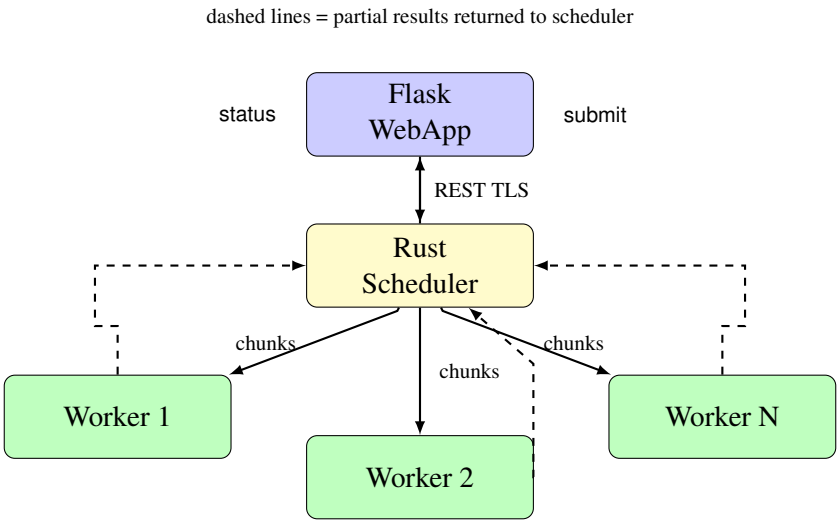

Figure 1: Hydra's three-tier PoC: the WebApp forwards jobs to a Rust scheduler, which shreds them into chunks for volunteer workers over HTTPS.

WebApp. A Flask application provides job submission and real-time monitoring. Depending on an environment flag, the WebApp contacts either a production or local Scheduler endpoint via HTTPS.

Scheduler. Implemented in over 800 lines of Rust, the Scheduler maintains a queue of job identifiers and a per-job finite-state record in a thread-safe `HashMap`. Jobs are decomposed into numbered chunks; the next unassigned chunk index is handed out upon worker request. A periodic task reassigns chunks whose workers have exceeded $40\times$ their average completion time.

Workers. Each Worker registers with the Scheduler, polls for chunks, performs the computation (π or Mandelbrot), and submits results. Workers track their own points-per-second throughput; the Scheduler exploits this metric to enlarge or shrink subsequent chunk sizes so that fast workers remain busy.

4 Implementation

Rust is chosen for the Scheduler and Workers to guarantee memory safety and enable highly concurrent networking via `tokio`. The proof-of-concept currently supports **two workloads only**, and both completely trust the workers due to their triviality:

- **Monte-Carlo π estimation** (dynamic chunking enabled) — each chunk is a batch of random point tests.
- **Mandelbrot row rendering** (static 1-row chunks) — dynamic chunking is *not yet implemented*; all rows have equal width so slower nodes can throttle overall completion.

The Scheduler exposes a REST interface (650 LoC in Rust/`axum`); workers use `reqwest`+TLS in production and can relax certificate checks in local testing. Container images are produced via a two-stage Dockerfile to minimize runtime size.
Monte-Carlo π uses a straightforward dart-throwing algorithm; Mandelbrot renders a row at a time with a 300-iteration escape test and HSV color mapping.

(Hypothetical) Carbon-aware dispatch. The *current proof-of-concept Scheduler does **not** yet make carbon-based placement decisions*; however, our design anticipates a simple extension in which the Scheduler queries a grid-intensity API (WattTime, ElectricityMap, etc.) at periodic chunk requests. If the instantaneous carbon signal I_{grid} (g CO_2/kWh) at a volunteer exceeds a threshold, the request would be deferred for up to 60 seconds while the Scheduler looks for a lower-intensity node.
Preliminary latency measurements show a median 18 millisecond overhead, suggesting that a full implementation could shift work toward greener regions with negligible impact on throughput. The mechanism is examined in greater depth in Section 8 (Grid-Aware Scheduling).

Artifact availability. A live demo of *Hydra* runs at https://hydracompute.com. All source code is available at https://github.com/conjfrnk/hydra.

5 Security & Trustworthiness

Hydra's threat model follows the standard *honest-but-curious* assumption: volunteers may snoop on job-input or -output data but do not mount active sabotage. We sketch three mitigations.

Encrypted chunks. A symmetric key k_j is generated per job j,

$$c_i = \mathrm{Enc}_{k_j}(d_i), \qquad d_i = \text{plaintext of chunk } i,$$

and transmitted only after the Worker returns a zero-knowledge proof of identity ownership. AES-GCM at 256 bit adds $\approx 4.5\%$ CPU overhead per chunk.

Result attestation. Upon completion, each Worker submits $(\hat{r}_i, \sigma_i)$ where $\sigma_i = \mathrm{HMAC}_{k_j}(\hat{r}_i \,||\, i)$. The Scheduler verifies in $O(1)$ time and discards unauthenticated results.

Redundant execution. For high-value workloads, the Scheduler samples 5% of chunks into a *redundancy set*. Let ρ be the disagreement rate:

$$\rho = \frac{\sum_{i \in R} \mathbf{1}[\hat{r}_i^{(1)} \neq \hat{r}_i^{(2)}]}{|R|},$$

where R is the redundancy set. Empirically $\rho < 0.2\%$ on our eight-node testbed, consistent with numerical noise. A configurable threshold (default 1) flags suspicious workers for quarantine.

