The AI Economics of Scale

How Autonomous Production Collapses While Verification, Observability, and Trust Do Not — Evidence from a Single-Builder, 52-Million-Record Natural Experiment

Ayo Ashiru
Independent researcher · Working paper · July 2026

Abstract

Classical economies of scale predict that unit cost falls as output rises. The arrival of capable, agentic artificial intelligence appears to deliver an extreme case: a single person, directing AI agents, can now produce software artifacts at a scale that formerly required a mid-sized engineering organisation. We ask what, precisely, scales — and what does not. Drawing on a fully instrumented, single-builder production system (ARQERA: 2,688 API endpoints, 805 services, ~653,700 lines of backend code, and a hash-chained evidence substrate of **52.1 million records** accumulated over 174 days, built and operated by one engineer directing AI agents), we present a decomposition of the cost of delivering *governed AI value* into four components — production, verification, observability, and trust — and argue, both formally and empirically, that **AI capability collapses the production term while leaving the other three approximately invariant.** We formalise the consequence as an Amdahl-style bound: the maximum achievable "AI economy of scale" is bounded above by the reciprocal of the non-production cost fraction, regardless of how capable the underlying models become. We provide direct measurements of each non-collapsing term: verification does not collapse (**37.6%** of autonomous agent runs required human escalation; in a larger historical sample, over 80% escalated); observability does not collapse and in fact *grows adversarially with production volume* (nine distinct self-observability defects surfaced in a single operating session; a family of production failures traced to the system's inability to read its own 52-million-record substrate, remediated for **19×–300×** latency reductions); and trust does not collapse (a single expired platform credential silently disabled inference for every tenant while their own valid credentials sat unused). We report a management result — the fix-to-feature commit ratio was driven from a historical **2.1 : 1** to **0.88 : 1** across 4,742 commits by converting engineering discipline into automated gates — as evidence that the non-production curves are partially *bendable* through structural investment. We then supply the constructive half: a **reference architecture for scaling AI**, drawn from and measured on the same system, of six composable structures that each lower a non-production cost — work reified as governed, criteria-bearing packets routed to capacity ("Uber for work"); north-star adjudication that certifies completion against evidence rather than self-declaration; an event-ledger-and-projection read model that holds observability cost (and read latency) constant as state grows without bound (a ≈760× read-model compression; a `COUNT` from 44.75 s to 7 ms); a tiered capability ladder with a self-hosted floor that held blended inference cost **one to two orders of magnitude below a uniform-frontier deployment**, with ~14% of calls at zero marginal cost; and graded, evidence-backed governance (a 9.4-million-row action ledger) that scales safety through ex-post accountability rather than universal human approval. We argue this architecture — not larger models — is the operative answer to how AI scales across cost, latency, and safety, and that once the capability bound is near, the returns to building it dominate the returns to capability. We are explicit throughout about the methodological limits of a single-case natural experiment and treat them as first-order.

Keywords: economies of scale; AI agents; large language models; software economics; verification; observability; transaction costs; Amdahl's law; scaling laws; governance.

---

1. Introduction

1.1 The puzzle

Two literatures make confident, opposing-feeling predictions about scale. The economics of production (Marshall, 1890; Chandler, 1990) holds that as output rises, fixed costs amortise and unit costs fall — economies of scale — until coordination and complexity impose diseconomies. The economics of software has long been the awkward exception: Brooks's Law (Brooks, 1975) famously observes that adding engineers to a late project makes it *later*, because communication overhead grows combinatorially with team size; software has historically exhibited *dis*economies of scale in production. Meanwhile, the empirical scaling laws of deep learning (Kaplan et al., 2020; Hoffmann et al., 2022) describe a different curve entirely — smooth, predictable improvements in model *capability* with compute, data, and parameters.

Capable agentic AI collides these curves. A single individual directing AI agents can now generate software output at volumes that formerly demanded a large team — apparently escaping Brooks's Law by removing the human-communication term, and apparently importing the smooth capability curve of the models into the economics of *production*. The natural inference is a new and extreme economy of scale: near-zero marginal cost of software production, bounded only by model capability, which itself keeps improving.

We argue this inference is half right and dangerously incomplete. It is right about *production*. It is silent about the three cost curves that, in a governed, deployed, trusted system, do not collapse — and which therefore come to dominate. **The interesting economics of AI at scale is not the collapse of production cost; it is the non-collapse of everything else.**

1.2 The gap

The scaling-law literature measures the cost of *training and running models*, not the cost of *delivering governed value from them*. The software-economics literature predates agentic AI and models human teams, not human-plus-agent production. The transaction-cost economics of the firm (Coase, 1937; Williamson, 1985) provides the right *conceptual* apparatus — the cost of coordinating and verifying production is distinct from the cost of production itself — but has not been applied to the human-plus-agent regime with instrumented data. There is, to our knowledge, no empirical decomposition of the cost of AI-produced, deployed, governed value into its production and non-production components, measured on a real system at scale. This paper offers one.

1.3 Contributions

1. **A cost decomposition** (§3) of delivering governed AI value into four terms — production, verification, observability, trust — and a formal statement of the *collapse asymmetry*: capability reduces the first and approximately preserves the other three.
2. **An Amdahl-style bound** (§3.3): the maximum AI economy of scale is bounded above by the reciprocal of the non-production cost fraction, independent of model capability. We derive it and discuss its estimation.
3. **A single-builder natural experiment** (§4): ARQERA, a 52.1-million-record production platform built and operated by one engineer directing AI agents, with an unusual property — the operator is, in part, itself an AI

agent — allowing a *reflexive* test of whether the non-production costs bind AI operators as well as human ones.

4. **Direct measurements** (§5–§6) of the production collapse and of each non-collapsing term, including the escalation rate, the observability-defect census, the performance-remediation factors, and the trust-failure case.
5. **A management result and its strategic reading** (§7): the non-production curves are partially bendable by converting discipline into automated structure (gates, projections, adjudication), evidenced by a fix-to-feature ratio driven from 2.1 : 1 to 0.88 : 1.
6. **A reference architecture for scalable AI** (§8–§18): six composable, measured structures — governed work packets and dispatch ("Uber for work"), north-star completion adjudication, the event-ledger/projection read model, tiered routing with a self-hosted floor, and graded evidence-backed governance — that together lower (1-*p*) across cost, latency, and safety, constituting the constructive answer to how AI scales.

1.4 Roadmap

The paper is in three parts. **Part I (§2–§7) — the bound.** §2 situates the work; §3 develops the framework and the Amdahl bound; §4 states the method and its limits; §5 reports the production collapse; §6 the three non-collapsing costs; §7 estimates the bound and reports what worked and what did not. **Part II (§8–§18) — the architecture.** §8 bridges from the bound to the engineering; §9–§13 give five structures that lower the non-production costs (work packets and dispatch; completion adjudication; the projection read model; tiered routing; graded governance); §14 adds the evidence flywheel and §15 the self-hosted-inference economics; §16 reports the negative results (what did not scale); §17 reconciles the architecture to the enterprise's own language of cost, latency, and safety; and §18 collects the seven principles into a reference architecture. **Part III (§19–§21) — the implications.** §19 discusses implications for firms, providers, and the theory of the firm; §20 is a full threats-to-validity treatment, load-bearing given the single-case design; §21 concludes.

---

2. Background and related work

Economies of scale and the firm. Marshall (1890) and Chandler (1990) establish scale economies in physical production; Coase (1937) and Williamson (1985) reframe the firm as a device for economising on *transaction and coordination costs*, distinguishing the cost of doing a thing from the cost of organising and verifying its being done. This distinction is the seed of our decomposition: AI attacks the "doing" cost, not the "organising and verifying" cost.

Software economics. Brooks (1975) identifies communication overhead as the source of software's dis-economies of scale; Boehm's COCOMO (1981) formalises effort as a super-linear function of size. Both model *human* teams. The human-plus-agent regime is novel precisely because it appears to sever the communication term — one human's intent, many agents' execution, no inter-agent politics — while, we will argue, leaving the *verification* term (did the agents do the right thing?) not merely intact but *enlarged*.

Scaling laws for AI. Kaplan et al. (2020) and Hoffmann et al. (2022, "Chinchilla") establish smooth power-law relationships between model loss and compute/data/parameters. These describe *capability supply*. They say nothing about the cost of converting capability into governed, deployed value — the demand-side economics this paper addresses.

Verification, generation, and judgment. A deep intuition from complexity theory is that verifying a solution is easier than finding one (the P-vs-NP asymmetry). A central empirical finding of this paper is that, *operationally*, the asymmetry can invert: when a system can generate work almost for free, verifying that the generated work is correct, complete, and safe becomes the binding cost. The LLM-as-judge literature (Zheng et al., 2023) attempts to automate verification with models themselves, but inherits the very self-grading problem we document — the generator and the judge are the same class of system.

Agent systems. Reasoning-and-acting agent architectures (Yao et al., 2023, "ReAct"; and the tool-use and autonomous-agent lineage) demonstrate capable execution; they do not solve the *finishing* problem — reliable completion against a specification adjudicated by a party other than the agent — which we treat as a first-class cost.

Autonomic computing and observability. IBM's autonomic-computing vision (Kephart & Chess, 2003) and the site-reliability literature (Beyer et al., 2016) presuppose that systems can be monitored; we find that the *ability of a system to answer questions about itself* — its wieldability — is itself a cost that does not fall with production capability and can rise against it.

Principal-agent theory. Jensen and Meckling (1976) formalise the cost of aligning an agent's action with a principal's interest. In the AI setting this reappears twice: between the human principal and the AI agent (the verification cost), and between the platform and the customer whose credentials and value it holds (the trust cost). The "agency cost" Jensen and Meckling attribute to monitoring and bonding is, in our decomposition, exactly the sum of the verification and trust terms — a lineage worth making explicit, because it locates our finding within a fifty-year-old theory rather than presenting it as new: what is new is that the *production* term, which agency theory took as given, has now collapsed, throwing the agency costs into relief.

The economics of AI. The most direct antecedent is Agrawal, Gans, and Goldfarb (2018), who frame AI as a drop in the cost of *prediction* and argue that the economic complements to cheap prediction — *judgment*, *data*, and *action* — rise in relative value. Our contribution can be read as an empirical extension and sharpening of their thesis: we identify verification, observability, and trust as the specific, measurable complements that do not fall, and we supply magnitudes (a 37.6% judgment/escalation load; a 52-million-record observability burden) where their account is qualitative. Brynjolfsson and McAfee (2014) and the "productivity paradox" literature (Brynjolfsson, Rock, & Syverson, 2021) document a recurring lag between a general-purpose technology's capability and its measured productivity gains, attributing it to the *complementary intangible investments* — process redesign, new skills — that capability alone does not supply. Our four non-production costs are a concrete instance of exactly such complementary investments, and the Amdahl bound (§3.3) is a formal reason the paradox is not a lag but a *ceiling* absent that investment. Acemoglu and Restrepo (2019) model automation at the level of *tasks*, distinguishing tasks that automate from tasks that are created or reinstated; our decomposition is a task-level claim about a single production pipeline — the production tasks automate, the verification/observability/trust tasks do not, and may be *reinstated and enlarged* by the automation of the first (§3.4).

Amdahl's law. Amdahl (1967) bounds the speedup of a computation by its non-parallelisable fraction: if fraction s is serial, speedup is at most $1/s$ regardless of processor count. We import the form directly: if fraction $(1-p)$ of value-delivery cost does not collapse under AI, the speedup is at most $1/(1-p)$, regardless of model capability. This is, to our knowledge, a novel application of Amdahl's structure to the economics of AI-produced value, and it connects the two literatures above: it is the formal object the productivity-paradox account gestures at (a ceiling, not a lag) and the quantitative complement to the prediction-machines account (a bound on how much cheap prediction alone can deliver).

3. Theoretical framework

3.1 The four-term decomposition

Let the total cost of delivering one unit of **governed AI value** — a completed, deployed, trustworthy piece of work — decompose as

$$C = C_p + C_\mathrm{v} + C_\mathrm{o} + C_\mathrm{t}$$

where

- C_p **(production)** is the cost of generating the artifact: writing the code, drafting the document, taking the action.
- C_v **(verification)** is the cost of establishing that the produced artifact is correct, complete, and satisfies its specification — the finishing cost.
- C_o **(observability)** is the cost of knowing the state of the system: what is deployed, what it did, why it failed — the wieldability cost.
- C_t **(trust)** is the cost of establishing and maintaining the authority, custody, and provenance under which the work is permitted and can be audited — the governance cost.

