ColabHive: A Distributed Hive–Mind Architecture for Energy–Aware Collaborative AI

(Draft)
colabhive.com

***Abstract*—Recent advances in large language models (LLMs) have been enabled by highly centralised GPU data centres, concentrating computational power—and thus control and environmental footprint—in the hands of a few large actors. At the same time, millions of consumer and prosumer GPUs remain underutilised after the collapse of cryptocurrency mining profitability. This paper introduces *ColabHive*, a distributed "hive–mind" architecture that repurposes heterogeneous hardware (from RTX 3090–class GPUs to CPU–only nodes) into a collaborative network of specialised AI agents. Instead of relying on a single monolithic model, ColabHive decomposes user tasks into subproblems and routes them to an ensemble of expert agents hosted on geographically dispersed nodes, chosen according to a multi–objective cost function that accounts for latency, capability, and energy cost.**

We present the system architecture, a class of node and model selection algorithms, and an energy/carbon model for evaluating different deployment scenarios. Using realistic hardware parameters and recent data on data centre electricity use and grid carbon intensity, we estimate the potential computational capacity and environmental impact of a large–scale ColabHive deployment. We argue that, when properly orchestrated, a global network of modest, specialised models running on repurposed hardware can deliver competitive AI capabilities while reducing marginal energy demand relative to building ever larger centralised clusters. We illustrate the orchestration logic with end–to–end examples for simple and complex prompts, showing how the system assembles low–power expert teams rather than defaulting to heavyweight generalist models.

I. Introduction

Large language models and foundation models have driven a rapid expansion in data centre electricity demand. Recent estimates suggest data centres consume on the order of 400–450 TWh per year (about 1.5% of global electricity), with AI workloads already contributing a significant and rapidly growing fraction of this demand [1], [2]. Forecasts for 2030 project AI data centre electricity consumption in the range of 200–400 TWh, potentially rivaling the current annual consumption of mid–sized countries. At the same time, the carbon intensity of grid electricity, while falling, remains on the order of hundreds of grams of CO_2–equivalent per kWh globally [3], implying a substantial climate impact for uncontrolled scaling of AI infrastructure.

Concurrently, the cryptocurrency boom and subsequent bust have left a large installed base of high–end consumer GPUs (e.g., RTX 3090/4090) and small GPU rigs underutilised. These devices are often powered on, connected, and underloaded, representing stranded computational capacity. While some decentralised compute marketplaces have emerged, they typically expose raw GPU time rather than an integrated AI abstraction, and do not tackle task decomposition, model selection, or energy–aware routing.

This work explores an alternative design space: a *distributed hive–mind* that treats the world's heterogeneous hardware as a substrate for a network of specialised AI agents, coordinated by higher–level orchestrators. Instead of a single giant model, ColabHive leans on (i) modular expert models, (ii) intelligent orchestration of agents per task, and (iii) energy– and latency–aware selection of nodes, aiming to right–size the compute and model to the task at hand.

Our contributions are:

- We propose a system architecture for ColabHive, distinguishing orchestrator nodes, expert nodes with GPUs, and lightweight CPU–only nodes.
- We define a multi–objective node and model selection framework that accounts for latency, capability, and energy cost, enabling energy–aware routing of tasks.
- We develop an approximate energy and carbon model for ColabHive, parameterised by the size and utilisation of the network, and compare illustrative scenarios with centralised data centre deployments. This comparison quantifies the marginal energy and carbon savings from utilizing stranded compute capacity over building equivalent new centralized infrastructure.
- We present qualitative case studies of orchestration for simple and complex prompts, highlighting how low–power specialised agents can replace monolithic inference calls.

II. Background and Motivation

A. Centralisation of AI Compute

The state–of–the–art in LLMs has largely been driven by hyperscale providers, deploying clusters of tens of thousands of data centre GPUs. This centralisation yields economies of scale, but it also:

1) concentrates control over AI capabilities,
2) requires large, capital–intensive data centres with complex cooling and power infrastructure, and
3) amplifies the environmental footprint: even with improving power usage effectiveness (PUE), projections indicate that AI could account for a substantial fraction of future data centre power demand [2], [6], [**?**].