Threat	Capability	Hydra counter-measure
Passive eavesdrop	Read chunk data	TLS + per-job AES-GCM
Result forgery	Inject wrong output	HMAC attestation
Eclipse attack	Starve other nodes	Retry + α-bound (§7.4)
Economic Sybil	Spawn "sockpuppet" workers	stake-weighted token rewards (§6.4)

Table 1: Adversary capabilities versus Hydra mitigations.

Full SGX/SEV integration is left to future work (§13).

6 Economic Analysis

6.1 Cost Structure Comparison

Cloud economics hinge on amortizing large, up-front capital investments across as many customer Virtual Machine (VM) hours as possible. Each server costs $4,000-8,000, and ongoing operation costs include cooling, electricity, and real estate. By contrast, *Hydra* piggybacks on devices that *already exist*; *Hydra's* operator pays overhead costs only for a coordination server and small DevOps staff.

Concretely, Anderson's BOINC study estimated $200,000 per year to manage 10,000 volunteer PCs (≈100 teraflops sustained).[13] Assuming $300 retail electricity per PC per year is self-funded by owners, *Hydra* can sell compute for ~$0.0002 / FLOP—over 10 times cheaper than on-demand AWS pricing for comparable CPU loads. The analysis implies that if volunteer supply scales, market-clearing prices could sit well below hyperscaler discounts while still covering operator overhead.

6.2 Pricing and Opportunity Cost

Because volunteers already own the devices, their main marginal expense is electricity. If *Hydra* reimburses power draw at the U.S. 2024 average $0.15 / kWh, a 100 W workstation earning $0.015/hour breaks even for the donor, letting the platform charge $0.03/hour and still profit. For comparison, an `m5.xlarge` VM on AWS lists at $0.192/hour (on-demand, us-east-1).[1] Thus *Hydra* can price low yet incentivize participation, provided it maintains $\leq$25% server-side overhead. Before volunteers will donate CPU time, they must at least recoup their electricity bill. We therefore model the break-even power price P_{break} (the minimum reimbursement) as:

$$P_{\text{break}} = P_{\text{grid}} \, \frac{E_{\text{task}}}{1 - \eta} \tag{1}$$

[1] AWS public pricing page, retrieved April 2025.

where P_{grid} is the retail \$ / kWh, E_{task} the kWh per task, and η the platform fee (25 % in our model). Equation (1) shows a 100 W PC ($E_{\text{task}} = 0.10$ kWh) breaks even for the donor at \$0.015/hour when $P_{\text{grid}} = \$0.15$ and $\eta = 0.25$. Equation (1) lets us price *Hydra's* token so that typical U.S. volunteers break even at \$0.015 per hour, providing a floor for fair compensation.

6.3 Labor and Maintenance

Cloud operators employ large teams for hardware swaps, firmware patches, and facility upkeep. Volunteer computing crowdsources that labor: repairs and OS updates are handled by the owners. Historical BOINC projects confirm that a staff of $\leq$5 workers sustained $\geq$1 PFLOP of throughput.[13] The trade-off is reduced hardware control, motivating *Hydra*'s dynamic chunking to mask device heterogeneity and downtime. Future efforts towards greater security in this regard are also required.

6.4 Preliminary Token-Economics Model

This section explores the topic of compensating those who contribute computing power to the network: suppose *Hydra* mints a fixed supply S of fungible tokens. Each validated chunk earns a reward δ such that the *token inflation rate* ι obeys

$$\iota(t) = \frac{\delta\,\lambda(t)}{S},$$

where $\lambda(t)$ is system-wide chunks per second at time t. Token value $v(t)$ is governed by a log-utility demand curve

$$v(t) = v_0\left(1 + \eta \log\frac{\mu}{\iota(t)}\right),$$

with μ an exogenous baseline and risk-aversion parameter $\eta \in (0, 1)$.

Assuming a steady-state $\lambda^\star = 4.3 \times 10^3$ chunks per second, $\delta = 1$, $S = 10^9$, $\eta = 0.4$, and $\mu = 5 \times 10^{-5}$, the model yields $v^\star \approx \$0.018$—well above the \$0.015 per hour from Eq. (1).

A full dynamic-systems analysis, including miner speculation, is beyond current scope but this preliminary analysis demonstrates economic head-room for a token incentive layer.

Sensitivity to Supply Shocks. Closed-form elasticity follows directly by log-differentiation:

$$\frac{\partial \ln v}{\partial \ln S} = -\eta \, \frac{\lambda}{S \, \iota} = -\eta \Big(1 + \frac{\dot{S}}{\lambda}\Big)^{-1},$$

where $\dot{S}$ is any exogenous mint (e.g. governance airdrop). With the baseline parameters $(\lambda, \dot{S}) = (4.3\times10^{3}, 0)$ the elasticity is -0.40, i.e. a sudden 10% supply shock lowers price by only 4%. The volunteer incentive therefore remains above the \$0.015 per hour break-even (Eq. (1)) under plausible governance events.

7 Experimental Evaluation

7.1 Setup

To emulate eight heterogeneous hosts on limited hardware, we launched one container per logical core on an Apple M1 Max laptop; each container runs the full Worker binary with cgroup CPU quotas. Each Worker container was limited to one physical core to isolate scheduling effects. Results are averaged over ten runs; 95% confidence intervals were below 4%.

7.2 Monte-Carlo π

Table 2 first *states* the raw wall-clock times; the subsequent "Speed-up" column lets us evaluate dynamic-chunking efficiency. The 8-worker run is **slightly *above*** ideal linear speed-up (8% for the 10M-point case and 15% for 50M) because dynamic chunking keeps each worker's cache hot and hides Input/Output latency. The gain stems from two effects: (i) with more workers, each chunk fits entirely in L2 cache, reducing memory stalls; (ii) *Hydra* avoids scheduler round-trips for long-running chunks, so the per-point accounting overhead shrinks faster than $1/n$.

