Agentic AI Systems Should Be Designed as Marginal Token Allocators

Siqi Zhu
University of Illinois Urbana-Champaign
siqizhu4@illinois.edu

Abstract

This position paper argues that agentic AI systems should be designed and evaluated as *marginal token allocation economies* rather than as text generators priced by the unit. We follow a single request—a developer asking a coding agent to fix a failing test—through four economic layers that today are designed in isolation: a router that decides which model answers, an agent that decides whether to plan, act, verify, or defer, a serving stack that decides how to produce each token, and a training pipeline that decides whether the trace is worth learning from. We show that all four layers are solving the *same* first-order condition—marginal benefit equals marginal cost plus latency cost plus risk cost—with different index sets and different prices. The framing is deliberately minimal: we do not propose a complete theory of AI economics. But adopting marginal token allocation as the shared accounting object explains why systems that locally minimize tokens globally misallocate them, predicts a small set of recurring failure modes (over-routing, over-delegation, under-verification, serving congestion, stale rollouts, cache misuse), and points to a concrete research agenda in token-aware evaluation, autonomy pricing, congestion-priced serving, and risk-adjusted RL budgeting.

1 Introduction

Consider a developer who types "the CI test on `auth/login` is failing—fix it" into a modern coding agent. Before a single line of code is touched, the system has already made four economic decisions. A *router* decides whether to spend a cheap model (fast triage, possibly wrong) or a frontier model (slow, expensive, more likely correct) [8, 30]. An *agent policy* decides how the chosen model should spend its tokens—reading the repository, planning, editing, running tests, or asking the developer to clarify [45, 39, 44]. A *serving stack* decides how to produce those tokens, juggling prefill for the long context, decode for the patch, and KV cache for the test logs [20, 34, 46, 13]. And a *training pipeline* decides, after the dust settles, whether this trace is worth learning from—rollout, verifier, or update tokens to spend now for capability later [32, 11, 7, 47].

Each layer charges a different price for what looks, on the API invoice, like the same token. The router prices a token in dollars per million; the agent prices it in expected risk of an irreversible action; the serving stack prices it in queueing delay; the trainer prices it in marginal capability gain over a discount horizon. This decoupling is hidden by the dominant accounting fiction—tokens are units of text, billed at a flat rate [6]. That fiction was workable when LLMs were chat completions. It is misleading once tokens cause actions, occupy infrastructure, and become training data.

This position paper argues that *agentic AI systems should be designed and evaluated as marginal token allocation economies*, in which routers, agents, serving schedulers, and trainers are mechanisms that decide where the next unit of tokenized computation should be spent under joint quality, cost, latency, and risk constraints. The claim is narrower than "token economics is a complete theory of AI" and stronger than "tokens are billed by the unit." We argue that a *single* first-order condition—marginal

Preprint.

benefit equals marginal cost plus latency cost plus risk cost—is the right minimum vocabulary, because the four layers above are not parallel engineering problems but vertical slices of one allocation problem. The router screens the demand side, the agent contracts on the action side, the serving stack produces on the supply side, and the trainer accumulates capital on the investment side. They are the same equation, evaluated at four shadow prices that today no single layer can see.

The central tension. Each layer optimizes locally and competently. Routers minimize cost subject to quality [8]; agents maximize success rate [25]; serving stacks maximize throughput [1]; trainers maximize evaluation score [11]. Yet local rationality aggregates into global misallocation: an aggressive router downgrades a high-stakes request, the agent compensates by burning extra verification tokens, the serving stack queues those verifier calls behind unrelated long-context traffic, and the trainer learns from a noisy trace that will not generalize. The pattern is the textbook problem of unpriced externalities [35, 9], transposed to token economies. Marginal token allocation is the shared price language that lets the four layers cooperate rather than merely stack.

Contributions. (i) We formulate a single optimality condition—marginal token allocation—and show that routers, agents, serving stacks, and trainers are instances of it (Section 2). (ii) We trace one request through all four layers, using standard tools from microeconomics: screening with hidden types [3, 41], principal–agent contracts [29, 16], multi-stage production with congestion [35, 43], and capital accumulation [40] (Section 3). (iii) We show that recurring failures across the stack—over-routing, over-delegation, under-verification, congestion, stale rollouts, cache misuse—are corner cases of the same equation when one of the four prices is mis-set (Section 4). (iv) We address principled objections (Section 5), discuss design implications, limitations, and an open research agenda (Section 6), and conclude (Section 7). Throughout, our objective is not to summarize the literature but to defend a single design stance.

2 One Equation, Four Prices

The primitive object. Let an LLM *system* face a stream of tasks. For each task it has a finite set of *token uses*, indexed by i, between which it must allocate computation. Concretely, i ranges over choices such as {cheap model, frontier model, retrieval, planning, tool call, verifier, prefill capacity, decode capacity, KV transfer, RL rollout, reward computation, gradient update}. Each use i has a marginal quality contribution ΔQ_i, a marginal compute cost ΔC_i, a marginal latency cost ΔL_i, and a marginal risk ΔR_i (e.g., probability of a wrong action weighted by its consequence). Let V denote task value. The system should spend the next token on

$$i^* \;=\; \arg\max_i \;\Big[\, V\,\Delta Q_i \;-\; \Delta C_i \;-\; \lambda\,\Delta L_i \;-\; \rho\,\Delta R_i \,\Big], \tag{1}$$

where $\lambda \geq 0$ and $\rho \geq 0$ are user- or operator-specific shadow prices on latency and risk. Equation 1 is the standard marginal-utility decision rule of microeconomics [28] transposed to tokenized computation. At an interior optimum, the Marshallian equimarginal condition holds:

$$V\,\Delta Q_i \;-\; \lambda\,\Delta L_i \;-\; \rho\,\Delta R_i \;=\; \Delta C_i \quad \forall\, i \in \mathcal{A}^*, \tag{2}$$

where $\mathcal{A}^*$ is the set of token uses with strictly positive allocation. "The marginal benefit of a token equals its full marginal cost," once latency and risk are properly priced.