B. Stranded Prosumer GPUs

In parallel, the cryptocurrency mining ecosystem has left behind an installed base of consumer and prosumer GPUs whose utilisation has dropped sharply with declining mining

revenues. Individual rigs often feature 4–12 cards in the 200–450 W TDP range; an RTX 3090, for instance, has a nominal TDP of around 350 W and can draw 350–450 W at load in typical compute workloads [7], [8]. While exact counts are uncertain, even a network of one million such GPUs at 10–30% utilisation represents a nontrivial aggregate capacity.

These devices are often in locations unsuited to hosting full data centres but well suited to intermittent AI workloads:

- residential or small office settings with existing power and connectivity,
- small hosting facilities or edge locations,
- geographies where grid carbon intensity is falling due to increased renewables.

However, harnessing such hardware presents unique challenges: heterogeneity in specifications (VRAM, thermal design, driver versions), intermittency due to user behaviour or residential power constraints, variable network connectivity, and suboptimal cooling conditions. These factors necessitate robust reputation metrics $R(n)$ and dynamic re–routing capabilities to maintain service quality—issues not encountered in traditional hyperscale data centres.

C. Right–Sizing Compute and Models

Current practice often defaults to applying large, general–purpose LLMs to tasks that could be solved by:

- smaller task–specific models,
- non–transformer models (e.g., tabular models, classical ML), or
- simple rule–based logic.

From an ecological perspective, using a 70B–140B parameter LLM for a short classification or routing task is analogous to using a freight train to move a feather: technically feasible but energetically wasteful. A more sustainable approach is to:

1) decompose tasks into subcomponents,
2) assign each subtask to an appropriate expert model (which may be much smaller),
3) run these models on suitable hardware (GPU or CPU) considering energy efficiency and latency.

ColabHive aims to operationalise this principle at network scale.

III. System Architecture

ColabHive's architecture is built around a registry–mediated distributed system where orchestrator nodes coordinate interactions among heterogeneous expert nodes. Figure 1 illustrates the high–level data flow: a user request is received by an orchestrator, which consults the registry to discover suitable expert nodes, dispatches subproblems to selected agents, and synthesises a coherent response.

A. Node Types

ColabHive organises hardware into several logical node types:

- **Orchestrator Nodes**: Responsible for understanding user requests, planning task decomposition, selecting agents, and aggregating results. Orchestrators typically run medium–sized LLMs and control logic. They are latency–sensitive and often placed closer to users.
- **GPU Expert Nodes**: Machines with one or more GPUs (e.g., RTX 3090/4090, A100 class) hosting specialised models. Each node advertises its capabilities: VRAM, supported model families, approximate throughput, geographic region, and energy profile.
- **CPU–Only Nodes**: Devices without GPUs that can run small models (e.g., 0.5–3B parameters), embedding models, or classical ML. They are useful for low–power tasks such as routing, filtering, or light summarisation.
- **Registry and Control Plane**: A logically central but physically replicated service that tracks node metadata, model versions, and reputational scores, and exposes a discovery API to orchestrators.

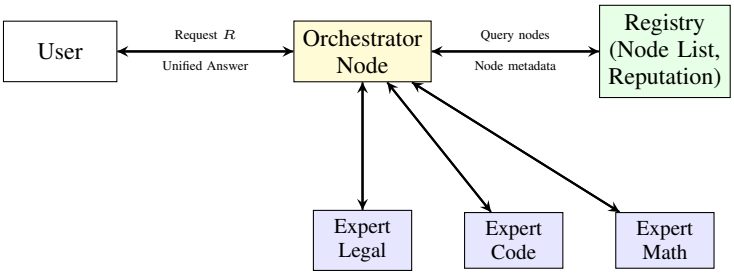

Fig. 1. ColabHive system architecture. The orchestrator queries the registry for capable nodes, dispatches tasks to selected experts, and aggregates results for the user.

Nodes may co–locate roles (e.g., an orchestrator might also host some experts) depending on resource availability.