The decomposition is a claim, not a definition: we assert these four are the exhaustive, first-order components of *governed, deployed* value (as opposed to raw generation), and that they are conceptually separable. §5–§6 measure each.

3.2 The collapse asymmetry

Let $k \geq 1$ denote the **production collapse factor** — the multiple by which capable AI reduces C_p relative to the pre-AI human baseline. The empirical scaling laws (Kaplan; Hoffmann) imply k grows with model capability, which grows with compute; over the observation window k is large (§5).

Our central theoretical claim is the **collapse asymmetry**:

> **Proposition 1 (Asymmetric collapse).** Capable AI reduces C_p by a large factor k, while C_v, C_o, and C_t are, to first order, invariant to k. Formally, $\partial\, C_p/\partial\, k < 0$ with $|\partial\, C_p/\partial\, k|$ large, while $\partial\, C_\mathrm{v}/\partial\, k \approx \partial\, C_\mathrm{o}/\partial\, k \approx \partial\, C_\mathrm{t}/\partial\, k \approx 0$.

The intuition for each invariance, developed and measured later:

- C_v **does not collapse** because the party best able to verify AI work at scale is *another AI*, which shares the generator's failure modes (self-grading), so verification either falls back to a human (costly, and measured at 37.6% of runs, §6.1) or to an adjudicator that must itself be built and trusted.
- C_o **does not collapse, and grows adversarially with** k, because more production means more state to observe, while the ability to observe is a *separate surface* that AI production does not automatically create (§6.2). A system can generate 64,000 records a day and be unable to answer "how many are there?" efficiently.

- **C_t does not collapse** because trust is a property of *relationships and custody*, not of generation: a platform can produce infinite intelligence and still break every customer with one expired shared credential (§6.3).

3.3 The Amdahl bound for AI economies of scale

Let $p = C_p / C$ be the production fraction of value-delivery cost at the pre-AI baseline. Under Proposition 1, AI scales $C_p \to C_p/k$ while the other terms are held. The **speedup** in delivering governed value is

$$S(k) = \frac{C}{C'(k)} = (1)/((1-p) + \frac{p}{k}).$$

Taking the limit of unbounded capability,

$$\boxed{\lim_{k\to\infty} S(k) = \frac{1}{1-p}.}$$

> **Proposition 2 (Amdahl bound).** The maximum achievable AI economy of scale in delivering governed value is bounded above by the reciprocal of the non-production cost fraction, $1/(1-p)$, independent of model capability k.

The parallel to Amdahl (1967) is exact: there, the serial fraction bounds parallel speedup; here, the non-collapsing (verification + observability + trust) fraction bounds AI speedup. The strategic corollaries are immediate and, we argue, under-appreciated:

> **Corollary 2.1.** Beyond a point, investing in model capability (k) yields vanishing returns in delivered value; the binding investment is in *lowering (1-p)* — i.e., in reducing verification, observability, and trust costs structurally.
> **Corollary 2.2.** Two firms with identical model access can differ in delivered-value throughput by the ratio of their non-production cost fractions. Competitive advantage in the AI era accrues to whoever has the smaller $(1-p)$.

We do not claim $(1-p)$ is a constant; §7 reports that it is partially *bendable* by structural investment, which is the actionable heart of the paper. But it does not fall for free with k, and that is the whole point.

3.4 AI diseconomies of scale

Proposition 1's invariance clause is, for observability, an understatement, and the understatement deserves its own formal treatment because it names a genuine *dis*economy — the AI-era analogue of the classical diseconomies that eventually bend the average-cost curve back up.

Classical economies of scale reach a limit when coordination and complexity costs rise with output, turning the falling average-cost curve upward. We claim an analogous but sharper effect for observability. Let V denote cumulative production volume — the size of the state the system has generated (in our case, the substrate, growing at ~64,000 records/day toward 52 million). The cost of *observing* that state on the request path, absent a maintained projection, is not invariant to V but *increasing* in it: a scan of the substrate costs $O(V)$, so the per-query observability cost rises linearly with everything the system has ever produced.

> **Proposition 3 (Observability diseconomy).** When observation is performed by traversal of produced state, the marginal observability cost is increasing in cumulative production volume: $\partial C_o / \partial V > 0$. Because AI raises the production rate $\partial V/\partial t$ by the collapse factor k, it raises the *rate of accumulation* of observability cost by the same factor. Observability cost thus scales *with* the very production AI accelerates — an anti-economy of scale.

This is not a corner case; it is the mechanism behind the "everything is offline" failure family (§6.2), in which a system producing 64,000 records a day became progressively unable to read itself, the `OFFSET`-scan cost rising with the 11-million-row table it scanned. The remediation (§6.2, §7.2) is precisely to break the $O(V)$ dependence — to make observation cost $O(\log V)$ or $O(1)$ via indexes, projections, and estimates — which converts the diseconomy back into an invariance. The strategic content is stark: **left unaddressed, AI does not merely fail to lower observability cost; it actively raises it in proportion to how much it produces.** A firm that scales AI production without scaling its observability *architecture* buys itself an accelerating liability, not a dormant one. Proposition 3 is thus the formal warning behind Corollary 2.1: the non-production costs are not merely stubborn; one of them is actively hostile to production scale unless structurally defused.

3.5 Interactions and boundary conditions

Two refinements keep the framework honest: the terms interact, and the framework has a domain of applicability outside which it does not hold.

Interaction: production collapse *raises* verification load. The four terms are not independent. Lowering C_p does not merely change the *relative* weight of C_v (the mechanical effect that raises $1\text{-}p$); it can raise C_v in *absolute* terms, because cheaper production means *more* artifacts to verify per unit time. If an agent produces ten times the code, a fixed per-artifact verification cost yields ten times the verification work. Formally, if $C_v = c_v \cdot N$ where N is the number of produced artifacts and N scales with the production rate (which scales with k), then $\partial C_v / \partial k > 0$: **the very collapse that motivates the technology enlarges the term that bounds it.** This is the absolute-terms companion to Proposition 3's observability diseconomy, and it explains an otherwise puzzling field observation — that faster production did not feel like relief but like a rising tide of things to check. The relief is real only to the extent verification is *also* made cheaper structurally (§6.1's adjudication), which is why the two must be co-invested.

Boundary conditions: where the framework does not apply. A theory that applies everywhere explains nothing; we state where ours does not.

1. *Formally-verifiable domains.* Where correctness is machine-checkable at low cost — a compiler's type system, a proof assistant, a domain with cheap exhaustive tests — C_v *does* collapse toward zero, because verification is itself production-like and automatable. In such domains the framework degenerates: the bound approaches $1/(1 - p_o - p_t)$ with the verification term removed, and AI economies of scale can be large. The framework's bite is proportional to how *non-mechanical* verification is — which, for open-ended operational work (did the agent do the *right* thing, not merely a *valid* thing), is most of it.
2. *Low-trust-requirement domains.* Where the work carries no custody, authority, or audit obligation — a throwaway script, a personal draft — $C_t \to 0$ and the trust term drops. The framework is a theory of *governed* value; ungoverned value scales more freely (and is worth less).
3. *Low-state domains.* Where the system holds little cumulative state, Proposition 3's observability diseconomy is inert (V is small), and C_o is a modest fixed cost. The diseconomy bites in proportion to accumulated state

— which, for any system that logs, remembers, or audits, grows without bound.

These boundaries are not escape hatches; they sharpen the claim. The framework predicts that AI economies of scale will be *largest* exactly where verification is mechanical, trust is absent, and state is small — and *smallest*, bounded, exactly where work is open-ended, governed, and stateful. Enterprise operational AI — the domain of the case study, and of most enterprise AI ambition — sits squarely in the second regime. That is precisely why the enterprise AI economy of scale is the hard one, and the one worth theorising.

---

4. Methodology

4.1 Research design

This is an **instrumental single-case study** (Yin, 2018) exploiting a **natural experiment**. The case, ARQERA, is a production AI-operations platform with three properties that make it unusually suited to isolating the collapse asymmetry:

1. **Single builder.** It was designed, built, and operated by one engineer directing AI agents. This removes the inter-human coordination term that confounds Brooks's-Law effects, so the observed output-per-human is a comparatively clean estimate of the production collapse factor *k*.
2. **Full instrumentation.** The platform emits a hash-chained evidence record of its own operations; the substrate, the version-control history, the run outcomes, and the live database are all directly measurable. Every figure in §5–§6 is measured, not estimated, and stated with its source (Appendix A).
3. **A reflexive operator.** The platform is operated, in part, by AI agents — including the autonomous engineering session during which much of this paper's data was gathered. This allows a *reflexive* test: if the non-production costs bind even an AI operator, they are a property of the system, not of human limitation.

4.2 Data sources and measurement

- **Production scale**: source-tree census (files, endpoints, service modules, lines of code) and version-control history (commit counts, message-prefix classification).
- **Substrate**: direct aggregate queries against the live production database, executed with the system-level access required to bypass row-level security (a subtlety that itself became a finding — see §6.2 and §20).
- **Verification cost**: the recorded status distribution of autonomous agent runs (a durable, per-run outcome table), complemented by historical ledger forensics.
- **Observability cost**: a census of the workarounds an operating intelligence was forced into during a single session, and before/after latency measurements of the remediations.
- **Trust cost**: a traced production incident and the configuration counts (bring-your-own-key provider records).

All figures are measured on the live system on 23 July 2026. Latency figures are single-run measurements demonstrating a mechanism, not population averages (§20). Where a number is illustrative rather than measured — notably the fraction *p* in §7 — we say so explicitly.

4.3 On generalising from n = 1

We state plainly what a single case can and cannot support. It can establish an *existence proof* (the asymmetry occurs), *measure mechanisms* (why each term does or does not collapse), and *ground a theoretical framework* in

real magnitudes. It cannot establish a *population distribution* of p or k, nor rule out that ARQERA is idiosyncratic. §20 treats this as a first-order limitation rather than a closing caveat, and §21 proposes the multi-case work that would test external validity. The contribution is the framework, the bound, and the measured mechanisms — not a claim that ARQERA's specific numbers generalise.

---

5. Results I — the production collapse

5.1 The magnitude of output

Directed by one engineer, AI agents produced a system whose surface area matches a mid-sized software organisation (Table 1).

Table 1 — Production output (measured 2026-07-23).

Dimension	Value
API endpoints	2,688
API routers	402
Service modules	805
Data models	170
Backend Python (lines)	653,701
Frontend modules	1,440
Total commits	4,742
Build window	2025-12-20 → 2026-07-23 (215 days)
Evidence records accumulated	52,128,738 (all tables)
— of which, the primary evidence ledger	11,050,873 rows / 17 GB
Distinct recorded action types	252
Substrate accumulation rate	~64,000 records/day (evidence ledger)

Industry effort models (Boehm, 1981) would place a 650,000-line system at tens of person-years of effort. Delivered by one person directing agents in 215 days, the implied production collapse factor k is at least one, plausibly two, orders of magnitude. We resist a precise figure because effort-model baselines are themselves contested; the qualitative claim — *production cost collapsed by a large factor* — is robust and is all Proposition 1 requires.

5.2 Escaping Brooks's Law — and importing a new cost

The single-builder design is not incidental; it is the mechanism. Brooks's Law attributes software dis-economies to inter-human communication, which grows as $O(n^2)$ in team size n. Directing AI agents removes that term:

there is one locus of intent and many executors that do not negotiate with each other. This is the clean form of the production collapse — output scaled without the coordination penalty that bounds human teams.

But the coordination cost did not vanish; it *changed identity*. What a human team spends on communicating *what to build* and reviewing *whether it was built right*, the single-builder-plus-agents regime must still spend on the second half — reviewing whether the agents built the right thing. The communication cost fell to near zero; the **verification cost did not** (§6.1). Brooks's Law was not repealed so much as *re-localised*: from between humans to between the human and the agents' output.

5.3 The discipline result (foreshadowing §7)

A codebase produced this fast should, by the standard intuition, drown in rework. The fix-to-feature commit ratio — commits prefixed `fix` versus `feat` — measures this: a ratio above one means more repair than construction. An early window of this project recorded a ratio of **2.1 : 1**, an explicit signal that rework was dominating. Across the full 4,742 commits the ratio is **0.88 : 1** (1,842 features, 1,616 fixes): *more construction than repair*. §7 argues this recovery is the empirical signature of *bending the non-production cost curves down* through structural investment, and is the paper's central management finding.

