SpacePilot: A Provenance-First Decision Layer for Single-Owner AI Compute Fleets

MotionVector

Design paper — single-machine core implemented; fleet design not yet evaluated.
Repository: *(link TBD)*

August 2026

Abstract

Model and runtime optimizations are well served by open source; the layer that decides *where a workload should run* is not. Production schedulers assume datacenter physics — nodes that are up or failed, specifications that predict performance, durable cluster state. None of the eleven systems we survey models a suspended consumer machine as a schedulable state. Yet the fastest-growing deployment target for open models is the single-owner fleet: a laptop, a workstation, occasionally rented cloud machines, and managed inference APIs, all owned or paid for by one person or small team.

We present the design of SpacePilot, a decision layer for such fleets, built on three commitments. **Provenance is typed:** every performance number is labeled *flown* (measured on this fleet), *on paper* (cited and dated, external), or *unflown* (no evidence), and the type system refuses to blend them. **Readiness beats specification:** placement weighs which machine already holds the weights in memory, not rated FLOPS, via a cost model that prices cold starts explicitly. **Sleep is a first-class state:** a suspended machine is *asleep, not gone*; work may wait for it, and the interface says so. Fleet state divides into three classes by reconciliation semantics — an append-only measurement log that merges by union, a single-author signed order set, and an observed live picture that is never persisted — which removes the need for consensus in a fleet with one owner. A single-machine core (hardware probing, model registry, measured runs, ranked execution) is operational; multi-machine coordination is a design with a working protocol boundary. We compare against datacenter schedulers, sky brokers, and consumer-device inference systems, and state what a single-owner scope gives up.

1 Introduction

The open-model stack has three layers: models and their formats; inference engines and runtimes; and the orchestration above them. The first two layers are rich with open-source work — weights hubs, quantization formats, and runtimes for every major silicon family. The third layer is where open deployment fails in practice. Developer surveys report that open models lead adoption yet trail closed models in reaching production, and attribute the gap to tooling rather than resources [1]. The missing tooling is not another runtime. It is the decision: *given this workload, this hardware, this budget, and this moment — what runs where?*

That decision has an answer inside every hyperscaler, computed by closed cluster schedulers over private fleet state. It has no answer for the person who owns a laptop and a workstation, rents a GPU box by the hour, and holds API keys to a few providers. We call this a **single-owner fleet**. It differs from a datacenter in physics, not just scale: machines sleep and wake; they move networks; their real capacity (what the GPU driver will actually grant, what is already resident in memory) diverges from their specifications; and the owner personally pays for every rented hour, so a wrong placement is a visible bill.

1.1 Design goals

1. **Honest answers from instruments, not memory.** The system measures the machines it manages and records where every number came from. A guess must be distinguishable from a measurement at