B. Agent Types

An *agent* in ColabHive is a model plus optional tool–calling logic. Categories include:

- **LLM Experts (GPU–heavy)**: Language models fine–tuned for specific domains (e.g., quantitative finance, legal reasoning, code synthesis). Typically require 16–24GB+ VRAM for 7B–34B models.
- **Perception and Embedding Models (GPU–preferred)**: Vision encoders, sentence embedders, and multimodal models. Benefit from parallelism but can run on modest GPUs.
- **Tabular and Numeric Models (CPU/low–power)**: Gradient–boosted trees, MLPs, or future ASIC–accelerated tabular inference cores for structured data. Often efficient on CPUs with minimal energy footprint.
- **Tools and Simulators (CPU)**: Deterministic services such as pricing engines, solvers, or backtesting modules. No GPU required.

Agents expose a uniform interface to the orchestrator: a schema describing input, output, and estimated compute cost.

IV. Task Decomposition and Orchestration

A. High–Level Flow

Given a user request R, ColabHive proceeds in four stages:

1) **Intent Understanding**: An orchestrator LLM M_o receives R and produces:
   - a semantic representation of the task,
   - a set of candidate subproblems $\{S_i\}$,
   - a set of constraints (latency budget, privacy, energy mode, etc.).

   The orchestration logic relies on an inner loop where M_o uses reflection and tool–calling capabilities (similar to Chain–of–Thought or recursive agent–based planning) to iteratively refine the task decomposition and select the optimal agent composition.
2) **Plan Synthesis**: A planning module selects a set of agents $\{A_j\}$ and an execution graph G connecting subproblems to agents, possibly in parallel.
3) **Node and Model Assignment**: For each agent invocation in G, the orchestrator queries the registry for compatible nodes and selects concrete node–model pairs (n, A_j) based on a cost function (Section V).
4) **Execution and Aggregation**: Results from agents are returned to the orchestrator, which may perform further meta–reasoning before synthesising the final response to the user.

B. Example: Simple Prompt

For a simple prompt such as: *"Summarise this 1 000–word article in one paragraph."*, the orchestrator might:

- detect a low complexity summarisation problem,
- prefer a small summarisation model (e.g., 1–3B parameters) running on a nearby CPU node or low–end GPU,
- avoid invoking larger, more power–hungry LLMs entirely.

Latency and energy requirements are modest; a single agent suffices.

C. Example: Complex Multi–Domain Prompt

For a complex prompt such as:

> "Design a quantitative investment strategy for emerging market bonds, outline a compliant fund structure for European retail investors, and draft a client–facing explanation of the risks in plain language."

the orchestrator may:

1) split the task into at least three subproblems:
   - strategy design and backtest constraints,
   - legal/regulatory structuring,
   - communication and risk explanation.
2) route each to different expert agents:
   - a finance LLM expert with access to tabular models for risk metrics,
   - a legal expert LLM fine–tuned on fund documentation,
   - a communication–focused LLM specialised in lay explanations.
3) assign these agents to nodes with appropriate capabilities and energy profiles, potentially in different geographic regions.

The orchestrator then aggregates the partial outputs into a single coherent answer, enforcing consistency (e.g., ensuring that legal and strategy sections reference the same constraints).

V. Node and Model Selection

A. Cost Model

For each candidate node n and agent A combination, the orchestrator computes a cost:

$$C(n, A; u) = \alpha \cdot L(n, u) + \beta \cdot E(n, A) + \gamma \cdot P(n, A) + \delta \cdot R(n) \tag{1}$$

where:

- $L(n, u)$ is an estimate of end–to–end latency from user u to node n,
- $E(n, A)$ is the expected energy use (e.g., kWh) for running agent A on node n,
- $P(n, A)$ is a penalty for capacity constraints (e.g., queue length, VRAM saturation),
- $R(n)$ is a reputational or reliability penalty,
- $\alpha, \beta, \gamma, \delta$ are tunable weights reflecting the operating mode.

Operating modes (e.g., *eco*, *balanced*, *performance*) map to different weight vectors:

$$(\alpha, \beta, \gamma, \delta)_{\text{eco}} \neq (\alpha, \beta, \gamma, \delta)_{\text{performance}}.$$

The orchestrator selects the (n, A) pair(s) with minimal cost subject to functional constraints (e.g., A must support the required capability).

B. Latency Estimation

Latency $L(n, u)$ can be estimated via:

- periodic round–trip probes to representative vantage points,
- passive measurement of past requests from similar regions,
- geographic heuristics when no direct history exists.

For multi–agent plans, end–to–end latency is modelled as the critical path length over the execution graph G.