5.4 Velocity without a collapse in quality

The production collapse is a *rate* claim, and rate without maintained quality is worthless — a system that ships fast and breaks faster has not scaled, it has borrowed. Two observations argue the velocity was real rather than borrowed. First, the fix-to-feature recovery (§5.3) is itself a quality-at-velocity signal: the ratio measures whether new construction is generating proportionate new repair, and a sub-unity ratio *sustained across nearly five thousand commits* indicates the velocity did not come at the cost of a growing repair backlog. Second, the *deploy* cadence — the rate at which produced code actually reached production behind the full gate suite — was high and safe simultaneously: in a single working session, sixteen distinct backend versions were built, gate-checked, and deployed (a version series we label m147 through m162), alongside dozens of commits, with each deploy passing seven fail-closed gates before it was permitted to land. This is the empirically interesting combination: *high production rate and high deploy rate and maintained repair ratio, together.* The standard trade-off — move fast and break things, or move slowly and safely — was not resolved by choosing a point on it but by *changing the curve*, through the gates that made fast deploys safe (§7.2). The velocity, in other words, was purchased not from quality but from *structure* — which is the same mechanism that bends the non-production costs, seen from the production side.

We are careful not to over-read the deploy cadence: sixteen deploys in a session demonstrates that the pipeline *permits* safe high-frequency deployment, not that every session sustains it. But it establishes the possibility, and it is a direct measurement of the production-plus-verification pipeline operating at rate — the two terms co-invested, exactly as §3.5's interaction argument requires them to be.

---

6. Results II — the three costs that did not collapse

6.1 Verification does not collapse

If production is near-free, the binding question becomes: *is the produced work correct and complete?* We measure this directly. The platform records every autonomous agent run as a durable, status-carrying row. Across

237 runs (Table 2):

Table 2 — Autonomous run outcomes.

Outcome	Count	Share
Self-completed (succeeded)	102	43.0%
Escalated to a human	**89**	**37.6%**
Failed	44	18.6%
Blocked	2	0.8%

More than **one run in three required a human** — the system correctly declining to certify a completion it could not prove. This is the verification cost made visible: the fraction of AI production that a human (or a purpose-built adjudicator) must still touch. It does not fall with model capability, because the failure it guards against — an agent confidently declaring incomplete work done — is a property of self-grading, not of capability.

Historical forensics on a larger sample of the same platform sharpen the picture and reveal a *second-order* pathology. In a ledger of tens of millions of governed actions, the escalation rate exceeded 80%; of escalations that reached a human, on the order of 98% were ultimately *rejected*; and a large fraction of actions were *mis-tiered* — assigned a governance severity that did not match their true risk, so that low-risk actions were escalated and the escalation channel was flooded. The reading is instructive: the verification cost is not merely high, it is *mis-calibrated*. A naive system pays C_v twice — once for the genuine verification need, and once more for the false positives its mis-calibration generates. This distinguishes two components of the verification term: an *irreducible* part (work that genuinely needs a human) and a *waste* part (work escalated by poor calibration). The irreducible part is bounded below by the domain's true ambiguity; the waste part is an engineering defect, and — crucially for the strategic argument — it is *bendable* (§7.2). A system that governs by verb-tier heuristics rather than grounded risk assessment manufactures its own verification cost; the forensic finding that a single class of ungrounded severity mappings drove most of the mis-tiering is direct evidence that much of the observed C_v was self-inflicted and therefore removable — which is the optimistic corollary of an otherwise sobering number.

The architectural response the platform adopts — reifying work as a **packaged item** with explicit acceptance criteria, and adjudicating completion against emitted evidence by a party *other than the executing agent* ("north-star adjudication") — is a direct attempt to lower C_v structurally rather than to eliminate it. Consistent with the paper's honesty discipline, we note this mechanism was built and reviewed but not yet enabled by default during the observation window; it is a designed reduction of C_v, not yet a measured one. That verification is being *engineered as a first-class cost* is itself the evidence for Proposition 1: a cost that collapsed with capability would not warrant a dedicated subsystem.

6.2 Observability does not collapse — and grows against production

The most striking non-collapse is observability. A system producing 64,000 records a day should, one imagines, be richly knowable. The opposite held: production of *state* outran the ability to *observe* it.

The census. In a single operating session, an intelligence attempting to operate the platform was forced into nine distinct workarounds — reaching around the product into raw infrastructure — because the system could not answer basic questions about itself: which version was live, why a transaction failed, what created a particular

account, what was slow. Each workaround is a unit of observability cost the system imposed rather than absorbed.

The adversarial growth. Worse, the observability cost *rose with production volume*. A family of user-facing failures — every affected page reporting, opaquely, "offline" — traced to the system scanning its own 52-million-record substrate on the request path. The primary evidence ledger (11 million rows) was queried with an `OFFSET (total − N)` pattern that forces the database to scan and discard ~11 million rows to reach the offset. The larger the substrate grew, the slower the system became at reading itself — observability cost scaling *with* production, the signature of adversarial growth. Table 3 reports the remediations, which replaced raw-substrate reads with maintained projections, index-backed existence checks, and planner estimates.

Table 3 — Observability remediations (single-run, mechanism-demonstrating).

Operation	Before	After	Factor
Chain verification (large tenant)	45 s	2.3 s	≈19×
Recent-N: `OFFSET` → index cutoff	16.3 s	0.10 s	163×
Public endpoints (unauthenticated)	60 s+	0.1–0.2 s	≈300×
Tenant `COUNT` → projection	44.75 s	7 ms	≈6,000×

Seventeen request-path substrate scans were found; the worst were *public, unauthenticated* endpoints running 44-second aggregates on every anonymous request — simultaneously a performance defect and a denial-of-service surface. The lesson is general and is the empirical core of Proposition 1's observability clause: **the ability to observe a system is a separate artifact from the ability to produce it, and AI production creates the second without the first.**

The reflexive finding. Because the operator during this session was itself an AI agent, we obtain a reflexive test: the non-production costs bound AI operators, not merely humans. Indeed the AI operator hit a sharper version — a diagnostic query reported a core table "empty (0 rows across 147 tenants)," which would have been a catastrophic false finding; it was an artifact of the query running without the security context that governs row visibility, and the true count (156) was recovered only by an explicit bypass. *The instrument was blind for the same structural reason the system was.* This yields a meta-principle — verify the measurement harness before trusting the measurement — and, more importantly for the thesis, direct evidence that observability cost is a property of the *system*, binding any intelligence that operates it.

6.3 Trust does not collapse

The third non-collapse is trust. Producing intelligence cheaply does nothing to establish that the intelligence may be trusted with a customer's authority, credentials, and audit. The platform's design separates three concerns that the standard reseller model fuses: *paying* for inference, *holding* the credential, and *proving* what happened. Its stance — the customer's own model key, routed directly to the provider, never platform-funded or platform-proxied, with an independent hash-chained ledger as witness — is a structural reduction of C_t: it removes the platform as a custody honeypot and as a single point of trust failure, and it separates the party that runs inference from the party that attests to it.

The cost's non-collapse was demonstrated by a production incident. Before the stance was fully enforced on one path, the platform routed customers' inference through its own shared gateway. That gateway's authentication

token expired. Because it was a *shared platform credential*, **every** provider routed through it returned `401`, silently disabling inference for **every** tenant at once — while each tenant's *own, valid* credential sat unused. One expired platform secret broke everyone; no amount of model capability would have prevented it, because the failure was a trust-topology failure, not a capability failure. The remediation (BYOK routed direct-to-provider) removed the shared point of trust failure entirely; verified live, the assistant streamed real output from the customer's own key where it had previously failed. The evidence for Proposition 1's trust clause is exactly this independence: trust is a property of *how authority and custody are arranged*, and that arrangement does not improve when the models do.

The same non-collapse recurs at a second boundary. A freshly-linked integration reported itself "expired or revoked" though its credential was valid at the provider — the platform having stored the connection under an internal identifier the provider had never issued, so the platform's own health check disowned a working connection. Two subsystems, two mechanisms, one disease: **a platform-internal identity or routing layer breaks the user's valid credential and then blames the user.** Every layer that re-identifies or proxies a customer asset is a place trust silently fails — a cost that scales with the *number of such layers*, not down with model capability.

---

7. Analysis — estimating the bound and reading the evidence

7.1 A sensitivity analysis of the bound

Proposition 2 bounds the AI economy of scale at $1/(1-p)$. Estimating p rigorously requires an activity-cost accounting we do not claim to have; we instead perform a *sensitivity analysis* — reporting the bound across the plausible range of p and of the production collapse factor k — so the reader can locate their own priors. Table 4 evaluates $S(k) = 1/[(1-p)+p/k]$.

Table 4 — Delivered-value speedup S as a function of production fraction p and collapse factor k.

p (production share)	k=10	k=100	k=1000	$k \to \infty$ (the bound)
0.5	1.8×	2.0×	2.0×	**2.0×**
0.6	2.2×	2.4×	2.5×	**2.5×**
0.7	2.7×	3.2×	3.3×	**3.3×**
0.8	3.6×	4.6×	4.9×	**5.0×**
0.9	5.3×	9.2×	9.9×	**10.0×**

Two features of Table 4 carry the argument. First, **the bound is reached early**: at p=0.7, going from a tenfold to a thousandfold production collapse moves delivered speedup only from 2.7× to 3.3× — a hundredfold increase in raw production capability buys a 22% increase in delivered value. This is the vanishing return of Corollary 2.1, made numeric. Second, **the columns matter far less than the rows**: delivered value is governed by *where the cost sits* (p), not by *how capable the model is* (k). A firm at p=0.9 (production-dominated) extracts a 10× economy; a firm at p=0.5 (verification/observability/trust-dominated) extracts 2×, from the *same* models. The strategic lever is the row, not the column.

Software cost models place construction below the majority of lifecycle cost once verification, integration, and operations are included (Boehm, 1981), which would put many real pipelines in the $p \approx 0.5$–0.7 band — i.e., near a 2–3.3× ceiling. Our measured evidence is consistent with this band: production output rose by orders of magnitude (§5, a large k), while the delivered system remained bottlenecked on verification (37.6% human escalation), observability (a family of "offline" failures growing with the 52-million-record substrate), and trust (the gateway incident) — the low-single-digit system-level gap the bound predicts for a mid-range p. We stress this is a *consistency* result, not a point estimate of p; §20 treats the difficulty of measuring p as a first-order limitation and §21 proposes the accounting that would pin it down.

7.2 What worked

- **Removing inter-human coordination** (single builder + agents) delivered the production collapse cleanly, escaping the Brooks's-Law term (§5.2).
- **Converting discipline into automated structure** bent the non-production curves down. Seven fail-closed deploy gates — each born from a specific class of production incident (a login failure from code shipping ahead of its schema; a checkout failure from a stale price identifier; a silent missing-dependency build) — turned recurring verification and observability failures into named, blocking, one-time events. The measured signature is the fix-to-feature ratio moving from **2.1 : 1 to 0.88 : 1** across 4,742 commits: a system building faster than it repairs, at scale. This is the empirical claim that $(1-p)$ is *bendable*: structural investment (gates, projections, adjudication) lowers the non-production fraction and raises the bound.
- **The projection-read pattern** (§6.2) lowered observability cost by up to ~6,000×, converting an adversarially-growing cost into a bounded one.
- **Separating payment, custody, and provenance** (§6.3) removed whole classes of trust failure structurally rather than patching them.

7.3 What did not work — and the honesty about it

- **Self-declared completion** failed and had to be replaced by external adjudication; where self-grading remained, the verification cost surfaced as a 37.6% (historically 80%+) escalation load and a mis-calibrated escalation-to-approval ratio.
- **Self-observability was absent by default.** The system could not answer basic questions about itself; the capability to do so existed (a database that could always compute its slow queries; a deploy that always knew its version) but the *surface* did not — most observability defects were missing surfaces over existing capability, not missing capability.
- **Trust-by-routing failed** (the shared-gateway and phantom-identity incidents): the platform's own convenience layers disowned valid customer credentials.
- **Formalism theatre.** An internal decision-making framework of six mathematical components, audited honestly, contained only two load-bearing "engines"; the other four were organising heuristics, and in one case a formula that existed in doctrine (a recency-weighted trust score) was found *hard-coded to a constant* in a live path. We report this against our own work because a paper about verification cost that did not verify its own claims would refute itself.