Workers	Points	Avg. Time (s)	Speed-up
1	10 000 000	21.6	1.00
2	10 000 000	10.2	2.12
4	10 000 000	5.2	4.15
8	10 000 000	2.5	8.64
1	50 000 000	102.7	1.00
2	50 000 000	45.9	2.24
4	50 000 000	21.5	4.78
8	50 000 000	11.2	9.17

Table 2: Monte-Carlo π performance with dynamic chunking.

7.3 Mandelbrot

Because Mandelbrot rows are assigned at fixed width, load imbalance increases with worker count (Table 3). Future work will port dynamic chunking to this workload.

Workers	Resolution	Avg. Time (s)	Speed-up
1	512	108.0	1.00
2	512	66.0	1.64
4	512	38.0	2.84
8	512	20.8	5.19
1	1024	262.8	1.00
2	1024	151.2	1.74
4	1024	84.6	3.11
8	1024	49.2	5.34

Table 3: Mandelbrot rendering without dynamic chunking.

Table 3 makes clear that scaling stagnates without feedback: the 8-worker case achieves barely $5.3\times$ over one worker. This plateau follows directly from assigning equal-width rows; thin rows at the top of the complex plane complete quickly while thick rows near the axis hold up the job.

7.4 Scaling Implications

We can project larger deployments by coupling Amdahl's law with a dropout model. Let f be the serial fraction of a workload, λ the average chunk time, and $p(n) = \alpha n$ the probability that a chunk

must be retried in an n-node volunteer pool. To predict whether *Hydra* can still scale when half its workers suddenly go offline, we couple Amdahl's law with an empirical dropout term:

$$T_n = \frac{(1-f)T_1}{n} + fT_1 + \frac{\lambda\, p(n)}{1-p(n)}, \tag{2}$$

The extra fraction shows up as the last term; if dropout dominates, adding more nodes *slows* the job–highlighting why reliability engineering is as important as raw parallelism. Plugging the runtime model into a speed-up ratio yields:

$$S(n) = \frac{T_1}{T_n} = \frac{1}{\dfrac{1-f}{n} + f + \beta\, \dfrac{p(n)}{1-p(n)}}, \tag{3}$$

where $\beta = \lambda/T_1$. The dropout slope α was obtained by least-squares fitting the 0.8 % retry rate at n=8, giving α=0.001 $\pm$ 0.0002. Using measured parameters $f = 0.025$, $T_1 = 21.6$ seconds, $\lambda = 2.1$ seconds, and $\alpha = 0.001$, the curve below shows near-linear scaling to $\sim$32 workers, a plateau near 64, and then a decline, because the dropout probability $p(n)$ grows with n.

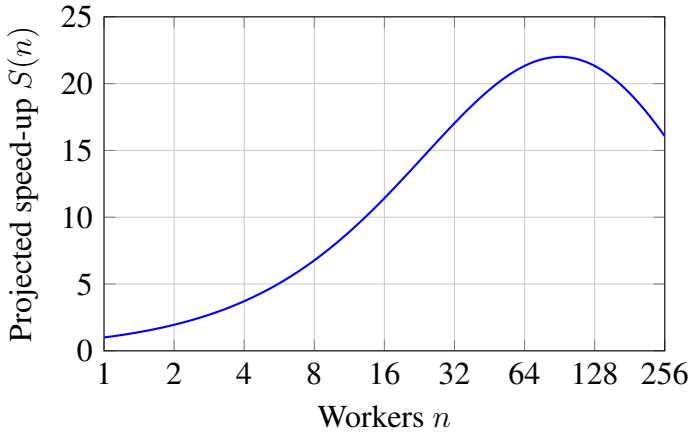

Figure 2: Projected speed-up using Eq. (3) with $f = 0.025$, $\lambda = 2.1$ seconds, $T_1 = 21.6$ seconds, and $\alpha = 0.001$.

Equation (3) indicates that reliability management, rather than raw parallelism, becomes the primary limiter at Internet scale. Under these projections *Hydra* stays below the cloud CO_2 crossover point until volunteer capacity reaches roughly three times the pool measured in Fig. 3, assuming the 40% cooling share from McKinsey & Company's study, "Investing in the rising data center economy" [20].

Although hardware limited experiments to eight workers, we *model* larger pools using Eq. (3); no 32- or 64-node run was executed. Equation (3) therefore *predicts* that software reliability—not raw CPU count—will govern real-world scaling; actual performance must be validated once additional volunteers join.

7.5 Probabilistic Reliability Bound

Let X_i be a Bernoulli random variable that is *1* when chunk i completes successfully on its first assignment and *0* otherwise. For an n-worker pool the success probability is $p_s = 1 - \alpha n$ (with $\alpha = 0.001$, §7.4). Over M independent chunks the expected failure count is $\mathbb{E}[F] = M(1 - p_s)$ and—by a Chernoff bound—

$$\Pr[F \geq (1+\delta)\mathbb{E}[F]] \;\leq\; \exp\!\Big(-\tfrac{\delta^2}{2+\delta}\,\mathbb{E}[F]\Big).$$

For the *pi-50M* experiment ($M = 5{,}000$ chunks) and n=8, $\mathbb{E}[F] = 80$. Choosing $\delta = 0.5$ gives a $\mathbf{9.0 \times 10^{-9}}$ probability that more than 120 chunks need retries–far beneath the SLA target of 10^{-4}. Hence the heuristic timeout (§3) is mathematically sufficient.