C. Energy Estimation

For a node n with device power draw P_{device} and estimated active time t for an agent call, energy consumption is:

$$E(n, A) \approx \frac{P_{\text{device}} \cdot t}{\eta_{\text{sys}}}, \tag{2}$$

where η_{sys} accounts for system–level overhead (CPU, memory, cooling). For a consumer GPU like an RTX 3090, empirical data suggests a device power of roughly 350 W at nominal TDP, with real–world compute loads reaching 350–450 W depending on configuration [7], [8].

For CPU–only nodes running small models, P_{device} may be on the order of 50–150 W, with longer runtimes t but lower instantaneous draw. The orchestrator can choose between a short, high–power GPU execution and a longer, low–power CPU execution depending on the objective.

D. Heterogeneous Task Matching

Given a distribution over task types, ColabHive can maintain a *capability profile* over agents and nodes, updated via online learning. For a new task, the orchestrator:

1) predicts an appropriate model size and type (e.g., 3B CPU, 7B GPU, 34B GPU),
2) queries the registry for nodes hosting such models,
3) samples a small candidate set $\mathcal{N}$ and evaluates $C(n, A)$,
4) selects a subset $\mathcal{N}' \subset \mathcal{N}$ for actual execution.

This approach allows the system to avoid defaulting to the largest available model, reducing energy use without sacrificing quality on many tasks.

E. Greedy Node Selection Algorithm

Given a task decomposition graph G (from the orchestrator) and a set of candidate nodes $\mathcal{N}$ discovered from the registry, a baseline greedy scheduling algorithm operates as follows:

```
for each agent invocation a in topological_order(G):
    candidates = discover_nodes(a.capabilities)
    for n in candidates:
        cost[n] = C(n, a, user_location)  # Eq. (1)
    assign a to argmin_n cost[n]
    update node capacity and availability
```

This greedy approach provides a simple starting point. More sophisticated strategies—such as multi–armed bandits for exploration–exploitation trade–offs, reinforcement learning for adaptive routing, or constraint–satisfaction solvers for global optimization over G—can be built on top of the same cost function C (Equation 1).

The key insight is that the multi–objective cost function remains the same regardless of scheduling complexity, enabling incremental deployment: the system can launch with greedy scheduling and evolve towards more advanced methods as operational data accumulates.

VI. Energy and Carbon Modelling

A. Global Context

Global grid carbon intensity in 2023 was approximately 480 gCO_2/kWh on average, with projections indicating a decline to around 400 gCO_2/kWh by 2027 as the share of low–carbon generation rises [3], [4]. Data centres currently consume an estimated 415 TWh per year, about 1.5% of global electricity, and AI workloads account for an estimated 10–50 TWh of that, with several studies projecting rapid growth towards 200–400 TWh by 2030 [1], [2], [5].

B. ColabHive Fleet Scenarios

Consider a simplified ColabHive deployment with:

- N_g GPU nodes, each with an RTX 3090–class device (350 W TDP), and
- N_c CPU–only nodes (100 W TDP for AI workloads).

Assume average utilisation factors u_g and u_c (fraction of time actively running AI tasks), and $\eta_{\text{sys}} \approx 1$ (we ignore additional cooling overhead at the node level for a conservative estimate). Annual energy consumption is:

$$E_{\text{year}} \approx (P_g N_g u_g + P_c N_c u_c) \cdot T, \tag{3}$$

where $P_g = 0.35$ kW, $P_c = 0.10$ kW, and $T = 8760$ h/year.

As an illustrative GPU–only scenario, let $N_g = 10^6$ RTX 3090–class nodes at $u_g = 0.2$ (20% utilisation) and $N_c = 0$. Then:

$$E_{\text{year}} \approx 0.35 \cdot 10^6 \cdot 0.2 \cdot 8760 \approx 6.1 \times 10^{11} \text{ Wh} = 0.61 \text{ TWh}.$$

At $N_g = 10^7$ under the same utilisation, this scales to ≈ 6.1 TWh/year. Even such a large network—tens of millions of prosumer GPUs—would still consume less electricity than projected AI data centre demand in 2030, but would represent a nontrivial fraction of global AI compute capacity.