7.4 A worked vignette: the three costs interacting

The terms are cleanest in the abstract and most convincing in a single concrete failure where all three appear at once. Consider one incident from the observation window: a checkout endpoint returning an opaque server error.

- **Production (collapsed).** The endpoint had been produced quickly and looked correct; its logic was sound. Production had done its job cheaply — the code existed and read plausibly.
- **Verification (did not collapse).** The failure was not one bug but *three, layered*, each masking the next. The first was a stale configuration identifier; fixing it revealed a second, an invalid parameter combination the payment provider rejected; fixing that revealed a third — an entire *class* of latent defects introduced by a library-version upgrade in which response objects had silently stopped supporting a method the code called throughout. No amount of production capability surfaces layered defects; only verification does, and here verification required tracing each layer live because self-declaration had certified the endpoint as done. A subsequent sweep found seventeen more instances of the third defect class elsewhere — the verification cost of one produced feature rippling across the codebase.
- **Observability (did not collapse; obstructed the diagnosis).** Establishing *why* the endpoint failed required reaching around the product into raw logs and in-pod reproduction, because the system could not report its own failure cause — the observability defect of §6.2, here not a separate failure but the *tax on diagnosing* the verification failure.
- **Trust (implicated).** The root of the third defect was the same class as the trust incident of §6.3 — a platform layer's assumption about a provider's interface that no longer held — showing the terms are not merely co-present but *causally entangled*: a trust-boundary change manufactured a verification burden that an observability gap made expensive to discharge.

The vignette is the framework in miniature: production was free, and the entire cost of turning that free production into a working, trustworthy checkout was borne by verification, observability, and trust — exactly the non-collapsing terms, exactly interacting, exactly as Propositions 1 and 3 and §3.5 predict.

7.5 Synthesis

The evidence composes into a single account. AI collapsed production (§5). The saved cost did not become free value; it re-emerged as verification (§6.1), observability (§6.2), and trust (§6.3) costs that model capability does not touch. Where those costs were attacked *structurally* — gates, projections, adjudication, custody separation — they fell, and the system's health improved measurably (§7.2). Where they were left to capability or to good intentions, they bound the system (§7.3). This is the AI economics of scale: **the returns to capability are real but bounded; the returns to structurally lowering the non-production costs are where scale is actually won.**

Part II — The Architecture of Scale: Bending the Non-Production Curves

Part I established a bound: the AI economy of scale is capped at 1/(1-p), the reciprocal of the non-production cost fraction, and that fraction does not fall with model capability. It also established, in §7.2, that the fraction is nonetheless bendable *by structural investment. Part II is about the structure. If Part I is the physics — what cannot be escaped — Part II is the engineering — what can be built to approach the bound. Each of the following sections takes one non-production cost and describes a concrete, measured mechanism that lowers it, then generalises the mechanism into a principle any builder of scalable AI systems can apply. The three non-production costs have operational names that a practitioner will recognise: verification is "does the work*

reliably finish?", observability is latency and knowability, and trust is safety and cost-control. The architecture below is, therefore, simultaneously an answer to how AI scales and to the three questions every enterprise asks of it — will it finish, will it be fast and legible, and will it be safe and affordable.

8. From the bound to the architecture

The strategic content of Proposition 2 is a target: to raise the achievable economy of scale, lower (1-*p*). This is not a metaphor but an engineering programme with three work-streams, one per non-production cost. The programme's premise, borne out by §5–§7, is that each cost is dominated by a *structural* deficiency — a missing unit of work (verification), a missing read model (observability), a missing routing and governance topology (trust and cost) — rather than by anything model capability addresses. It follows that the same capability, deployed on a system with these structures, delivers a materially higher fraction of its potential value than on a system without them. Part II describes six such structures, drawn from a single production system and measured on it, and argues that together they constitute a *reference architecture* for scalable AI (§13.7) — the science of scaling that the bound demands but does not itself supply.

We stress the epistemic status up front, consistent with the paper's discipline: these mechanisms are *implemented and measured on one system*, and the measurements demonstrate that each lowers its target cost; they do not prove the mechanisms are optimal or unique, and §20 treats their generalisation as an open, testable claim rather than a settled result. The contribution of Part II is a *worked, instrumented existence proof* that the non-production curves are bendable and *how* — which is exactly the constructive complement Part I's bound requires to be actionable.

9. Work as the unit of scale: the packaged work-item

The first and most consequential structure attacks verification cost by changing the *unit* of AI work.

9.1 The prompt is the wrong unit

The default unit of AI work is the prompt: a natural-language instruction, executed once, producing an output. The prompt is a superb interface for a human eliciting a single response. It is a catastrophic unit on which to build an *economy* of work, for four structural reasons. A prompt is **ephemeral** (it leaves no durable, inspectable record of what was asked, under what constraints); **unroutable** (it carries no machine-readable statement of the capability it requires, so it cannot be dispatched to the best-fit executor); **unpriceable** (it has no bound budget or metered cost, so work cannot be costed or capped); and — the crux for Part I — **unverifiable** (it carries no acceptance criteria, so "done" can only be self-declared). One cannot build a market, a supply chain, or a governance regime on prompts, for the same reason one cannot build a logistics network on verbal requests: the unit does not carry the metadata the system needs to route, price, govern, and confirm it.

9.2 Reifying work as a first-class item

The structure is to promote work from a prompt to a **packaged, first-class, durable item** — a *work packet* — carrying, as machine-readable fields, everything the system needs to route, govern, price, and verify it:

- **Intent** — the goal in the requester's terms.
- **Bound resources** — the specific accounts, connectors, and data the work is *permitted* to touch (an allow-list, not an implicit grant).

- **Acceptance criteria** — the checkable conditions that constitute "done."
- **Budget** — the spend and step ceiling.
- **Governance tier** — the authority class (auto / soft / mediated / hard) of the actions the work may take.
- **Context** — the assembled situation and the executable instruction.

The move is small to state and large in consequence. Each field closes one of §9.1's four deficiencies: bound resources and governance tier make the packet *governable*; the budget makes it *priceable*; the intent and required-capability make it *routable*; and the acceptance criteria make it *verifiable*. The packet is the smallest object that carries its own governance and its own definition of success. In the studied system this is not a proposal but a running spine: work is dispatched as durable, status-carrying items (§9.4), and a completion adjudicator (§10) checks the acceptance criteria against emitted evidence.

9.3 "Uber for work": routing work to capacity

Once work is a first-class item bearing its required capability, a second structure becomes possible: a *dispatch market*. Just as a ride-hailing platform routes a trip (a packaged request with an origin, destination, and constraints) to the best-available driver by proximity, rating, and cost, a work-dispatch layer routes a work packet to the best-available *executor* — an individual agent, a multi-agent workforce, a mixed human-and-agent pod, or a human — by a scoring over **capability × availability × workload × cost × trust**. The executor is not fixed at authoring time; it is *selected*, per packet, against the current supply of capacity. We call this, for its structural analogy, **"Uber for work"**: work becomes a routed, priced, governed item flowing to whatever capacity can best complete it, rather than a prompt hard-wired to a single model.

This reframing has a specific scaling payoff beyond routing efficiency. It makes *heterogeneity* a first-class property: because the packet carries its required capability rather than a chosen model, a single work stream can be served by many different executors — different AI providers, self-hosted models, human specialists — each selected where it is best and cheapest (the cost consequence is developed in §12). The dispatch layer is where the cost, latency, and capability trade-offs of Part I's non-production terms are actually *made*, per unit of work, rather than fixed globally.

9.4 The measured coordination substrate

The dispatch model is live and load-bearing, not a design sketch. In the studied system, work flows over a coordination substrate carrying, at the time of measurement, **1,213 coordination tasks, 4,131 inter-executor messages, and 31 registered executors**, with inter-agent communication itself governed and recorded as **178,910 envelopes** — every delegation and hand-off is a governed, evidence-emitting action, so the supply chain of work is auditable end to end. Work packets are executed as durable, status-carrying runs; across the 237 recorded runs analysed in §6.1, the status distribution (43% self-completed, 37.6% escalated, 18.6% failed) is precisely the *verification signal* the packet structure exposes: because each packet carries acceptance criteria, the system can *decline* to mark incomplete work done and route it onward, which a prompt-based system cannot.

9.5 Metering and pricing work as an economic unit

Reifying work as a packet does not only make it routable and verifiable; it makes it *economic*. A prompt cannot be priced, metered, capped, or sold, because it carries no budget and leaves no accounting. A packet carries a budget field and, on execution, accrues a metered cost against it — so work becomes a unit that can be *costed before it runs, capped while it runs, and reconciled after it runs.* This is the precondition for two things a prompt-based system cannot have: a real budget discipline (no packet can exceed its ceiling, so runaway agent spend is

structurally bounded rather than hoped for), and a genuine *marketplace* (work with a price and a definition of success can be offered, bid on, and settled — the economic layer atop the dispatch market of §9.3).

The studied system meters work at this granularity. Beneath the 237 packet-level runs sits a fine-grained ledger of token-allocation records — the atomic accounting of compute consumed against each packet's budget — and per-call inference-usage rows attributing every model call to a tenant, a model, and a cost (§12.3). This is what turns "an agent did some work" into "this packet consumed X of Y model's capacity against its Z budget, and here is the receipt" — an accounting a prompt does not produce and a governed economy of work requires. The budget field is also a *safety* structure, not only a cost one: an irreversible or high-authority action inside a packet that has exhausted its budget is refused, so the budget ceiling is simultaneously an economic cap and a blast-radius limit — an instance of the cost–safety coupling §17 develops. Work-as-a-packet is thus the unit at which the cost, verification, and safety controls all attach; the prompt is the unit at which none of them can.

The pricing consequence completes the "Uber for work" analogy. A ride-hailing platform can price a trip because the trip is a bounded, specified item with a metered cost; it can build a marketplace because priced, specified items can be matched to supply. A work-packet is the equivalent bounded, specified, metered item for AI work — which is why the packet, and not the prompt, is the unit on which an *economy* (a priced, governed, auditable market for AI-performed work) can be built. That fine-grained allocation ledger and the metered per-call spend are that economy in miniature: work costed to the fraction of a cent, capped by budget, and reconciled against a governed ledger.

9.6 Generalisation

> **Principle 1 (The unit of scalable AI work is the governed, criteria-bearing packet, not the prompt).** To scale AI work beyond a single elicitation, work must be reified as a first-class item carrying its intent, bound resources, acceptance criteria, budget, and governance tier. Only then can work be routed to the best-fit capacity, priced and capped, governed by authority class, and — decisively — verified against a definition of success it carries with it. The prompt is where AI work begins; the packet is where it scales.

This principle is the constructive core of the verification-cost result. Part I showed verification does not collapse; the packet is what makes the residual verification *tractable and adjudicable* rather than a diffuse human obligation — the subject of §10.

10. Ensuring completion: north-star orchestration

The packet defines success; something must *hold the executor to it*. This section describes the mechanism that lowers verification cost from "a human must check everything" toward "the system adjudicates completion against evidence," and it begins with the failure it exists to prevent.

10.1 The universal failure: goal drift

The dominant failure mode of autonomous multi-step work is not a wrong answer; it is **goal drift**. An agent, blocked on a sub-task, retreats to something it *can* do; it thrashes among alternatives; it loses the thread of the original objective; it declares the sub-task complete; and it thereby reports success while failing the actual ask. This is a structural property of open-ended autonomous execution — the longer the horizon and the more sub-tasks, the more surface for the objective to slip — and it is *invisible to self-grading*, because the agent, having drifted, sincerely believes the drifted task was the point. Goal drift is why "the model can do each step" does not

imply "the system completes the job," and it is the concrete mechanism behind the 37.6% escalation rate (§6.1): the system's honest response to un-adjudicable completion.

10.2 The four disciplines

The structure that addresses goal drift — **north-star orchestration** — imposes four disciplines on every packet's execution:

1. **Anchor the objective.** The packet's intent is the *north star*, held as the fixed reference for the whole run.
2. **Re-inject at every step.** The objective is re-presented to the executor at each step, so drift is corrected continuously rather than discovered at the end.
3. **Diagnose, do not retreat.** On a blocker, the discipline is to diagnose the blocker against the objective and resolve it — not to substitute an easier reachable task. (This is the behavioural analogue of the entropy-minimisation compass of the companion mathematics paper: route toward the objective, not away from the obstacle.)
4. **Adjudicate completion.** "Done" is decided by verifying the packet's acceptance criteria against the *emitted evidence*, by a party other than the executing agent — never self-declared.