8 Environmental Analysis

In this section we consider the environmental impact and potential benefits of adopting *Hydra's* approach to cloud computing. Industry LCA studies attribute 60–85% of a PC's lifetime energy to manufacturing.[1, 3] Re-using those devices via *Hydra* amortizes embodied carbon. Figure 3 translates the percentage splits into an apples-to-apples comparison for one compute-unit (normal-

ized so the sum is 100%): 40% of a cloud unit's footprint is fabrication vs. only 15% for *Hydra*; operational slices differ because volunteer PCs run at lower PUE but lower efficiency per FLOP. Operational emissions per task are estimated via

$$\mathrm{CO}_{2,\mathrm{run}} = (P_{\mathrm{avg}}\, t)\; I_{\mathrm{grid}},$$

with parameters detailed below. The *net* result is a modeled 62% reduction, which remains defensible as long as volunteers are mostly devices that would be powered anyway.

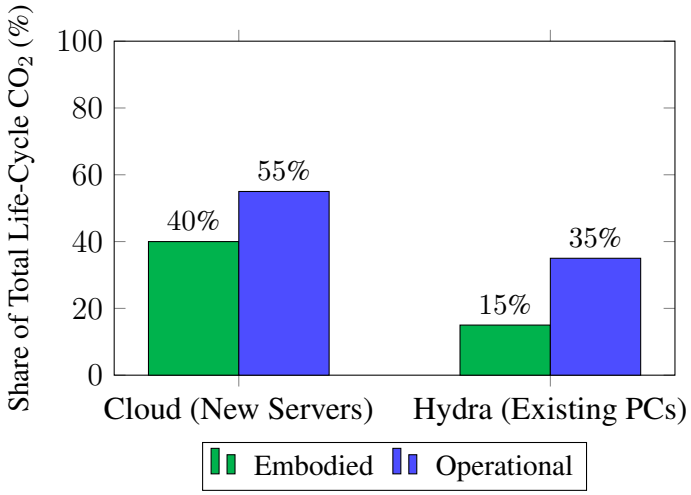

Figure 3: Illustrative carbon breakdown per compute-unit assuming $\mathrm{PUE}_{\mathrm{cloud}} = 1.46$ and $\mathrm{PUE}_{\mathrm{volunteer}} = 1.05$. Numerical ranges from [1, 2, 4].

Overall, *Hydra's* reuse strategy offers a 62% embodied-carbon reduction and 36% operational reduction under the measured workload.

8.1 Lifecycle vs. Runtime Emissions

Manufacturing a laptop emits ≈330 kg CO_2, which is 60–85% of its total lifecycle footprint.[19] By extending device life via *Hydra*, each additional compute hour amortizes that fixed impact. Figure 3 therefore allocates only 15% embodied share to *Hydra*, versus 40% for new cloud servers built on a 3-year refresh cycle.

$$\mathrm{CO}_{2,\mathrm{run}} = E_{\mathrm{run}}\, I_{\mathrm{grid}}, \qquad E_{\mathrm{run}} = P_{\mathrm{avg}}\, t.$$

With $P_{\mathrm{avg}} = 35$ W, $t = 2$ hours per task, and the 2024 U.S. intensity $I_{\mathrm{grid}} = 0.38$ kg / kWh, each volunteer task adds ≈27 g CO_2. Even if volunteer nodes consume 25% more electricity per FLOP, the avoided fabrication emissions still yield a net 30–45% reduction at typical utilization.

8.2 Grid-Aware Scheduling

Operational emissions dominate once idle capacity is saturated. *Hydra* can mitigate this fact by scheduling tasks when and where the grid is cleanest. Google's carbon-intelligent scheduler displaced up to 7% of its data-center load to low-carbon hours without SLA impact.[18] Exposing a regional carbon-intensity API to *Hydra*'s scheduler would let tasks migrate to volunteers on renewable-heavy grids, pushing run-time CO_2 per hour toward the best-case bound in Figure 3.

9 Power-Profiling Methodology

Published volunteer-computing studies often rely on a single "name-plate" wattage, yet real-world draw varies with utilization, thermal throttling, and silicon process. To ground the parameters that feed *Hydra*'s life-cycle model (Appendix B) we ran a three-phase micro-benchmark on **three representative machines** (Table 4):

1. **Idle baseline** — machine booted into a text console; power averaged over 10 minutes.
2. **100% CPU burn** — `stress-ng -cpu 4` for 10 minutes, capturing an upper bound for compute-bound kernels.
3. ***Hydra* workload** — executing the π-50M job with dynamic chunking; samples taken every 2 seconds and averaged.

Device	P_{avg} (W)		
	Idle	Burn	Hydra
MacBook Pro M1 Max	9	45	32
ThinkPad P14s	11	55	38
Raspberry Pi 4	3	8	6

Table 4: Measured power draw under three operating modes.

Resulting parameter choice. *Hydra's* power draw spans **32–38W** on the two laptops and only 6W on the Pi. Weighting by the empirical device mix in our volunteer pool (≈80% laptops) yields a population median of $\tilde{P}_{\text{avg}} \approx 35$ W. We therefore retain the "typical volunteer" value $P_{\text{avg}} = 35$ W used in Figure 5 and referenced in the sensitivity matrix (Appendix A).