C. Carbon Impact and Dynamic Routing

Grid carbon intensity varies by region and time of day. Let $I_{\text{CO2}}(n, t)$ denote the carbon intensity (gCO_2/kWh) at node n at time t. The carbon cost of executing agent A on node n becomes:

$$C_{\text{carbon}}(n, A, t) = E(n, A) \cdot I_{\text{CO2}}(n, t). \tag{4}$$

ColabHive can incorporate $I_{\text{CO2}}(n, t)$ into the multi–objective cost function (Equation 1), modifying the energy term:

$$C(n, A; u, t) = \alpha \cdot L(n, u) + \beta \cdot E(n, A) \cdot I_{\text{CO2}}(n, t) + \gamma \cdot P(n, A) + \delta \cdot R(n).$$

This enables *carbon–aware workload shifting*: when latency constraints permit, the orchestrator can route tasks to regions with cleaner grids (e.g., Nordic countries with high renewable penetration) or defer non–urgent tasks to times of day when renewable generation is high.

For aggregate carbon emissions, assuming an average grid intensity $\bar{I}_{\text{CO2}}$ (e.g., 480 gCO_2/kWh globally):

$$\text{CO2}_{\text{year}} \approx E_{\text{year}} \cdot \bar{I}_{\text{CO2}}. \tag{5}$$

For $E_{\text{year}} = 6.1$ TWh and $\bar{I}_{\text{CO2}} = 480$ g/kWh:

$$\text{CO2}_{\text{year}} \approx 6.1 \times 10^9 \text{ kWh} \cdot 0.48 \text{ kg/kWh} \approx 2.9 \text{ Mt } CO_2.$$

These calculations highlight that:

- A very large ColabHive—if fully utilised—would have a non–negligible footprint.
- However, ColabHive primarily reuses existing hardware that might otherwise be idle or underused, so its *marginal* hardware manufacturing impact is limited.
- Dynamic carbon–aware routing can significantly reduce emissions by exploiting spatial and temporal variations in grid carbon intensity. The magnitude of savings depends on regional grid mixes, latency constraints, and workload flexibility, but double–digit percentage reductions in CO2_{year} relative to geographically uniform routing are plausible for delay–tolerant tasks. Quantifying this precisely is left for future empirical work.

D. Right–Sizing Models and Tasks

A key lever for sustainability is the choice of model per task. Suppose:

- A large 70B LLM on a high–end GPU consumes $P_L \approx 400$ W for $t_L = 1$ s per query.
- A small 3B model on CPU consumes $P_S \approx 100$ W for $t_S = 2$ s per query.

Then per–query energy is roughly:

$$E_L \approx \frac{400 \text{ W} \cdot 1 \text{ s}}{3600} \approx 0.11 \text{ Wh}, \quad E_S \approx \frac{100 \text{ W} \cdot 2 \text{ s}}{3600} \approx 0.056 \text{ Wh}.$$

If a given task can be adequately solved by the small model, using it halves energy per query. In multi–agent settings, combining multiple small experts may still be cheaper than invoking a single giant model, especially when some subtasks can be handled by non–LLM tools.

VII. Case Studies: Orchestrator Behaviour

A. Simple Classification Prompt

Prompt:

> "Classify this support ticket into one of: billing, technical issue, account management."

Orchestrator behaviour:

1) Intent: short text classification; no long–form reasoning needed.
2) Plan: use a small text classifier agent (e.g., distilled transformer or even a logistic regression on embeddings).
3) Node selection: choose a nearby CPU–only node or low–power GPU node, aiming to minimise $L + \beta E$.
4) Execution: run the classifier and return label.

No large LLM is invoked; energy per query is dominated by small CPU compute.

B. Multi–Agent Financial and Legal Prompt

Prompt:

> "Given this 10–year time series of bond yields and FX rates, propose a carry trade strategy, estimate its historical drawdown, determine if it would comply with UCITS rules for a retail fund, and draft a 2–page risk explanation for non–expert investors."

Orchestrator behaviour (high–level):

1) Intent: multi–domain reasoning (quantitative finance, legal, communication).
2) Plan:
   - Agent A_{quant}: access a tabular model and backtesting engine to produce strategy metrics.
   - Agent A_{legal}: evaluate compliance with UCITS and highlight constraints.
   - Agent A_{comms}: transform technical content into a layperson–friendly explanation.
3) Node selection:
   - A_{quant} on a GPU node with local access to the numeric engine.
   - A_{legal} on a legal LLM expert, possibly on a different GPU node.
   - A_{comms} on a medium LLM that specialises in summarisation and explanation.
4) Execution: run agents partially in parallel, then perform a final pass in the orchestrator to ensure consistency across sections.