The fourth discipline is the load-bearing one and deserves emphasis. It converts verification from an act of *trust* (believe the agent's report) into an act of *checking* (confirm the criteria against the ledger), and it relocates the judge outside the judged. In the studied system a completion adjudicator implementing this is present and reviewed; and the discipline has a proven precedent elsewhere in the same platform, where readiness for a code merge is *not* something a worker may self-assert — a substrate-side adjudicator fetches the ground-truth state (the exact commit, the required checks, the mergeability) and emits the canonical verdict, so a worker may only *submit a claim* and the adjudicator decides. North-star adjudication generalises that discipline from "is this code ready?" to "is this work done?"

10.3 Why this lowers verification cost

Adjudication does not eliminate verification cost — Part I forbids that — but it changes its *scaling*. A human check is O(\text{runs}) and does not parallelise; an evidence check against acceptance criteria is automatable and O(1) per criterion. Adjudication therefore converts the *bulk* of verification from an un-scalable human obligation into a scalable machine one, leaving humans only the *irreducible* residue — genuinely ambiguous completions — rather than every completion. It also directly attacks the *waste* component identified in §6.1: the mis-calibrated over-escalation (historically 80%+ escalation, ~98% of escalations ultimately rejected) is precisely the cost of having *no* trustworthy adjudicator, so the system escalated everything to be safe. A grounded adjudicator that can *certify* a completion removes most of that waste. In the four-term model, north-star adjudication is the mechanism that makes C_v approach its irreducible floor instead of its self-grading ceiling.

10.4 Generalisation

> **Principle 2 (Completion must be adjudicated against evidence, not self-declared).** In any autonomous multi-step system, "done" decided by the executing agent is not evidence of completion; it is evidence of the agent's belief, which goal drift renders unreliable. Scalable autonomous work requires that the objective be anchored and re-injected, that blockers be diagnosed against the objective rather than retreated from, and that completion be adjudicated against the packet's acceptance criteria by a party other than the executor. This is the single most important primitive for converting capable agents into *finishing* systems.

11. Observability at constant cost: the event-ledger and projection pattern

This section attacks observability cost — operationally, *latency and knowability* — and defuses the diseconomy of Proposition 3 by construction.

11.1 The problem restated as an architecture failure

Proposition 3 showed observability cost rising as O(*V*) with cumulative produced state, because observation was performed by *traversing* that state. In the studied system this was literal: reads scanned an 11-million-row ledger (and, across the substrate, tens of millions of rows), so every added record made the system fractionally slower to read itself — the mechanism behind the "everything is offline" failure family (§6.2). The failure is not the volume of state; abundant state is the point of an audit system. The failure is *answering questions by traversing the state*. The architecture must sever the read path from the write path.

11.2 The pattern: append-only ledger + maintained projections + estimates

The structure is a disciplined form of Command-Query Responsibility Segregation (CQRS):

- **The write side is an append-only event ledger** — the source of truth, optimised for integrity and immutability, never for query. In the studied system this is the hash-chained evidence ledger (11.05 million rows) plus the append-only action ledger (**9.39 million rows**): ~20 million immutable events that are *written*, essentially never *scanned* on a request.
- **The read side is a set of maintained materialised projections** — small, query-shaped aggregates recomputed from the ledger on a schedule. The primary such projection in the studied system is a daily rollup of **14,486 rows** standing in for the 11-million-row ledger — a **≈760× compression** — answering "how many, of what type, when" in milliseconds without touching the ledger.
- **Whole-population counts use planner estimates** — a database's own row-count statistics (`pg_class.reltuples`) answer "roughly how many total?" in O(1), reserving exact counts for the rare cases that need them.

The effect is to convert observability cost from O(*V*) to O(1) or O(\log *V*): reads answer from the small projection or the estimate, whose size is independent of (or logarithmic in) how much the system has ever produced. Proposition 3's diseconomy is not merely mitigated; it is *removed by construction* — observation no longer traverses production.

11.3 The latency consequence, measured

Because observability cost *is* read latency, defusing the diseconomy is a latency result, and it is the paper's answer to the latency dimension of scale. The remediations of Table 3 are exactly the substitution of ledger scans by projection and estimate reads: chain verification **45 s → 2.3 s**; a tenant count **44.75 s → 7 ms** (a ≈6,000× reduction); public endpoints **60 s+ → 0.1–0.2 s**. Seventeen request-path scans were relocated to the read side. The general claim is strong and, we think, under-appreciated in AI-systems design: **an AI system that logs, remembers, or audits its own behaviour — as any governed or self-improving system must — will accumulate state without bound, and its latency will degrade with that state unless the read path is architecturally separated from the write path.** The projection pattern is how observability stays a constant cost as production scales, which is the precondition for the latency of a scaling AI system to stay flat.

11.4 Generalisation

> **Principle 3 (Separate the write ledger from the read projections; never answer a query by scanning production).** In any AI system that accumulates state, observability cost — and therefore read latency — grows with cumulative production unless observation is served from maintained projections and planner estimates rather than from the source-of-truth ledger. The append-only ledger is written and never scanned on the request path; the projections are read and never authoritative. This construction is what converts the observability diseconomy (Proposition 3) into a flat, constant cost, and it is a precondition for a self-observing AI system to scale without its latency degrading against its own memory.

12. Tiered routing: the cost and latency architecture

This section attacks the cost dimension directly, and latency again from the routing side. It develops the consequence of §9.3's heterogeneity: once work is routable, *where* it is routed determines its cost and speed.

12.1 The problem: uniform routing to the frontier is expensive and slow

The default deployment routes every request to a single, capable, frontier model. This is simple and wrong at scale, for a reason the four-term model makes precise: it pays the frontier's marginal cost and latency for *every* unit of work, including the majority of units that a far cheaper, faster model would serve identically. In a system doing high-volume operational work — classification, extraction, routing, drafting — the frontier is over-provisioned for most calls, and the inference bill and tail latency scale linearly with volume at the frontier's unit cost.

12.2 The structure: a capability ladder with a self-hosted floor

The structure is a **multi-tier routing ladder**, with each unit of work routed to the *lowest* tier that can serve it, by task complexity:

- **Tier 0 — self-hosted.** A small family of *fine-tuned, task-specialised* small models (in the studied system, a set of fine-tuned 8-billion-parameter models served on a serverless inference plane, specialised for the platform's own recurring tasks). Their marginal cost is the compute floor — effectively free at the margin for the highest-volume, lowest-complexity work — and they can be co-located for low latency.
- **Tier 0.5 — free-tier providers.** Provider free allowances, used before any paid call.

- **Tiers 1–3 — budget, mid, and premium providers.** Paid capability, ascending in cost, reserved for the work that genuinely needs it, with the frontier (tier 3) reserved for the hardest reasoning.

Two further structures make the ladder economical. First, **bring-your-own-key** (BYOK, developed in the companion paper): the tenant's own provider credential pays for the tenant's own inference, routed directly to the provider — so the platform bears no token cost and takes no token risk, and the tenant sees provider price competition directly. Second, **complexity-based routing**: a lightweight classifier estimates each task's difficulty and routes it to the lowest sufficient tier, so the ladder is descended by default and ascended only on demand.

12.3 The measured economics

The ladder's effect is directly measurable. Across the recorded inference calls, the blended cost per call was a fraction of a cent, and **~14% of all calls cost nothing**, served by the free and self-hosted floor. The routing spread is real: **seven distinct providers and thirteen distinct models** were actually used, evidence that work descended and ascended the ladder rather than piling onto one model. The economic reading is stark against the uniform-frontier baseline: a frontier-only deployment of the same call volume at a typical frontier unit cost would be **one to two orders of magnitude more expensive**; the ladder plus the self-hosted floor is what keeps the average call at a fraction of a cent. This is the cost answer to scaling AI: **inference cost stays bounded as volume scales not by using a cheaper model, but by using the cheapest *sufficient* model per unit of work, with a self-hosted floor capping the marginal cost of the high-volume tail.**

The latency consequence follows the same structure: latency-sensitive and high-volume work routes to fast local or budget tiers, reserving the slower frontier for the rare hard calls, so the *tail* latency of the frontier is not paid on every request.

12.4 Generalisation

> **Principle 4 (Route each unit of work to the lowest sufficient capability, with a self-hosted floor and pay-your-own-key).** Inference cost and latency scale linearly with volume only if every unit is served by the frontier. A capability ladder — a self-hosted, task-specialised floor beneath free and paid provider tiers — with complexity-based routing that descends by default, and with the consumer's own credential paying for their own inference, keeps the average cost per unit a fraction of the frontier's and the tail latency off the common path. The economically scalable AI system is not the one with the best model; it is the one that uses the best model *only where the work requires it*.

13. Governance by tier: safety that scales

The final structure attacks trust cost — operationally, *safety and its scalability*. It resolves the central tension of autonomous AI: a human in every action loop is safe but does not scale; no human in any loop scales but is not safe.

13.1 The tension made precise

Safety in autonomous systems is conventionally purchased with human oversight: a person approves each consequential action. This is safe and *anti-scaling* — it reintroduces a human into the loop the automation was meant to remove, and the studied system's own forensics show why it fails at volume: with a naive escalation policy, escalation exceeded 80%, the escalation-to-approval ratio was on the order of hundreds to one, and ~98%

of escalations were ultimately rejected (§6.1). Universal human oversight does not merely fail to scale; it *drowns* the human, who then rubber-stamps — the worst of both worlds. The structural question is therefore: how does safety scale *without* a human in every loop?

13.2 The structure: graded governance, an immutable floor, and an evidence substrate

The answer is three co-designed structures.

Graded governance. Every action an agent may take is classified, by risk, into one of four tiers — **AUTO** (proceed and record), **SOFT** (proceed with a lightweight check), **MEDIATED** (a checkpoint, trust-gated), and **HARD** (explicit human approval required). The overwhelming majority of actions — reads, low-risk operations, reversible steps — are AUTO: they proceed at machine speed and are merely *recorded*. Only the genuinely risky, irreversible, or high-authority actions are escalated to a human. Human oversight is thus spent where risk warrants it, not uniformly — the mechanism by which safety scales. Trust-gating sharpens this further: proven patterns earn higher autonomy over time (a pattern that has succeeded reversibly many times may graduate from MEDIATED to AUTO), while new or irreversible actions default to caution — a recency-and-evidence-weighted autonomy dial.

An immutable floor. Beneath the graded tiers sits an unamendable prohibition — in the studied system, a "Law Zero" that the platform must never be used to cause harm — encoded as an *infinite-cost barrier* (the Bellman formalism of the companion mathematics paper): a class of actions the optimiser can never traverse regardless of how cheap or approved they appear. Graded governance decides *how much oversight* an action needs; the floor decides *what is never permitted at all*, and it is not subject to the trust dial.

An evidence substrate. The reason graded autonomy can be safe is that it is *accountable*: every action — AUTO included — emits an immutable, hash-chained, attributable evidence record. In the studied system this is the **9.39-million-row action ledger** and the 11-million-row evidence ledger. Because every autonomous action is provable, auditable, reversible, and attributable after the fact, autonomy does not mean unaccountability. The evidence substrate is what makes it safe to let most actions proceed without prior human approval: oversight shifts from *ex-ante* (approve before) to *ex-post* (prove and, if needed, reverse after) for the low-risk majority, reserving ex-ante approval for the high-risk minority. Safety scales because accountability, not approval, becomes the default control.

13.3 The honest frontier: grounded tiering is the hard part

Part II's discipline requires naming where this structure is not yet solved. The efficacy of graded governance depends entirely on *correct tiering* — on classifying each action's risk accurately. The studied system's forensics expose the failure mode when tiering is *ungrounded*: a large fraction of actions were **mis-tiered**, driving the escalation floods of §13.1. The finding that a single class of ungrounded severity mappings produced most of the mis-tiering is instructive: heuristic, keyword-based risk classification does not scale, because it mis-estimates risk and either escalates the safe (drowning the human) or, more dangerously, auto-approves the risky. The frontier of scalable safety is therefore *grounded* risk assessment — tiering that reflects an action's true reversibility, blast radius, and authority rather than a surface heuristic. This is unsolved, and we present it as the open problem it is; the *architecture* (graded tiers + immutable floor + evidence substrate) is sound and necessary, but its safety guarantee is only as good as the grounding of its risk classifier, which remains a research frontier.