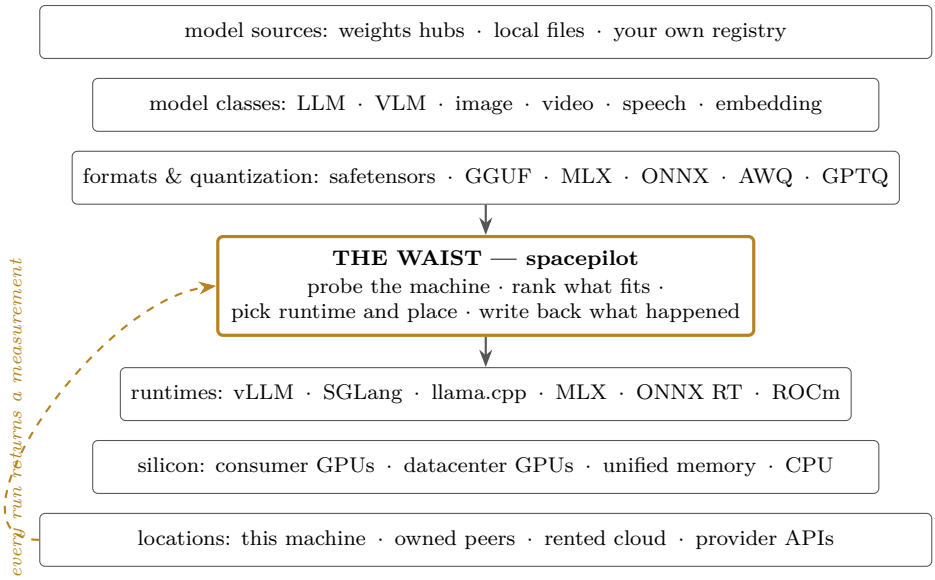

Figure 1: The hourglass. Every wide band has strong open-source players; the waist is the layer this paper describes. The dashed return edge is the part that compounds: each run writes a measured record, so the next decision uses this fleet's numbers rather than a model card's.

the type level.

2. **Decide on readiness, not ratings.** Two identical GPUs are not equal if one already holds the model. The scheduler's inputs are what is loaded, warm, and awake now.

3. **Model the fleet people actually have.** Consumer machines that sleep, one owner, no operations team, money that is personal. No consensus protocols where a single author suffices; no assumption that a quiet machine is a dead machine.

4. **One state, many windows.** CLI, web UI, and agent-facing API are views onto the same live state, never copies. The base capability must work with zero daemon, zero configuration, and zero network.

1.2 The narrow waist

SpacePilot deliberately owns very little. Model sources, formats, quantizations, runtimes, and silicon all have strong open-source ecosystems; composing them is integration, not research. What no open project provides is the waist of the hourglass (Figure 1): probe the machine, pick the model variant that fits, pick the runtime, pick the place — and learn from what happened.

1.3 Contributions

1. A three-class decomposition of fleet state by reconciliation semantics (§3.2) that replaces consensus with union-merge, single-authorship, and deliberate non-persistence.

2. A typed provenance system for performance knowledge — *flown / on paper / unflown* — extended to model capabilities through a caveat taxonomy that records whether a compressed variant still does the asked-for job (§3.3).

3. Sleep as a first-class scheduler state, with a liveness model (*ready / busy / asleep / gone*, each fact aged) and a readiness-aware cost model that prices cold starts (§3.5, §3.6).

4. A protocol boundary (Substrate) under which a daemon-backed fleet and a daemon-less single machine expose identical semantics, verified by a parity test (§3.1).

5. An open implementation of the single-machine core, and an honest account of which parts of the design it exercises (§4).

2 Related work

Datacenter cluster schedulers. Borg [2], Meta's MAST, and Microsoft's Singularity [3] schedule ML workloads over fleets they assume are provisioned, monitored, and always-on; a node that stops responding is failed and drained. These systems solve harder problems than ours at scale (preemption, migration, global placement) and none of ours at the edge: no notion of a machine that intends to return, no per-decision monetary consent, no provenance typing of performance claims. Their fleet state is also private to their operators — the decision layer exists, but only inside.

Sky brokers. SkyPilot [4] and dstack broker workloads across clouds for teams: find capacity, provision, run, tear down. They answer "get me a machine" well. They do not model machines the user already owns as scheduling targets with residency and sleep, and their catalogs describe instance types, not measured behavior of specific machines. SpacePilot treats a broker as one possible dock, not as the decision layer.

Serving orchestration. NVIDIA Dynamo, llm-d, AIBrix [5], and vLLM's sleep mode optimize serving *within* a cluster already chosen — KV-cache management, disaggregated prefill, autoscaling. This layer is trending open on both sides of the Pacific, and SpacePilot composes it rather than competes with it: a runtime's sleep mode is a mechanism; deciding whether the sleeping copy on machine A beats a cold start on cheaper machine B is policy, and it needs fleet-level knowledge these systems do not hold.

Consumer-device inference. exo [7] and Petals [6] aggregate consumer hardware for inference — the closest systems in spirit. Petals pools volunteer GPUs across owners; exo clusters one owner's devices. Neither models readiness or sleep: exo stalls when a member machine suspends and instructs users to disable sleep entirely. Both also treat placement as topology (split the model across what is present) rather than decision (should this run here at all, at what cost, with what evidence).

Benchmark corpora. Public efforts measure model × quantization × runtime × device configurations at scale. SpacePilot consumes such corpora as high-trust *on paper* evidence and never as *flown*; the distinction is the design point, since a public number describes a configuration and a flown number describes *your machine* under *your* contention.