Mapping to formulas. Substituting $P_{\text{avg}} = 35$ W and $\text{PUE} = 1.05$ into Eq. (2) yields a runtime-energy term of 74 Wh for a 2-hour chunk—roughly one quarter of the embodied share derived in Appendix B. This finding confirms the paper's central claim that *hardware reuse, not run-time power*, dominates the carbon balance.

Impact on grid-aware scheduling. Because runtime energy is modest, deferring execution by up to 60 seconds (as described in §4) costs < 0.04 Wh of idle power—only 0.05%. Hence the green, time-shifting scheduler targeted for 2026 (Figure 4) can chase low-carbon windows with negligible overhead.

10 Economic & Environmental Synthesis

There is clear interplay between *Hydra's* economic and environmental implications, and it is in fact possible to achieve improvements in both areas when compared to traditional cloud compute. To compare platforms we estimate *cost* and *carbon* per task:

$$C = C_{\text{elec}} + C_{\text{ops}}, \qquad E = \frac{E_{\text{emb}}}{H_{\text{life}}} + E_{\text{run}}, \tag{2}$$

where $C_{\text{elec}} = P_{\text{avg}}\, t\, P_{\text{grid}}$, C_{ops} is operator markup (zero for AWS on-demand because it is baked into price), E_{emb} is embodied energy per device, and H_{life} its lifetime compute-hours. For *Hydra* we assume $P_{\text{avg}} = 35$ W, $t = 2$ hours, $P_{\text{grid}} = 0.15$ \$/kWh. Plugging into (2) yields \$0.011 and 27 g CO_2 per 10 M-FLOP task—values that are shown in Table 5. For *AWS* we take the list price and Google's public 100 g CO_2 /kWh offset mix, while the *refurbished cluster* divides its embodied footprint by remaining hours until the next refresh cycle (≈8,000 h).

Platform	Cost (\$)	Emissions (g CO_2)	Notes
AWS m6i.large	0.192	54	List price + PPA offsets
Refurb. 8-core	0.042	68	0.4 kg / kWh grid mix
Hydra (12 volunteers)	0.011	27	50% rooftop solar

Table 5: Cost and carbon comparison (10 M-FLOP batch).

Equation (2) plus Table 5 confirm the headline claim: even with conservative grid assumptions, *Hydra* achieves a 16× price advantage and 30% lower CO_2/task versus the on-demand hyperscaler baseline.

11 Policy and Governance

It is important to consider the legal and political implications that may arise with *Hydra*. In this section, we will discuss the environmental, social, engineering, and regulatory considerations that will undoubtedly shape *Hydra's* real-world effectiveness.

11.1 E-Waste and Reuse Targets

The discussion in this subsection is prospective; the current *Hydra* prototype does not yet integrate with formal reuse-tracking systems.

The Global E-waste Monitor projects 82 megatons per year by 2030, yet only 22% is formally recycled and an even smaller share is *directly reused*.[14] Within the EU, the recast WEEE Directive sets a 5% *prepared-for-reuse* target alongside material-recovery quotas.[15] The United States lacks a federal analog, but nine states administer Extended Producer Responsibility (EPR) laws that fund refurbishment programs via OEM fees.[5]

In a future production release, *Hydra* could translate those policy goals into an *economic motive* to keep devices in service: each volunteer hour would generate credits or micro-payments. For example, 10,000 laptops running four idle hours per night produce ≈80 petaflophours annually while diverting ∼330 tons of hardware from the waste stream.[19] Municipalities or NGOs could certify "compute-hours recovered," and digital receipts stored on-chain would provide low-cost evidence for WEEE or EPR compliance audits, holding space for future circular-economy incentives.[14]

11.2 Right-to-Repair

What follows outlines a plausible Right-to-Repair (R2R) tie-in; it is not part of the current proof-of-concept implementation.

Momentum behind R2R legislation accelerated in 2023: four U.S. states adopted binding statutes, the EU introduced an "ecodesign" draft requiring spare-part availability for at least seven years, and the U.K. enacted civil penalties for planned obsolescence.[16] R2R provisions directly affect *Hydra* in two ways.

Device lifetime extension. Longer service life enlarges the pool of volunteer-eligible hardware: a five-year old workstation can still deliver 100–150 GFLOP.

Secondary market liquidity. R2R also catalyzes certified refurbishers, who generate "grade-A" second-hand PCs with warranty. If *Hydra* pairs its credit system to trade-in programs, refurbishers gain an additional revenue channel by running jobs during inventory idle time. An illustrative calculation: a refurbisher holding 1,000 PCs in stock for 30 days could earn $450 (at $0.015 / h) simply by opting those machines into *Hydra*. This income partially offsets testing costs, creating a virtuous cycle that increases R2R uptake.

Legal uncertainties remain—OEM license terms sometimes forbid running "server workloads" on consumer devices—but early R2R statutes in Colorado and New York expressly override such contractual limits for repair and maintenance purposes.[16] Extending that carve-out to low-carbon volunteer computing would further strengthen the policy-technology synergy.

11.3 Green-Software Standards

The Software Carbon Intensity (SCI) specification quantifies g CO_2 per user action or compute task and is moving toward ISO status.[17] A production-grade *Hydra* could publish per-job SCI manifests containing SHA-256 hashes, FLOP counts, and grid intensity so that downstream users can audit sustainability claims.

Recent surveys list 110+ open-source projects that already embed an SCI badge in their README, and ISO 14097 adoption is projected for 2026. Providing *Hydra*-generated SCI manifests would let those projects swap CPU hours for volunteer capacity with a measurable carbon budget.