13.4 Generalisation

> **Principle 5 (Safety scales through graded, evidence-backed governance, not universal human approval).** Autonomous AI cannot be made safe at scale by putting a human in every loop — that does not scale and, at volume, degrades into rubber-stamping. Safety scales through three co-designed structures: graded governance that reserves human approval for the genuinely risky minority while the low-risk majority proceeds accountably; an immutable, non-negotiable floor of prohibited actions; and an evidence substrate that makes every autonomous action provable, reversible, and attributable, so that oversight can shift from ex-ante approval to ex-post accountability for the common case. The unsolved core is grounded risk tiering; the architecture around it is what lets safety scale with autonomy rather than against it.

14. The reinforcement flywheel: scaling through accumulated evidence

The five principles of §9–§13 lower the non-production costs of *current* work. A sixth structure lowers them for *future* work, by turning the evidence a governed system already emits into a learning signal — so that the system improves with use, at near-zero marginal training cost. This is the scaling dimension that compounds.

14.1 The evidence substrate as a learning corpus

A governed AI system that records every action, decision, and outcome is, incidentally, generating the highest-value training corpus for its own domain: not scraped web text, but *labelled records of governed work* — what was asked, what the system did, whether it was accepted, escalated, or reversed, and what the human decided when it escalated. In the studied system this corpus is the 52-million-record substrate: 11 million evidence artifacts, 9.4 million action records, and — critically — the *outcomes* attached to them (the 237 runs' accept/escalate/fail labels of §6.1 are the smallest, cleanest slice; the historical ledger's accept/reject signals on tens of millions of actions are the larger one). Every escalation that a human approves or rejects is a labelled preference pair; every run that completes or fails is an outcome label; every action that is later reversed is a negative signal. The system does not need to *acquire* this data — it emits it as a byproduct of operating under governance. The governance substrate that Part I framed as a *cost* (observability, trust) is, from this angle, an *asset*: the labelled record that makes the system improvable.

14.2 Two learning loops: fast and slow

The corpus feeds two loops of different time constants.

The fast loop — conductance reinforcement. At the routing and dispatch layer, paths that succeed are reinforced and paths that fail or go unused decay, following the Physarum-style dynamics of the companion mathematics paper ($dP/dt = \alpha|F| - \beta P$). This is a *within-operation* loop: the system's routing preferences shift continuously toward what has recently worked, without any model retraining — proven patterns earn autonomy (the trust-gating of §13.2), stale patterns are re-validated. It is cheap, immediate, and — per the mathematics paper's honest audit — genuinely load-bearing in the flow-control layer, though its exact constants are chosen rather than fitted.

The slow loop — specialist fine-tuning. At the model layer, the accumulated preference and outcome data fine-tunes the self-hosted specialists (§15). Where a task is high-volume and narrow — the platform's own recurring operations — the labelled records of how that task should be done become a training set, and a small model fine-

tuned on them can match or exceed a general frontier model *on that narrow task*, at a fraction of the inference cost. The studied system has trained a family of such specialist adapters (§15) from exactly this kind of operational data; each is a slow-loop product of the fast loop's accumulated signal.

14.3 Why the flywheel is a scaling mechanism

The flywheel matters for the economics of scale because it *compounds*, and it compounds on the non-production side. Each unit of governed work emits evidence; the evidence improves routing (fast loop) and specialists (slow loop); improved routing and specialists lower the cost, latency, and verification burden of the *next* unit of work; lower per-unit cost enables more work, which emits more evidence. The loop's product is a per-unit non-production cost that *falls with cumulative operation* — the opposite of the observability diseconomy of Proposition 3, and the mechanism by which (1-*p*) can fall over time rather than merely being bent once. Where Principle 3 defused a diseconomy of scale, the flywheel is a genuine *economy* of scale on the non-production side: the more the system has done, the cheaper and more reliably it does the next thing.

14.4 Honest status

We are careful here, because "the system learns from itself" is the most over-claimed idea in applied AI. What is *measured and live*: the substrate exists at the stated scale and carries outcome labels; the fast-loop conductance reinforcement runs in the flow-control layer; specialist adapters have been trained from operational data (§15). What is *designed but not yet demonstrated at the level of a controlled result*: that the slow loop produces a measurable, sustained improvement in delivered value over time — i.e., that (1-*p*) demonstrably falls with cumulative operation. Establishing that requires a longitudinal measurement we do not yet have (§21 lists it as future work). The flywheel is therefore presented as a *mechanism with a live substrate and a plausible, partially-instantiated loop*, not as a proven compounding result. Its inclusion is warranted because the substrate and the loops are real; its limits are stated because the compounding is not yet measured.

14.5 Generalisation

> **Principle 6 (A governed system's evidence is its learning corpus; close the loop).** Any AI system operating under governance emits, as a byproduct, a labelled record of governed work — the highest-value domain-specific training and preference data obtainable. Feeding that record back — fast, into routing and autonomy; slow, into task-specialised models — turns the accumulated evidence into a falling per-unit non-production cost, a compounding economy of scale on exactly the side that Part I showed does not collapse from capability alone. The evidence substrate is not only the cost of governance; it is the asset that makes governed AI improve with use.

15. The economics of the self-hosted floor

Principle 4 asserted a self-hosted floor beneath the tiered ladder; this section develops its economics, because the make-versus-rent decision for inference is where a large share of the cost dimension of scale is won or lost.

15.1 The make-versus-rent structure

Inference can be *rented* (pay a provider per token, forever) or *made* (train a specialist once, serve it on owned or serverless compute at the compute floor). Renting has zero fixed cost and a linear marginal cost; making has a fixed training cost and a near-zero marginal cost. The crossover is volume: below some call volume, renting is

cheaper; above it, making dominates, and the dominance grows without bound with volume. For a platform's *own* highest-volume, narrowest tasks — the recurring operational work that constitutes the bulk of calls — volume is high and the task is narrow, which is precisely the regime where making wins decisively. The self-hosted floor is the architectural expression of this: the high-volume tail is served by made specialists at the compute floor, while the low-volume, high-variety head is served by rented frontier capability.

15.2 Specialisation beats generalisation on narrow tasks

The floor's models are not smaller general models; they are *fine-tuned specialists*. The studied system serves a family of task-specialised fine-tuned small (single-digit-billion-parameter) models behind an OpenAI-compatible interface — each specialised for one of the platform's own high-volume, recurring task types rather than for general use. The economic argument for specialisation is sharp: on a *narrow* task with abundant labelled examples (which the flywheel of §14 supplies), a small fine-tuned model can match a large general model's quality while costing one to two orders of magnitude less per call and running with lower latency and locality. Generalisation — one large model for everything — is the right architecture for the *head* of variety; specialisation is the right architecture for the *tail* of volume, and the tail is where the cost is. The platform's fine-tuning pipeline has produced on the order of fifteen such adapters across successive training runs, each trained (via parameter-efficient QLoRA fine-tuning on modest GPU budgets) from the operational corpus rather than from general data.

15.3 The measured floor

The floor's effect is visible in the aggregate cost of §12.3: ~14% of calls incurred zero marginal cost, served by the free and self-hosted floor, and the blended average was a fraction of a cent per call — a figure a frontier-only deployment cannot approach. The self-hosted floor is what makes the *marginal* cost of the high-volume tail approach the compute floor rather than the frontier's token price, and it is the structural reason the platform's total inference spend was one to two orders of magnitude below a uniform-frontier equivalent.

15.4 The training-cost caveat

Honesty requires the other side of the make-versus-rent ledger. Making carries a fixed cost — training compute, data curation, and, in the studied system's own account, the operational cost of a training pipeline that failed in instructive ways before it succeeded (pod misconfigurations, dependency-version mismatches, and data-format errors were each diagnosed and fixed across successive runs). The crossover volume above which making pays is real and must be respected: the self-hosted floor is an economy *only* for tasks whose volume clears it, which is why it is a *floor* under a rented ladder, not a replacement for it. The architecture's cost discipline is precisely knowing which tasks clear the crossover — high-volume, narrow, with a flywheel-supplied training set — and making only those.

15.5 Generalisation

> **Principle 4a (Make the high-volume tail; rent the high-variety head).** Inference cost at scale is governed by the make-versus-rent decision per task. High-volume, narrow tasks with abundant operational labels clear the crossover where a fine-tuned specialist served at the compute floor dominates per-token renting; high-variety, low-volume tasks do not. The cost-scalable architecture makes the tail and rents the head, and uses its own governed evidence (Principle 6) as the training set for the making.

16. Negative results: what did not scale

A paper that reported only the structures that worked would be an advertisement, not a study. This section reports, with the same evidentiary standard, the mechanisms that *failed* to bend the non-production curves — the negative results, which are as informative as the positive ones and which the honest reader should weight equally.

16.1 Self-declared completion failed (and had to be replaced)

The first and most expensive negative result: allowing the executing agent to declare its own work complete does not scale, because goal drift (§10.1) makes self-declaration unreliable in a way that is invisible to the agent. The measured cost of the failure was the escalation load — 37.6% in the studied sample, over 80% historically — as the system's honest fallback to human judgment in the absence of trustworthy self-declaration. The lesson (Principle 2) was learned by its failure: completion had to be relocated outside the executor. We report this as a negative result because the naive design — trust the agent's "done" — is the default in most agent systems, and its failure is not obvious until measured.

16.2 Heuristic risk-tiering failed (the mis-calibration)

The second negative result concerns safety's scaling. Graded governance (§13) is only safe if tiering is accurate; the studied system's *heuristic* tiering — classifying an action's risk by surface features such as verb keywords — mis-classified a large fraction of actions (the historical forensics found most actions mis-tiered), producing the escalation floods of §13.1 and, in the dangerous direction, occasionally auto-approving the risky. The negative result is specific and generalisable: **keyword- and verb-based risk classification does not scale**, because risk is a property of an action's reversibility, blast radius, and authority, not of its surface form. The lesson — that scalable safety requires *grounded* risk assessment — is stated as an open frontier (§13.3) precisely because the heuristic approach demonstrably failed.

16.3 Trust-by-convenience-layer failed (twice, identically)

The third negative result is the trust-topology failure of §6.3 and §13.5's cautionary examples, and it failed *twice by the same mechanism*: a platform convenience layer that re-identified or proxied a user's credential silently disowned a valid credential. Once it was an expired shared inference gateway (401 for every tenant); once a phantom connection identifier the provider had never issued. The negative result generalises to a design rule stated as a prohibition: **do not interpose a platform-owned re-identification or proxy layer between a user's credential and its provider** — every such layer is a place the platform can break the user while the user's asset is fine. The positive principle (BYOK-direct, of the companion paper) is the shape of the fix; the negative result is why the fix was necessary.

16.4 Formalism without grounding failed (the theatre)

The fourth negative result is reported against the system's own decision-making framework. An internal six-component mathematical framework, audited honestly (companion mathematics paper), contained only two load-bearing components; the other four were organising heuristics, and one — a recency-weighted trust score — was found *hard-coded to a constant* in a live path while existing in doctrine as a formula. The negative result is a methodological warning: **mathematical formalism that is not verified against its implementation is theatre**, and a system can carry the appearance of principled control (an equation in the design) while running an unprincipled constant (a hard-coded value in production). We report it against ourselves because it is the exact

failure a paper about verification cost would be embarrassed to have committed, and reporting it is the only defence.

16.5 Partial deployment failed (the drift)

The fifth negative result concerns the production side. Deploying a system's request-handling tier without deploying its asynchronous-worker and scheduler tiers in lockstep caused *drift*: in one measured incident, the request tier ran a recent version while the background workers executed code many releases stale — so agents ran outdated governance logic while the interface presented current behaviour. The negative result generalises: **a distributed AI system deployed partially desynchronises from itself**, and the desynchronisation is silent (each tier is internally healthy) and dangerous (the governance the interface promises is not the governance the workers enforce). The lesson — deploy all tiers of an AI system to the same version atomically — is a production-scaling discipline that the drift incident taught by violation.

16.6 The pattern in the negatives

The five negative results share a shape: each is a case where a *plausible default* (trust the agent's "done"; tier risk by keyword; proxy the credential for convenience; write the formula in the design; deploy the tier that changed) failed *silently* at scale, and the failure was invisible until it was measured against ground truth. This is itself a finding about scaling AI: **the failures that bound the non-production costs are not loud crashes but silent mis-behaviours** — a rubber-stamped escalation, a mis-tiered action, a disowned credential, a hard-coded constant, a stale worker — each of which a system reports as healthy while it quietly fails. The methodological corollary, which the whole paper practises, is that scaling AI safely requires *adversarial measurement against ground truth*, because the system's own report of its health is exactly what the silent failures corrupt.