3 Design

The system is one daemon per machine plus a shared vocabulary of state. Each subsection presents a component as the problem it answers.

3.1 Topology: one daemon, many windows

Problem: interfaces multiply, and copies of state drift.

Each machine runs one daemon exposing two doors: a Unix-domain socket for local callers, with filesystem permissions as authentication, and a TCP endpoint bound only to a private mesh network (a tailnet), so peer traffic inherits WireGuard identity and no port faces the internet. Peers authenticate with Ed25519 keys listed in a signed fleet manifest; removing a machine is deleting a line. There is no certificate authority and no coordinator election — a single-owner fleet has an owner.

Every interface — CLI, web cockpit, and the MCP server that agents call — speaks one protocol, **Substrate**, to whichever backend exists: `DaemonClient` over the socket, or `DirectLocal`, the same service functions in-process. The two backends must be indistinguishable to callers; a parity test asserting `DirectLocal` ≡ `DaemonClient` on identical fixtures is the test the architecture stands on. The consequence is a property we consider non-negotiable: `spacepilot check` answers honestly on a fresh machine with no daemon, no configuration, and no network.

3.2 State: three classes by reconciliation semantics

Problem: distributed state is hard, but only if all state is the same kind.

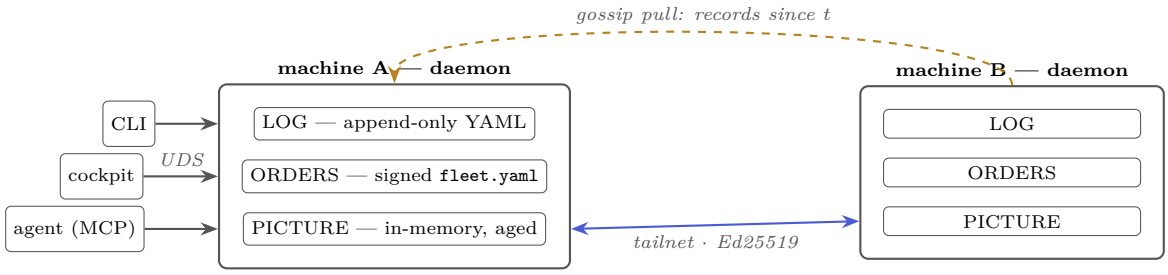

Figure 2: Topology. Interfaces are windows onto one daemon; daemons are peers over a private mesh. The three state boxes reconcile by different rules (§3.2): the LOG merges by union, ORDERS has one author, the PICTURE is observed and never merged.

Everything a fleet must share falls into three classes, each with a reconciliation rule simple enough to remove the usual machinery (Table 1).

Class	Contents	Author	Reconciliation	Persistence
LOG	measurements: what ran, where, how fast	every machine, append-only	union — immutable facts cannot conflict	YAML source of truth; derived SQLite index
ORDERS	fleet membership, keys, declared rates	exactly one (the helm)	versioned overwrite, Ed25519-signed	one signed file
PICTURE	liveness, load, what is warm now	each machine about itself	observed, never merged	none — memory only, every field aged

Table 1: Three state classes. The design refuses to let one class leak into another's semantics.

Measurements never mutate, so the LOG needs no CRDT and no consensus — gossip is a pull of records-since-a-timestamp, and any subset of machines converges by union. ORDERS has one author, so conflict is impossible by construction. The PICTURE is deliberately never stored: a persisted liveness fact is a stale liveness fact, and a test walks the data directory to prove no picture field ever reaches disk. Freshness is carried in-band — every picture field is aged (`ready · checked 8s ago`), and consumers render the age rather than trusting the value.

3.3 Provenance: flown, on paper, unflown

Problem: a guess that looks like a measurement poisons every decision downstream.

Every performance number carries one of three provenance types. **Flown**: measured on this fleet, stamped with the machine fingerprint, the exact resolved weight revision, and the contention conditions at measurement time. **On paper**: an external claim with source and date — a model card, a public benchmark corpus, a provider's rate sheet. **Unflown**: no evidence; rendered as the word *unflown*, never as a number. The rendering rule matters as much as the storage rule: a table cell shows a flown number with its sample count, a cited number with its date, or the admission that nothing is known. Blending is a type error.