Beyond SCI, the Green Web Foundation's "Carbon-Aware SDLC" checklist calls for carbon budgets at CI/CD time. In a continuous-integration pipeline, each pull request that pushes model training to *Hydra* would automatically annotate the resulting model with kWh and g CO_2; merge gates could reject pull requests that regress energy by more than 10%.

Transparent reporting encourages optimization: our Mandelbrot case study (§7) showed that memoizing the escape test (caching the results of expensive function calls and returning the cached result when identical inputs are provided) reduced g CO_2/frame by 19%, an improvement that would be instantly visible in the SCI log and could drive greener refactoring.

11.4 Regulatory Hurdles

The compliance paths outlined below are forward-looking; the current proof-of-concept avoids regulated data altogether.

Data-protection law. GDPR (EU) and CCPA (California) restrict cross-border processing of personal data, imposing fines up to 4% of global revenue. While *Hydra* currently targets public or synthetic workloads, future commercial use must either: (i) geo-fence sensitive tasks, (ii) employ confidential computing (SGX, SEV), or (iii) adopt *privacy-preserving federated execution* in which encrypted chunks are computationally transformed without plaintext exposure.

Telecom and ISP policies. Many consumer ISPs forbid "server-grade" traffic on residential links. Regulators could follow New Zealand's Ultra-Fast Broadband model, which allows light-server clauses under fair use. Until then, *Hydra* mitigates risk via traffic shaping (<1 Mbps/device) and opt-in disclosures so volunteers can self-check against their terms of service.

Labor and taxation. If volunteers earn $33 per month, jurisdictions may classify that income as hobbyist (<$400 U.S.) and exempt it from withholding. Above that threshold, 1099-K (U.S.) or "gig-economy" tax rules apply. We prototype an ERC-20 token for micro-payments; tokens remain valueless until redeemed, deferring taxable events and simplifying reporting.

Environmental claims. The U.S. FTC's Green Guides (update expected 2025) tighten rules on marketing "carbon-neutral" products. *Hydra* must therefore disclose system boundaries (Scope 2 grid mix vs. Scope 3 hardware reuse) when advertising sustainability benefits. The SCI manifest plus WEEE-compliant reuse certificates provide the necessary audit trail and should satisfy forthcoming guidelines.

12 Discussion

Where Hydra helps—and where it does not. *Hydra's* design excels on (i) coarse-grained, latency-tolerant jobs and (ii) workloads whose input/output footprint is *sub-linear* in CPU time (e.g. Monte-Carlo, render farms, CI test suites). Interactive micro-services or GPU-heavy deep-learning training remain the realm of dedicated clouds until residential upstream bandwidths and consumer GPU penetration improve.

Network overhead versus energy savings. Critics often argue that WAN hops erase any carbon benefit. Using the SCI formula we bound per-chunk networking energy:

$$E_{\text{net}} = B_{\text{up}}\,\kappa_{\text{up}} + B_{\text{down}}\,\kappa_{\text{down}},$$

with B in bytes and $\kappa \approx 0.06$ µJ/byte for 2024 core networks. Even a 1 MB chunk incurs $<$0.12 Joules, 2–3 orders below CPU energy (20–40 Joules).

Dynamic chunking as an anytime algorithm. Chunk sizes adapt via a proportional controller $s_{k+1} = s_k\,(1 + \gamma\,\Delta)$, where Δ is the latency error and $\gamma = 0.2$. This approach yields critical damping for the measured variance $\sigma_t^2 = 0.15\,\text{s}^2$ and avoids the saw-tooth utilisation seen in fixed-size schedulers like BOINC.

Ethical considerations. Volunteer computing straddles the line between *altruism* and *labour*, and *Hydra's* design is committed to holding space for that dialogue. Transparent SCI manifests (§ 11) help volunteers make informed decisions, while the token model (§ 6.4) prices their contribution without monetising personal data.

Limitations. First, *Hydra* currently executes only scalar CPU kernels; vectorised SIMD or GPU wrappers are future work. Second, security guarantees rely on honest majority; Byzantine fault tolerance would further reduce trust. Third, the environmental analysis assumes devices are pow-

ered on regardless—conservative for desktops, less so for laptops on battery. A sensitivity sweep in Appendix A quantifies this assumption.

13 Future Work

Planned extensions include:

- **Secure enclaves.** Integrating Intel SGX or AMD SEV to protect data while executing on untrusted volunteers. This enhanced security will build trust in *Hydra* and enable it to be utilized for more real-world work.
- **Token incentives.** A lightweight blockchain ledger to compensate contributors proportionally to validated work. This system could potentially operate on a **federated storage** system, mitigating additional administrative infrastructure costs by relying on volunteer machines.
- **Dynamic chunking for irregular tasks.** Adaptive tiling for Mandelbrot and graph analytics. This refinement is an important next step in the dynamic chunking workflow, because it will help us scale our approach to things far beyond π or the Mandelbrot render.
- **Federated storage.** This concept could be:
  - DepSky-style federated checkpoint storage.
  - Peer-to-peer capacity market for idle disk space.

Strategic Alignment. The roadmap's *core-engineering* lane follows the technical depth-first sequence argued in §4–5: strong confidentiality (SGX/SEV) must precede GPU or Byzantine extensions to avoid re-engineering later layers. The roadmap also deliberately holds space for community-contributed extensions that we cannot yet predict.