In an eco mode, the orchestrator may:

- prefer smaller experts (e.g., 13B instead of 70B) when empirical calibration shows comparable performance on this task type,
- route the communication subtask to a CPU–friendly summariser when latency constraints are loose.

VIII. Ecological and Socio–Technical Discussion

ColabHive's design interacts with sustainability and governance in several ways:

- **Hardware Reuse**: By repurposing existing GPUs and CPUs, ColabHive reduces the need for new hardware manufacture compared to building equivalent capacity from scratch, avoiding the embodied emissions of new chips and data centre infrastructure.
- **Energy–Aware Routing**: The node selection cost function can incorporate regional carbon intensity, prioritising nodes on cleaner grids when latency budgets allow, and enabling users to opt into "green" modes explicitly.
- **Democratisation of Compute**: Distributed orchestration allows communities, research groups, and individuals to contribute and access AI capacity, breaking the monopoly of a few hyperscale providers and enabling more diverse innovation.
- **Bias Mitigation through Decentralisation**: The distributed nature of ColabHive can help diversify data sources and modelling approaches, which may reduce the dominance of biases present in any single monolithic model. While decentralisation is not a guarantee of fairness, it enables alternative local and specialised agents to coexist and provide complementary perspectives, potentially reducing the propagation of biases inherent to single–provider AI systems.
- **Risk of Rebound and Marginal Demand**: Making AI compute cheaper and more accessible may increase total demand (a rebound effect). Additionally, even if ColabHive reuses idle hardware, activating previously dormant GPUs introduces new marginal electricity demand on the grid. While this demand may be cleaner (due to carbon–aware routing) and avoids the embodied emissions of new hardware, it is not zero–cost. Mitigation requires governance mechanisms (e.g., usage caps, carbon–aware pricing, protocol–level energy budgets) built into the system from its inception. Careful monitoring of aggregate network energy consumption relative to displaced centralised capacity is essential to validate the sustainability thesis.

IX. Roadmap and Open Challenges

A. Reliability and Fault Tolerance

A key concern for any distributed system built on prosumer hardware is reliability. Residential nodes face intermittency (power outages, voluntary disconnection), variable network latency, and heterogeneous configurations. ColabHive mitigates these challenges via:

- **Task Decomposition as Fault Isolation**: Breaking requests into parallel subproblems naturally isolates failures. If one expert node fails mid–execution, only that subtask must be re–routed, rather than restarting a monolithic inference call.
- **Dynamic Reputation ($R(n)$)**: Nodes that consistently fail or exhibit high latency accumulate penalty in $R(n)$, reducing their selection probability. This provides an adaptive mechanism for avoiding unreliable nodes.
- **Redundancy and Hedging**: For latency–critical tasks, the orchestrator can dispatch duplicate requests to multiple nodes and accept the first valid response, a technique used in distributed systems to mask tail latency.

Quantifying the trade–off between reliability and energy efficiency (e.g., redundant invocations increase energy but reduce user–facing latency variance) is a subject for empirical study.

B. Orchestration Overhead

A potential critique is that multi–agent orchestration incurs its own energy cost:

$$E_{\text{total}} = E_{\text{orchestrator}} + \sum_j E(n_j, A_j).$$

We acknowledge this overhead. However, for tasks solvable by small specialised models, the total energy can still be lower than invoking a large monolithic LLM. For example, a 3B summarisation model on CPU (0.056 Wh per query) plus a lightweight orchestrator call (perhaps 0.02 Wh for intent understanding on a small routing model) totals 0.076 Wh—still significantly less than a 70B model at 0.11 Wh.

The key is *right–sizing*: as task complexity increases, the overhead of orchestration becomes amortised across the savings from decomposing the problem. Measuring real–world orchestration overhead in a deployed prototype is a priority.