17. Cost, latency, and safety: the three operational faces of the bound

The reader operating an enterprise does not think in the paper's terms — verification, observability, trust — but in three others: **cost, latency, and safety.** This section makes the mapping explicit, because it is the form in which the architecture answers the practitioner's actual question, and because the paper's claim is that a scalable AI system must solve all three *simultaneously*, which the six-structure architecture does.

Cost is the operational face of trust (who pays, on whose key) and of the routing topology. The architecture bends it with three structures: BYOK removes the platform's token cost and risk entirely (the tenant's own key pays); tiered routing serves each unit of work at the lowest sufficient tier; and the self-hosted floor caps the marginal cost of the high-volume tail. The measured result is the cost dimension answered concretely: a blended cost per call of a fraction of a cent, ~14% of calls at zero marginal cost — one to two orders of magnitude below a uniform-frontier deployment.

Latency is the operational face of observability. The architecture bends it with the event-ledger/projection read model (Principle 3), which holds read latency constant as accumulated state grows without bound — the difference, measured, between a 44.75-second count and a 7-millisecond one, and between a 60-second "offline" page and a 0.2-second one — and with tiered routing, which keeps the frontier's tail latency off the common path. Latency does not degrade with the system's own memory, which is the precondition for a self-observing, self-improving AI system to stay fast as it scales.

Safety is the operational face of trust and governance. The architecture bends it with graded governance (human approval reserved for the risky minority), an immutable floor (an infinite-cost barrier on prohibited actions), and

an evidence substrate that makes every autonomous action provable, reversible, and attributable — shifting oversight from ex-ante approval (which does not scale) to ex-post accountability (which does) for the common case. The honest frontier (§16.2) is grounded risk-tiering; the architecture around it is what lets safety scale with autonomy rather than against it.

The three are not independent, and their coupling is the crux: a naive system improves one at the expense of the others (a cheaper model is less safe; more oversight is slower and costlier; faster reads risk stale governance). The architecture's claim is that the six structures improve all three *together* — BYOK is simultaneously cheaper and safer (no custody honeypot); adjudication is simultaneously more reliable and, by removing the waste escalation, less costly in human time; the projection read model is simultaneously faster and, by keeping governance data queryable, safer. A scalable AI system is one that has resolved the cost–latency–safety trilemma not by choosing a corner but by building the structures that relax all three constraints at once. That, in the enterprise's own language, is what "scaling AI" means, and it is what the architecture of Part II delivers.

---

18. Synthesis: a reference architecture for scalable AI

Seven principles, developed and measured across §9–§16, compose into a single reference architecture — Part II's constructive answer to the question Part I posed. Each targets a specific non-production cost; together they lower (1-*p*) across the board, and the bound 1/(1-*p*) rises with it. Table 5 collects them with the cost each bends, its operational face, and the primary measured evidence.

Table 5 — The reference architecture for scalable AI.

#	Principle	Bends	Operational face	Primary measured evidence
1	Work as governed, criteria-bearing packets, routed to capacity ("Uber for work")	verification, coordination	does it finish; what does it cost	1,213 tasks · 4,131 messages · 178,910 governed envelopes · 31 executors
2	Completion adjudicated against evidence, not self-declared	verification	does it finish	37.6% escalation exposed; adjudicator live; readiness-adjudicator precedent
3	Event ledger + maintained projections; never scan production	observability	latency	≈760× read-model compression; `COUNT` 44.75 s → 7 ms; 60 s → 0.2 s
4	Tiered capability ladder, complexity-routed, pay-your-own-key	trust, coordination	cost, latency	blended cost a fraction of a cent per call, ~1–2 orders of magnitude below uniform-frontier; 7 providers, 13 models used
4a	Make the high-volume tail (self-hosted specialists); rent the head	trust	cost	~14% of calls at zero marginal cost; a family of fine-tuned specialist adapters
5	Graded, evidence-backed governance with an immutable floor	trust	safety	9.4M-row action ledger; AUTO/SOFT/MED/HARD tiers; ex-post accountability
6	The evidence corpus is the learning signal; close the flywheel	all three (over time)	compounding	52.1M-record labelled substrate; conductance loop live; specialist slow-loop

Each row lowers one non-production cost; read down the "operational face" column and the architecture is seen to answer, together, the enterprise's three questions — will it finish (verification), will it be fast and legible (observability/latency), will it be safe and affordable (trust/safety/cost) — which §17 argued must be solved *simultaneously*, and which Principle 6 makes *compound*. The architecture is not ARQERA-specific — ARQERA is one instantiation that let us measure it — and that is the point: **the science of scaling AI is not a larger model but this architecture around the model.** A frontier lab that has solved capability ($k \to \infty$) is, by Proposition 2, bounded by exactly this architecture; the marginal value of its next capability increment is small next to the value of lowering its customers' — and its own — ($1-p$). The path from a capable model to scaled, governed, deployed value runs through the seven principles above, and once the bound is near, the returns to building them dominate the returns to capability alone.

We hold this synthesis to the paper's standard: it is a reference architecture *evidenced by a single instrumented system*, in which each element is measured to lower its target cost, not a proof that these seven are complete or optimal. §20 states what would falsify the claim; §21 proposes the multi-system work that would test whether the architecture generalises as the theory predicts. But the existence proof is, we believe, established: the non-

production costs that bound AI's economy of scale are bendable, the mechanisms that bend them are buildable and measurable, and their composition is the constructive answer to how AI scales.

Part III — Implications

19. Discussion

19.1 For firms adopting AI

Corollary 2.1 is the operative guidance: past a threshold, the marginal enterprise dollar is better spent lowering (1-*p*) than chasing *k*. Concretely, the firm that wins with AI is not the one with the most capable model access — access is increasingly commoditised — but the one that has built the verification, observability, and trust *surfaces* that convert capability into governed value. This reframes "AI strategy" from a procurement question (which model) to a systems question (which surfaces), and predicts that the durable advantage accrues to organisations that treat verification, observability, and trust as first-class engineering investments rather than afterthoughts.

19.2 For AI providers

If delivered value is Amdahl-bounded by (1-*p*), then a provider whose product is only the model captures a shrinking share of the value it enables: the customer's bottleneck is downstream, in the non-production costs the raw model does not address. This is a strategic argument for providers to move up the stack into the verification, observability, and trust layers — not to sell more capability, but to lower the customer's (1-*p*). It also reframes the go-to-market problem: the sale is not the model's benchmark score but the customer's path from capability to governed value, which lives entirely in the non-collapsing terms.

19.3 For the theory of the firm

Coase (1937) asked why firms exist if markets are efficient; the answer was transaction and coordination cost. The human-plus-agent regime lowers the *production* coordination cost toward zero while leaving the *verification* coordination cost intact — suggesting that firms in the AI era organise not around production capacity (abundant) but around verification, observability, and trust capacity (scarce). The boundary of the AI-era firm may be drawn where its ability to *verify and govern* AI output ends, not where its ability to *produce* it does.

This has a concrete corollary for firm scope and make-versus-buy. In the Coasean frame, a firm internalises an activity when the transaction cost of buying it in the market exceeds the coordination cost of doing it inside. AI collapses the *production* side of that comparison for a wide class of activities, which would, taken alone, push firms to internalise more (production is now cheap to do in-house). But the framework predicts the opposite pressure dominates: because the binding cost is now verification/observability/trust, the activities a firm should internalise are precisely those where it has *verification and governance advantage*, and it should *buy* the raw production — including the models themselves — from the market, where it is commoditising. The AI-era firm is thin on production and thick on governance: it outsources the making and internalises the vouching. This reframes vertical integration strategy in the AI era and predicts a specific disaggregation — production layers commoditising toward the market, verification-and-trust layers consolidating inside the firm — that is empirically testable and, we would argue, already visible in the single-builder case, where the human retained the governance work and delegated the production entirely to (external, commoditised) agents.

19.4 The reflexive result and its reach

That the non-production costs bound an AI operator (§6.2) and not only a human is more than a curiosity. It implies the costs are structural — properties of governed systems — and therefore that "more capable agents" do not dissolve them; a superhuman agent operating an un-observable, un-verifiable, un-trusted system is still bounded by 1/(1-*p*). The path to higher AI economies of scale runs through building the surfaces, for any operator, human or model.

19.5 Labour, skills, and the shape of the AI-era job

The decomposition predicts a specific reallocation of human labour, consistent with the task-level automation view (Acemoglu & Restrepo, 2019) but sharper about *which* tasks. If production automates and verification, observability, and trust do not, then the human work that remains — and the human work that is *created* — is disproportionately in the non-production terms: specifying acceptance criteria, adjudicating completion, designing observability surfaces, arranging custody and provenance. This is a testable labour-market prediction: demand should shift from *producers* (who write the code, draft the copy, take the action) toward *verifiers, architects of observability, and stewards of trust* — roles that in software today are diffusely spread across senior engineering, SRE, security, and compliance, and which the framework predicts will consolidate into first-class functions. The single-builder case is a preview: the one human's residual work was overwhelmingly the non-production work — reviewing agent output, designing the seven gates, tracing the observability failures, arranging the trust topology — while the agents did the producing. The AI-era senior contributor is, on this account, defined less by what they can *produce* (agents produce) and more by what they can *verify, observe, and vouch for*.

19.6 Why the productivity paradox is a ceiling, not a lag

The productivity-paradox literature (Brynjolfsson, Rock, & Syverson, 2021) reads the gap between a technology's capability and its measured productivity as a *lag* — the complementary intangible investments take time, but the productivity eventually arrives. Proposition 2 offers a less optimistic reading for the sub-case of governed AI value: absent the structural investment that lowers (1-*p*), the gap is not a lag that time closes but a *ceiling* that time cannot. Waiting for more capable models (*k*) does not help once the bound is near (Table 4). The paradox resolves only through the specific complementary investments the framework names — verification, observability, and trust surfaces — and a firm that mistakes the ceiling for a lag will wait indefinitely for a productivity gain that its architecture forbids. This is, we think, the most consequential practical claim of the paper: **the returns to AI are gated by an investment most organisations are not making, because they have mistaken an architectural ceiling for a temporary lag.**

19.7 Falsifiable predictions

A framework earns its keep by risking refutation. The theory makes the following predictions, each of which would, if violated, weaken or overturn it:

1. **The vanishing-returns prediction (Proposition 2).** Across firms with comparable non-production architecture, increasing model capability (*k*) beyond a moderate threshold should produce diminishing gains in *delivered governed value*, not merely in benchmark capability. A regime in which delivered value continued to scale roughly linearly with model capability, holding architecture fixed, would falsify the bound.
2. **The dominance-of-*p* prediction (Table 4).** Between two organisations with the *same* model access, delivered-value throughput should be predicted better by their non-production cost fraction (1-*p*) than by any

model-quality difference. If model quality dominated (1-*p*) in explaining throughput differences, the framework's central claim — that the row matters more than the column — would be wrong.

3. **The verification-rises-with-production prediction (§3.5).** Teams that increase production rate via AI without co-investing in verification should exhibit *rising*, not falling, verification load (escalation rates, review backlogs, defect-escape rates) — the opposite of the naive expectation that faster production means faster delivery. A consistent finding that verification load fell automatically as AI production rose would refute the interaction argument.
4. **The observability-diseconomy prediction (Proposition 3).** Systems that scale AI-produced state without scaling observability architecture should exhibit observability cost (diagnostic latency, mean-time-to-understand) *rising with cumulative state*. Flat observability cost as state grew, absent explicit projection/index investment, would falsify Proposition 3.
5. **The firm-disaggregation prediction (§19.3).** Over time, AI-era firms should be observed *outsourcing production* (including model provision) toward a commoditising market while *internalising verification and trust* functions. A trend toward internalising production and outsourcing governance would contradict the make-versus-buy corollary.

These are, deliberately, predictions about *populations and trends*, which the single-case design of this paper cannot itself test — a point we make not to excuse the limitation but to specify exactly what the multi-case programme of §21 must measure. The theory is offered as falsifiable, and the case study as an existence proof that the phenomena it predicts are real and measurable in at least one fully-instrumented instance.