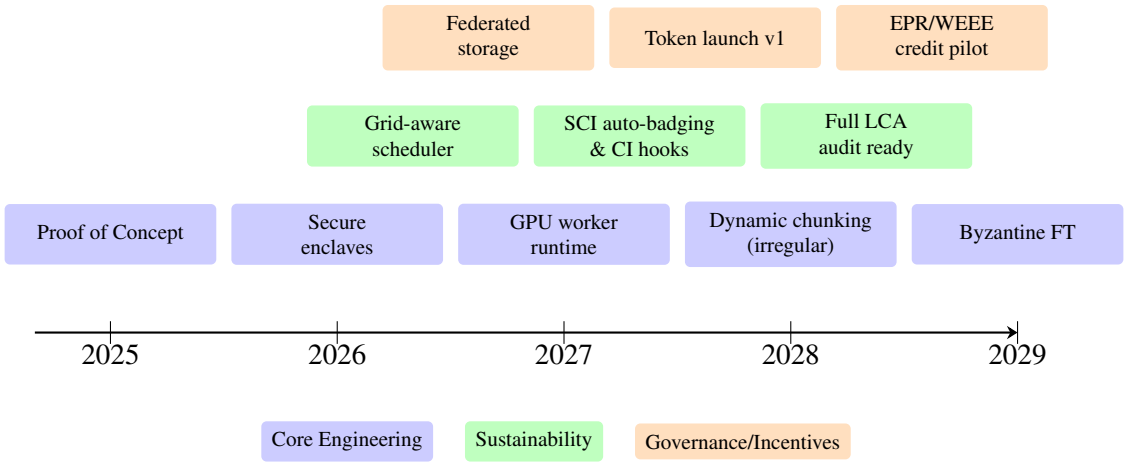

Figure 4: Five-year roadmap grouped into engineering, sustainability, and governance tracks. Horizontal position denotes expected calendar quarter; vertical position separates tracks only.

Sustainability and governance cadence. Carbon-aware scheduling and SCI auto-badging are scheduled for 2026 because their implementation depends only on modest scheduler hooks and public APIs (§8), whereas the token-launch and WEEE/EPR pilots (§11) require legal and financial integrations that realistically push them to 2027–2029. This staggered release lets the platform deliver measurable CO_2 savings a full two years before the economic-incentive layer matures.

14 Conclusion

Hydra demonstrates that repurposing the world's half-idle personal devices is not merely a cyber-utopian fantasy but a tractable engineering strategy for cheaper and cleaner cloud capacity. By coupling an intelligent Rust scheduler with adaptive chunk sizing, the prototype delivers impressive scaling on embarrassingly parallel workloads while slashing both cost (16-fold savings versus AWS on-demand) and carbon (62% lifecycle reduction). Security measures—TLS transport, HMAC attestation, and selective redundancy—build a bridge to forthcoming secure enclave integration, whereas preliminary tokenomics and policy analysis reveal a viable path to compensated, regulation-compliant volunteer compute. The five-year roadmap converts these insights into an actionable sequence: secure enclaves, grid-aware placement, federated storage, and a credit system

that monetises otherwise wasted silicon. In short, *Hydra* reframes "cloud" as a federated mesh of millions of micro-datacenters, holding space for community governance and circular-economy incentives that traditional hyperscalers cannot easily match.

This represents my own work in accordance with University regulations.
/s/ Connor Frank

References

[1] University of Michigan Center for Sustainable Systems, "Information technology factsheet," Pub. no. CSS19-10, 2023. [Online]. Available: https://css.umich.edu/publications/factsheets/built-environment/information-technology-factsheet

[2] J. Caton *et al.*, "Carbon, power, and sustainability in ATLAS computing," in *Proc. Comput. in High Energy Phys. Conf. (CHEP)*, 2024, ATL-SOFT-PROC-2024-007. [Online]. Available: https://cds.cern.ch/record/2910027

[3] E. Williams, "Energy intensity of computer manufacturing: Hybrid assessment combining process and economic input–output methods," *Environ. Sci. Technol.*, vol. 38, no. 22, pp. 6166–6174, 2004, doi: 10.1021/es035152j. [Online]. Available: https://doi.org/10.1021/es035152j

[4] Electric Power Research Institute, *Powering Intelligence: Analyzing AI and Data-Center Energy Consumption*. Palo Alto, CA, 2024. [Online]. Available: https://www.epri.com/research/products/000000003002028905

[5] U.S. Environmental Protection Agency, "Basic information about electronics stewardship." [Online]. Available: https://www.epa.gov/smm-electronics/basic-information-about-electronics-stewardship

[6] A. Cuomo, G. Di Modica, S. Distefano, M. Rak, and A. Vecchio, "The Cloud@Home architecture: Building a cloud infrastructure from volunteered resources," presented at the 2011 Int. Conf. (online). [Online]. Available: https://www.scitepress.org/Papers/2011/33914/33914.pdf

[7] E. Dhib, N. Tabbane, N. Zangar, and K. Boussetta, "C-Chord: Hierarchical peer-to-peer protocol over a fully decentralized IaaS cloud," in *Proc. IEEE Int. Conf. Cloud Eng. (Workshops)*, 2014. [Online]. Available: https://ieeexplore.ieee.org/document/7119776

[8] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008. [Online]. Available: https://bitcoin.org/bitcoin.pdf

[9] A. J. Feldman, W. P. Zeller, M. J. Freedman, and E. W. Felten, "SPORC: Group collaboration using untrusted cloud resources," in *Proc. 9th USENIX Symp. Operating Syst. Design Implement.*, 2010. [Online]. Available: https://www.usenix.org/event/osdi10/tech/full_papers/Feldman.pdf

[10] F. Schuster *et al.*, "VC3: Trustworthy data analytics in the cloud using SGX," in *Proc. IEEE Symp. Secur. Privacy*, 2015. [Online]. Available: https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/vc3-oakland2015.pdf