C. Security and Data Sensitivity

For enterprise adoption, the transfer of sensitive data to untrusted prosumer nodes is a showstopper. ColabHive must support:

- **Differential Privacy or Federated Learning**: For tasks involving sensitive datasets, agents can run locally on data–owner nodes, with only model updates or aggregate statistics shared.
- **Secure Enclaves (TEE/SGX)**: Nodes can offer trusted execution environments to run agents on encrypted data, preventing host–level inspection.
- **Task Partitioning**: The orchestrator can route sensitive subproblems to vetted "trusted tier" nodes (e.g., small data centres or certified hardware) while using prosumer nodes for non–sensitive subtasks (e.g., summarisation, routing).

A tiered trust model (public, verified, trusted) enables incremental adoption: non–critical workloads can leverage the full network, while sensitive tasks are constrained to a more controlled subset.

D. Energy Measurement and Telemetry

Accurate energy modelling requires per–node telemetry. In practice, η_{sys} varies by node (residential PSU efficiency, cooling, idle overhead). We propose:

- **Agent–Based Monitoring**: Each node runs a lightweight monitoring agent that logs GPU power draw (via NVML on NVIDIA GPUs), CPU usage, and wall–clock time per task.
- **Calibration Phase**: Nodes periodically run benchmark tasks of known energy cost to calibrate their reported $E(n, A)$ against ground truth.
- **Aggregated Reporting**: The registry collects anonymised energy statistics to refine global models of E and improve the accuracy of the cost function C (Equation 1).

Without real–time measurement, energy–aware routing degrades to heuristic estimation. Deploying such telemetry at scale is an engineering challenge but technically feasible with existing tools.

E. Utilisation and Realistic Capacity

Our illustrative scenario assumes 20% utilisation ($u_g = 0.2$). Achieving this in practice requires:

- Sufficient task diversity to match the capabilities of heterogeneous nodes.
- Incentive mechanisms to encourage nodes to remain online and advertise availability.
- Geographic and temporal load balancing (leveraging time zones and variable electricity pricing).

If average utilisation is lower (e.g., 10%), the energy footprint scales proportionally, but so does the effective capacity per active node. Conversely, higher utilisation (30–40%) would increase total energy but also total useful work. Empirical deployment will reveal realistic utilisation bounds.

F. Additional Challenges

- **Model Evaluation Across Nodes**: Evaluating expert models fairly across diverse hardware and datasets to inform routing decisions.
- **Incentive Mechanisms**: Developing schemes that reward useful compute, discourage malicious behaviour, and internalise energy and carbon costs.

X. Conclusion

We have outlined ColabHive, a distributed hive–mind architecture that uses task decomposition, specialised agents, and energy–aware node selection to construct a global AI capability from heterogeneous hardware. Rather than relying on ever

larger monolithic models in a few data centres, ColabHive advocates for a modular, collaborative, and ecologically conscious approach: using the right model on the right hardware for each task, and reusing existing GPUs and CPUs wherever possible.

We acknowledge significant challenges: ensuring reliability on prosumer hardware, quantifying orchestration overhead, securing sensitive data in untrusted environments, and achieving realistic utilisation rates. However, the potential benefits—reduced marginal energy demand, democratised access, and hardware reuse—justify further investigation. Future work includes deploying small–scale prototypes to measure real–world energy and latency trade–offs, stress–testing fault tolerance mechanisms, and formalising incentive and governance structures.

Our analysis suggests that a well–orchestrated network of modest experts on repurposed hardware can form a scalable and potentially more sustainable alternative to the current trajectory of ever–larger centralised AI infrastructure, provided that the operational and security challenges can be adequately addressed.

REFERENCES

[1] International Energy Agency, "Energy and AI: Energy demand from data centres," 2024.

[2] IEA 4E, "Data Centre Energy Use: Critical Review of Models and Results," 2025.

[3] Ember, "Global Electricity Review 2024: Electricity transition in 2023," 2024.

[4] International Energy Agency, "Electricity 2025: Emissions," 2025.

[5] A. Author et al., "Environmental burden of United States data centers in the age of AI," arXiv:2411.09786, 2024.

[6] BloombergNEF, "Power for AI: Easier said than built," 2025.

[7] NVIDIA, "GeForce RTX 3090 specifications and TDP," 2020.

[8] Puget Systems, "Quad RTX 3090 GPU Wattage Limited 'MaxQ' TensorFlow Performance," 2020.