Provenance extends from speed to *capability* through **caveats**. Compression changes what a model can do, not only how fast: a distilled speech model may lose word-level timestamps; a quantized image model may lose legible text rendering. A caveat is a structured record — capability slug, status (*preserved / degraded / untrained / absent*), detail, and its own provenance (*measured / inferred / declared*) — attached to a model variant in the registry. The confirmation prompt shows any degraded caveat *before* asking to run, because a model that fits, is licensed, and is fast — but silently cannot do the asked-for job — is precisely the failure this system exists to catch. Absence of caveats renders blank, not "none": absence of evidence is not evidence of preservation.

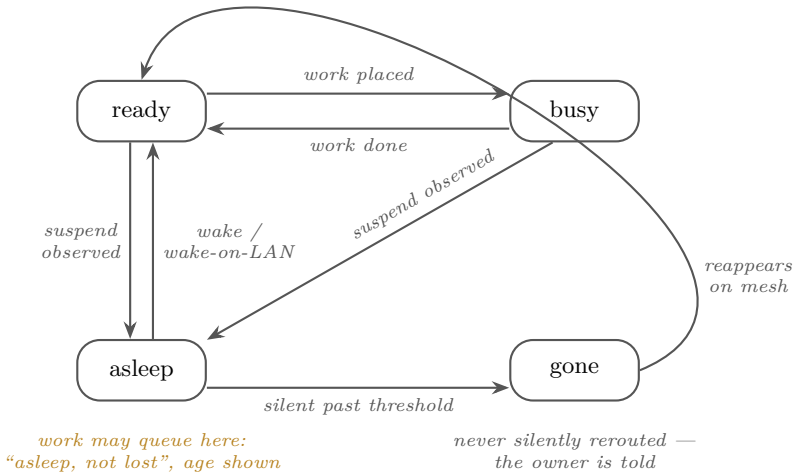

Figure 3: Liveness states. Every state is rendered with the age of the observation that produced it.

3.4 Weight resolution: one resolver, format as data

Problem: "works on my machine" enters through hardcoded weight paths.

A fleet that schedules a job onto a machine that cannot find its own weights is not a fleet. SpacePilot has exactly one resolver: resolve(*repo*, *revision*, *patterns*) → *files*, backed by the shared content-addressed cache other tools on the machine already use, consulted without network access. The resolver never inspects model format; format lives in the registry as a variant axis, each variant naming its repository and file patterns. This is what makes resolution behave identically for PyTorch, ONNX, and GGUF. Drivers hold no search lists. The resolver also returns the resolved revision, which is stamped into every flown measurement — a measurement records exactly which bytes produced the number.

3.5 Liveness: sleep is a state, not a failure

Problem: every surveyed scheduler treats a suspended machine as absent.

Each daemon reports its own liveness; peers observe *ready / busy / asleep / gone*, with ages (Figure 3). The distinction between *asleep* and *gone* changes scheduling: work targeted at an asleep machine can wait, and the interface says exactly that — queued, waiting for a machine that is asleep, not lost. Wake is attempted where the mechanism exists (wake-on-LAN for owned machines, a start call for rented ones) and honestly waited for where it does not. Daemons register with OS supervision so they restart after wake — the exact moment liveness must re-announce.

3.6 Scheduling: readiness priced, money loud

Problem: the cheapest machine on paper is often the most expensive machine in fact.

Residency is the scarce good. A candidate that must load weights before answering is not equal to one already holding them, and the cost model says so directly:

$$\text{score} = \text{price} + \big(\text{cold ? load\_seconds} \times \text{value\_of\_latency} : 0\big)$$

where load_seconds comes only from flown records of past loads on that machine — an unflown load time renders as unknown, never as a guessed constant — and value_of_latency is the owner's stated exchange rate between waiting and paying. The output is not a placement but a ranked table: each feasible option

labeled with its verdict (fits / tight / won't fit), its provenance-typed speed, its caveats, and its cost — estimated before, actual after, clock visible while running. Renting is always a decision presented to the owner, never a default. The wedge interaction is one command: `spacepilot run <workload>` prints the ranked honest table for the machine in front of you and asks `Run? [Y/n]`; the fleet version generalizes the same verb rather than introducing a new one.