19.8 For a frontier capability leader

The argument has a specific and, we think, counter-intuitive implication for an organisation whose core asset is frontier model capability. Such an organisation is, by construction, the party for whom *k* is largest — and therefore, by Proposition 2 and Table 4, the party for whom the *marginal* return to further *k* is smallest at the level of *delivered enterprise value*. This is not an argument against advancing capability, which has returns along many other dimensions the four-term model does not capture; it is an argument about where the *next unit of delivered enterprise value* is cheapest to obtain. For a capability leader selling to enterprises, that unit is overwhelmingly on the non-production side: the customer's bottleneck is not the model's benchmark but the customer's (1-*p*) — their inability to verify, observe, and trust what the model produces in their operations. The strategic corollary is that the highest-leverage move for a capability leader is to move *down the customer's (1-p)*, by providing — or acquiring — the verification, observability, and trust architecture of Part II, so that the customer converts more of the already-abundant capability into governed value.

This reframes both product and go-to-market. On product: the architecture of §8–§18 is a roadmap for what a capability leader might build *around* its models to capture the value they enable but do not themselves deliver — work packets and dispatch, completion adjudication, the projection read model, tiered routing, graded governance, the evidence flywheel. On go-to-market: the enterprise sale is not the model's capability (which the customer can increasingly assume) but the *path from that capability to governed value*, which lives entirely in the non-production architecture — so the party who can speak credibly to that path, and build it with the customer, captures the account. The paper's own existence is a small instance of the point: an instrumented instantiation of exactly this architecture, built and measured independently, is more informative about how to scale enterprise AI than a further increment of model capability would be — because the increment is not where the bound binds.

We note, for completeness and candour, that an instantiation of this architecture is itself an asset a capability leader could build or acquire rather than re-derive; the paper takes no position on that beyond observing that the architecture, not any particular instantiation, is the transferable science.

20. Threats to validity

We treat these as first-order, per §4.3.

External validity (generalisation). The gravest threat: $n = 1$. ARQERA may be idiosyncratic in its domain (governance-first AI operations), its builder, or its choices; the specific magnitudes (p, k, the escalation rate) almost certainly do not generalise as point estimates. Our defence is scope: we claim an existence proof, measured mechanisms, and a framework, not a population distribution. The framework's value is that it is *testable* on other cases (§21); the paper's numbers are grounding, not generalisation.

Construct validity (do the measures capture the constructs?). The four cost terms are asserted, not derived from a validated instrument; the fix-to-feature ratio is a proxy for maintainability sensitive to commit-message convention and gameable; endpoint and line counts measure surface, not value or quality; the escalation rate measures *routed-to-human*, which conflates genuine verification need with mis-calibrated over-escalation (a conflation we flag but do not fully separate). The production collapse factor k is not point-estimated because its baseline (person-years for 650k lines) rests on contested effort models.

Internal validity (are the causal claims sound?). The claim that the *gates* caused the fix-to-feature recovery (§7.2) is argued from timing and incident-to-gate mapping, not isolated experimentally; maturation (the codebase and builder improving) is an unaddressed alternative explanation. The performance figures (Table 3) are single-run, mechanism-demonstrating measurements on one tenant's data distribution, not population averages, and should not be read as expected speedups for arbitrary workloads.

Reliability and researcher position. The author built and operates the system, and part of the data was gathered by AI agents under the author's direction — a source of both privileged access and bias. We have countered the latter by grounding every figure in a live measurement, by re-verifying the headline numbers, by treating the measurement harness as suspect (§6.2), and by auditing our own framework against itself (§7.3), but the position is real and the reader should weight it.

A note on the Amdahl analogy. Amdahl's law assumes a fixed serial fraction; our $(1-p)$ is explicitly *not* fixed (it is bendable, §7.2), so the bound is a bound *at a given level of structural investment*, not an absolute constant. This is a feature — it is where strategy enters — but it means Proposition 2 should be read as a family of bounds indexed by structural investment, not a single ceiling.

---

21. Conclusion and future work

We set out to ask what scales in the AI era, and — because a diagnosis without a cure is only half a science — how to scale it. The diagnosis, from a fully instrumented single-builder system, is a sharp asymmetry: **production collapses; verification, observability, and trust do not.** Formally, the maximum AI economy of scale in delivering *governed* value is Amdahl-bounded by the reciprocal of the non-production cost fraction, independent of model capability — so that a hundredfold cheaper production yields a low-single-digit system-level multiple when most cost was never production. We measured each non-collapsing term directly — a 37.6%

human-verification load, an observability cost that grew adversarially with a 52-million-record substrate and was remediated for up to ~6,000×, and a trust topology in which one expired platform credential disabled every customer — and we showed the costs are partially bendable, with a fix-to-feature ratio driven from 2.1 : 1 to 0.88 : 1 by converting discipline into automated gates.

The cure is the reference architecture of Part II: seven measured, composable structures — governed work packets routed to capacity, completion adjudicated against evidence, an event-ledger-and-projection read model, tiered routing with a self-hosted floor and pay-your-own-key, graded evidence-backed governance, and an evidence flywheel — that between them lower every non-production cost, and in the enterprise's own language answer cost (blended inference a fraction of a cent per call, one to two orders of magnitude below a uniform-frontier deployment), latency (a 44.75-second count made 7 milliseconds), and safety (a 9.4-million-row accountability substrate under graded governance) *together*. We also reported the negative results — self-declared completion, heuristic risk-tiering, credential-proxy convenience layers, ungrounded formalism, and partial deployment each failed silently at scale — because the failures that bound the non-production costs are quiet mis-behaviours, not loud crashes, and naming them is part of the science.

The strategic reading is consistent across firms, providers, capability leaders, and the theory of the firm: **in the AI era, capability is abundant and bounded in its returns; advantage accrues to whoever most lowers the cost of verifying, observing, and trusting what the AI produces — and that lowering is an architecture, not a model.** For the frontier lab whose k is largest, this is the most actionable claim in the paper: the next unit of delivered enterprise value is cheapest on the non-production side, and the architecture of Part II is the map to it.

Future work should (i) replicate the decomposition across multiple systems and builders to test external validity and estimate a distribution of p; (ii) develop a validated instrument for the four cost terms, separating genuine verification need from mis-calibrated escalation; (iii) measure, rather than design, the effect of external adjudication on C_v and of the evidence flywheel on $(1-p)$ over time — the compounding claim of §14 that a single snapshot cannot establish; (iv) attack the open frontier of grounded risk-tiering (§16.2), on which the scalability of safety turns; and (v) formalise the *bendability* of $(1-p)$ as an investment function, turning Proposition 2's family of bounds into a normative theory of how much architectural investment a given level of model capability warrants. The single case establishes that the phenomena are real and measurable and that the architecture that addresses them is buildable; the economics of AI at scale will be written in the non-production terms, and won in the architecture that bends them.

Future work should (i) replicate the decomposition across multiple systems and builders to test external validity and estimate a distribution of p; (ii) develop a validated instrument for the four cost terms, separating genuine verification need from mis-calibrated escalation; (iii) measure, rather than design, the effect of external adjudication on C_v; and (iv) formalise the *bendability* of $(1-p)$ as an investment function, turning Proposition 2's family of bounds into a normative theory of how much structural investment a given level of model capability warrants. The single case establishes that the phenomenon is real and measurable; the economics of AI at scale will be written in the non-production terms.

References

Acemoglu, D., & Restrepo, P. (2019). Automation and new tasks: How technology displaces and reinstates labor. *Journal of Economic Perspectives*, 33(2), 3–30.

Agrawal, A., Gans, J., & Goldfarb, A. (2018). *Prediction Machines: The Simple Economics of Artificial Intelligence*. Harvard Business Review Press.

Amdahl, G. M. (1967). Validity of the single processor approach to achieving large scale computing capabilities. *AFIPS Conference Proceedings*, 30, 483–485.

Beyer, B., Jones, C., Petoff, J., & Murphy, N. R. (2016). *Site Reliability Engineering: How Google Runs Production Systems*. O'Reilly.

Brynjolfsson, E., & McAfee, A. (2014). *The Second Machine Age: Work, Progress, and Prosperity in a Time of Brilliant Technologies*. W. W. Norton.

Brynjolfsson, E., Rock, D., & Syverson, C. (2021). The productivity J-curve: How intangibles complement general purpose technologies. *American Economic Journal: Macroeconomics*, 13(1), 333–372.

Boehm, B. W. (1981). *Software Engineering Economics*. Prentice Hall.

Brooks, F. P. (1975). *The Mythical Man-Month: Essays on Software Engineering*. Addison-Wesley.

Chandler, A. D. (1990). *Scale and Scope: The Dynamics of Industrial Capitalism*. Harvard University Press.

Coase, R. H. (1937). The nature of the firm. *Economica*, 4(16), 386–405.

Hoffmann, J., Borgeaud, S., Mensch, A., et al. (2022). Training compute-optimal large language models (Chinchilla). *arXiv:2203.15556*.

Jensen, M. C., & Meckling, W. H. (1976). Theory of the firm: Managerial behavior, agency costs and ownership structure. *Journal of Financial Economics*, 3(4), 305–360.

Kaplan, J., McCandlish, S., Henighan, T., et al. (2020). Scaling laws for neural language models. *arXiv:2001.08361*.

Kephart, J. O., & Chess, D. M. (2003). The vision of autonomic computing. *IEEE Computer*, 36(1), 41–50.

Marshall, A. (1890). *Principles of Economics*. Macmillan.

Williamson, O. E. (1985). *The Economic Institutions of Capitalism*. Free Press.

Yao, S., Zhao, J., Yu, D., et al. (2023). ReAct: Synergizing reasoning and acting in language models. *ICLR 2023*.

Yin, R. K. (2018). *Case Study Research and Applications: Design and Methods* (6th ed.). SAGE.

Zheng, L., Chiang, W.-L., Sheng, Y., et al. (2023). Judging LLM-as-a-judge with MT-Bench and Chatbot Arena. *NeurIPS 2023 Datasets and Benchmarks*.

Appendix A — Measured figures and provenance

All figures measured on the live production system, 23 July 2026, and independently re-verified.

#	Figure	Value	Source
A1	API endpoints	2,688	source-tree route census
A2	API routers / service modules / data models	402 / 805 / 170	source-tree census
A3	Backend lines of code	653,701	line count, application tree
A4	Frontend modules	1,440	source-tree census
A5	Total commits	4,742	version-control log
A6	Feature / fix commits	1,842 / 1,616	commit-prefix classification
A7	**Fix-to-feature ratio**	**0.88 : 1** (historical window 2.1 : 1)	derived from A6
A8	Build window	215 days (2025-12-20 → 2026-07-23)	version-control log
A9	Total substrate records (all tables)	52,128,738	database aggregate (system access)
A10	Primary evidence ledger	11,050,873 rows / 17 GB	database aggregate
A11	Distinct action types / span / rate	252 / 174 days / ~64,000 per day	database aggregate
A12	Autonomous runs (total)	237	run-outcome table
A13	— succeeded / **escalated** / failed / blocked	102 / **89** / 44 / 2	run-outcome table
A14	**Escalation rate**	**37.6%** (historical sample 80%+)	derived from A13
A15	BYOK provider configurations	138	configuration table
A16	Providers in routing registry / with live adapters	23 / 18	provider registry
A17	Chain-verify remediation	45 s → 2.3 s (≈19×)	single-run latency
A18	`OFFSET` → index cutoff	16.3 s → 0.10 s (163×)	single-run latency
A19	Public endpoints	60 s+ → 0.1–0.2 s (≈300×)	single-run latency
A20	Tenant `COUNT` → projection	44.75 s → 7 ms (≈6,000×)	single-run latency
A21	Request-path substrate scans found	17	source audit
A22	Self-observability defects (one session)	9	operating-session census
A23	RLS-masked vs true membership count	"0" → 156	measurement-harness finding (§6.2)

Appendix B — Reproducibility notes

The production-scale figures (A1–A8) are reproducible from the source tree and version-control history by re-running the census and prefix-classification commands. The substrate figures (A9–A16) are reproducible by the same aggregate database queries, executed with system-level read access; a subtlety (A23) is that queries run without the appropriate row-security context return masked counts — the true totals require an explicit system bypass, and any replication must control for this or it will silently under-count. The latency figures (A17–A20) are single-run measurements of specific query patterns against the production data distribution and are intended to demonstrate the mechanism (scan-and-discard versus index/projection reads); they are not benchmarked averages and will vary with data distribution and cache state. The run-outcome distribution (A12–A14) is a snapshot of a modest sample (n = 237) and should be re-measured on a larger sample before being treated as a stable rate.