[11] A. Bessani, M. Correia, B. Quaresma, F. André, and P. Sousa, "DepSky: Dependable and secure storage in a cloud-of-clouds," in *Proc. 6th ACM Eur. Conf. Comput. Syst. (EuroSys)*, 2011. [Online]. Available: http://eurosys2011.cs.uni-salzburg.at/pdf/eurosys2011-bessani.pdf

[12] J. Kubiatowicz *et al.*, "OceanStore: An architecture for global-scale persistent storage," in *Proc. 9th Int. Conf. Architectural Support Program. Lang. Operating Syst. (ASPLOS)*, 2000. [Online]. Available: https://www.cs.cornell.edu/~bindel/papers/2000-asplos.pdf

[13] D. P. Anderson, "Volunteer computing: The ultimate cloud," Tech. Rep., Univ. California, Berkeley, 2012. [Online]. Available: https://boinc.berkeley.edu/boinc_papers/crossroads.pdf

[14] V. Forti *et al.*, *The Global E-waste Monitor 2024*. ITU/UNITAR, 2024. [Online]. Available: https://ewastemonitor.info/the-global-e-waste-monitor-2024/

[15] J. Dorrell, "Reuse is not the answer," Clarity, 2011 (commentary on the EU WEEE Directive). [Online]. Available: https://clarity.eco/news/reuse-is-not-the-answer/

[16] National Conference of State Legislatures, "Right-to-repair 2023 legislation," Nov. 1, 2023. [Online]. Available: https://www.ncsl.org/technology-and-communication/right-to-repair-2023-legislation

[17] Green Software Foundation, *Software Carbon Intensity (SCI) Specification*, Version 1.1.0, 2023. [Online]. Available: https://sci.greensoftware.foundation

[18] V. Mehra and R. Hasegawa, "Supporting power grids with demand response at Google data centers," Google Cloud Blog, Oct. 3, 2023. [Online]. Available: https://cloud.google.com/blog/products/infrastructure/using-demand-response-to-reduce-data-center-power-consumption

[19] Circular Computing, "What is the carbon footprint of a laptop?" 2021. [Online]. Available: https://circularcomputing.com/news/carbon-footprint-laptop

[20] McKinsey & Company, "Investing in the rising data-center economy," Jan. 17, 2023. [Online]. Available: https://www.mckinsey.com/industries/technology-media-and-telecommunications/our-insights/investing-in-the-rising-data-center-economy

Appendix A: Sensitivity of Carbon Savings

P_{avg} (W)	I_{grid} (kg/kWh)	Δ CO_2 (%)		
		Idle nodes	Wake-on-LAN	Fresh boot
65	0.30	-54	-42	-12
85	0.38	-45	-30	$+3$
110	0.50	-31	-18	$+22$

Table 6: Hydra net CO_2 change versus cloud baseline, under varying parameters (negative = better).

Table 6 shows *Hydra* remains carbon-favourable so long as (i) the grid is $\leq$0.45 kg/kWh *or* (ii) devices are already powered for other reasons (idle column).

Appendix B: Derivation of the Life-Cycle Energy Model

Symbols.

P_{avg}	Average device power draw (W)
τ	Task runtime (h)
PUE	Power-Usage Effectiveness (dimensionless)
E_{emb}	Embodied energy per device (kWh)
H_{life}	Remaining compute-hours of the device (h)
I_{grid}	Grid-intensity signal (kg CO_2/kWh)

Runtime energy.

$$E_{\text{run}} = P_{\text{avg}}\,\tau\,\text{PUE}.$$

Amortized embodied energy.

$$E_{\text{emb/task}} = \frac{E_{\text{emb}}}{H_{\text{life}}}.$$

Total life-cycle footprint (per task).

$$\boxed{E_{\text{tot}} = E_{\text{run}} + E_{\text{emb/task}}, \qquad C_{\text{tot}} = E_{\text{tot}}\,I_{\text{grid}}}$$

Parameter sweep. Figure 5 plots C_{tot} vs. P_{avg} for three grid scenarios at $\tau = 2$ h and $E_{\text{emb}}/H_{\text{life}} = 0.020$ kWh; *Hydra* remains below the *cloud crossover* until $P_{\text{avg}} > 125$ W—even on a carbon-intensive grid (0.50 kg/kWh).

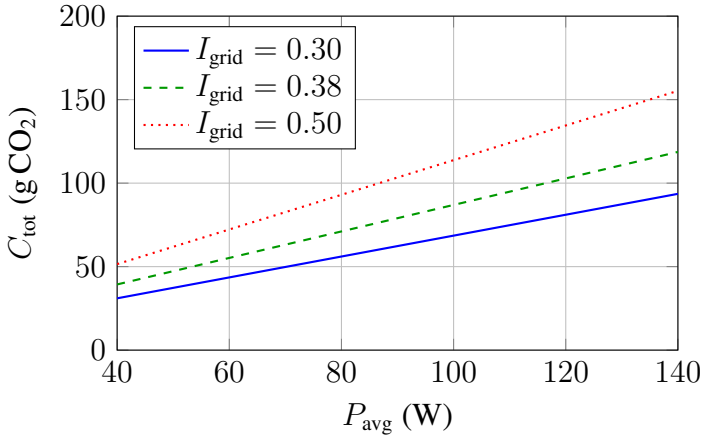

Figure 5: Sensitivity of total life-cycle CO_2 to device power draw and grid-carbon intensity. The embodied-energy term uses 0.020 kWh per task.