3.7 The agent surface

Problem: the fastest-growing caller cannot read a dashboard.

Increasing compute demand originates from agents, which need machine-readable honesty more than humans do — an agent cannot squint at a stale dashboard and compensate. Every interface property above (typed provenance, aged facts, explicit verdicts, loud money) is exposed unchanged through MCP tools, so an agent can ask what fits, what it costs, and how the system knows, and can be granted bounded autonomy over placement with the same consent gates a human sees.

4 Implementation status

The single-machine core is implemented and in daily use: hardware probing (silicon, memory ceiling as the driver reports it, with the source of any limit named), a model registry with per-variant files and licenses, measured runs recorded with fingerprint, revision, and contention (⟨*models: TBD*⟩, ⟨*variants: TBD*⟩, ⟨*recorded measurements: TBD*⟩), and speech generation running locally at ⟨× *realtime: TBD*⟩. The provenance join, ranked `run` table, caveat surfacing, and unified resolver are the active workstream. The daemon, Substrate parity, fleet gossip, liveness, wake, and the scheduler exist as the design of §3 with settled protocols and test criteria — designed, not evaluated. We state this because the paper's own rule demands it: these sections are *on paper*, and will be *flown* or revised.

5 Limitations

Single-owner scope. The three-class state model gets its simplicity from one owner signing ORDERS. Multi-tenant fairness, adversarial peers, and cross-owner trust are out of scope; Petals-style volunteer pools need machinery we deliberately avoid.

Placement, not parallelism. SpacePilot places whole workloads. It does not split a model across machines (exo's problem) and does not manage intra-cluster serving (Dynamo's problem); it decides *whether and where*, then delegates.

Evidence is only as good as the fleet's history. A fresh fleet has no flown records; early decisions lean on *on paper* evidence and say so. The compounding value of the LOG is a claim about time, unproven at time of writing.

Unverified design surface. The parity property, gossip convergence in practice, and wake reliability across consumer hardware are asserted by design and test plan, not yet by measurement.

6 Future work

Measured evaluation of the fleet path (two-machine mesh, sleep/wake cycles under real workloads); importing public benchmark corpora as *on paper* evidence with automatic promotion to *flown* when local runs confirm them; a curator that emits *inferred* caveats for new compressed variants, promoted or refuted by fleet measurements; provider-lane semantics (per-token rates are declared truth, not measurable residency — different verdict semantics, stated rather than faked); and training and fine-tuning as extensions of the same waist.

7 Conclusion

Personal fleets are not small datacenters; they are a different physics with an empty tooling niche. The design here bets that the missing layer is mostly discipline: type the provenance of every number, price readiness instead of trusting ratings, treat sleep as intention rather than failure, and keep exactly one source of truth per class of state. The parts that are built follow the discipline; the parts that are not are labeled. That labeling is not humility as decoration — it is the product working on its own paper.

References

[1] Mozilla & SlashData. *State of Open Source AI*. July 2026. n=1,410.

[2] A. Verma et al. *Large-scale cluster management at Google with Borg*. EuroSys 2015.

[3] D. Shukla et al. *Singularity: Planet-Scale, Preemptive and Elastic Scheduling of AI Workloads*. arXiv:2202.07848, 2022.

[4] Z. Yang et al. *SkyPilot: An Intercloud Broker for Sky Computing*. NSDI 2023.

[5] The AIBrix Team. *AIBrix: Towards Scalable, Cost-Effective Large Language Model Inference Infrastructure*. arXiv:2504.03648, 2025.

[6] A. Borzunov et al. *Petals: Collaborative Inference and Fine-tuning of Large Models*. ACL 2023 (demo).

[7] exo-explore/exo. GitHub repository. *(sleep behavior citation TBD)*.

[8] Meta MAST; NVIDIA Dynamo; llm-d; vLLM sleep mode; Gödel; Volcano; dstack. *(citations TBD)*.