Why the four prices live at four layers. Equation 1 packs the entire stack into one expression, but its terms are observed at different layers (Table 1). V is set by the user, who alone knows the value of her task; ΔC_i is set by the operator, who runs the GPUs; λ is set by the SLA, which arbitrates queueing; ρ is set by the safety team, which absorbs the consequences of wrong actions. No single layer sees all four. This is the structural reason why locally rational decisions compose into globally irrational allocations [42], and why a shared accounting object is needed.

Why "marginal" rather than "total". Industry dashboards typically report total or average token cost. But a 30% reduction in total tokens that comes from cutting verifier tokens may *raise* risk-adjusted cost, because the cost of an unverified wrong action exceeds the savings. Marginal analysis makes this explicit: the right object is $\partial U/\partial t_i$, not $U/\sum_i t_i$. We will see this gap is precisely where current systems misallocate.

Table 1: The same allocation primitive is observed at four organizational layers, each of which sees only one or two of the four prices in Equation 1. Marginal token allocation is the language that makes the four layers commensurable.

Layer	Mechanism	Index i	Price observed	Paragraph
Demand	Routing as screening	model tier	V, ΔC_i	§3.1
Action	Agent as principal–agent	plan/act/verify	ρ, V	§3.2
Supply	Serving as production	prefill/decode/KV	λ, ΔC_i	§3.3
Capital	Caches & RL as investment	rollout/store	ΔC_i, ρ	§3.4

A worked example. A small numerical instance of Equation 1 clarifies the stakes. Suppose two models are available: a cheap one with quality $q_c = 0.7$ at cost $c_c = 1$, and a frontier one with $q_f = 0.9$ at $c_f = 5$. For a low-value task ($V = 10$), surplus is $0.7 \cdot 10 - 1 = 6$ versus $0.9 \cdot 10 - 5 = 4$, so cheap wins. For a high-value task ($V = 100$), surpluses are 69 versus 85, and frontier wins. The crossover is at $V^* = (c_f - c_c)/(q_f - q_c) = 20$. Now add risk: if the cheap model has a wrong-action probability $r_c = 0.05$ versus $r_f = 0.01$ and risk price $\rho = 50$, the cheap-versus-frontier surplus gap shrinks by $\rho(r_c - r_f) = 50 \cdot 0.04 = 2$, shifting V^* to ≈ 10. A small change in one term in Equation 1 flips the optimal allocation. This is why marginal analysis is non-trivial in practice: each layer adjusts a different term, and small shifts compound across layers.

A closed-form sanity check. A Cobb–Douglas instance $Q(\mathbf{t}) = A \prod_i t_i^{\alpha_i}$ subject to $\sum_i p_i t_i \le B$, with full shadow price $p_i = \Delta C_i + \lambda \Delta L_i + \rho \Delta R_i$, has the textbook solution $t_i^* = \frac{\alpha_i}{\sum_j \alpha_j} \cdot \frac{B}{p_i}$ [28]. Three operational facts follow: irrelevant uses ($\alpha_i = 0$) consume zero tokens regardless of price; rising p_i proportionally squeezes use i, the substitution pattern observed in production schedulers [1]; and complements cannot be cut to zero without driving Q to zero—which is why "minimize tokens" fails when reading and verification complement editing.

Prices as Lagrange multipliers, and the welfare-theorem prescription. The four prices in Equation 1 are not chosen by fiat; they are the dual variables of the constrained primal of token allocation. Consider a system maximizing $\sum_x V(x)\, Q(\mathbf{t}_x)$ subject to a compute budget, a latency SLA, and a risk envelope. The Lagrangian

$$\mathcal{L} = \sum_x V(x)\, Q(\mathbf{t}_x) - \mu_C \left(\sum_{x,i} \Delta C_i\, t_{i,x} - \bar{C} \right) - \mu_L \left(\sum_{x,i} \Delta L_i\, t_{i,x} - \bar{L} \right) - \mu_R \left(\sum_{x,i} \Delta R_i\, t_{i,x} - \bar{R} \right) \tag{3}$$

yields KKT stationarity $V(x)\, \partial Q / \partial t_{i,x} = \mu_C \Delta C_i + \mu_L \Delta L_i + \mu_R \Delta R_i$ at the optimum, which is exactly Equation 1 with $(1, \lambda, \rho) = (\mu_C, \mu_L/\mu_C, \mu_R/\mu_C)$. Three implications follow. First, the prices are *endogenous*: determined by the binding constraints, not chosen a priori. Second, they obey complementary slackness, so a system whose latency SLA slacks should drive $\lambda \to 0$ rather than pin it to a constant—the standard production-stack practice. Third, by the first welfare theorem [28], if router, agent, serving stack, and trainer all maximize their own component of $\mathcal{L}$ taking the same (μ_C, μ_L, μ_R) as given, the resulting allocation is Pareto efficient: no reallocation of tokens across layers improves any payoff without hurting another's. The second welfare theorem implies that any efficient allocation can be sustained by some price vector. Together they yield a sharp design prescription: the question is not whether to centralize allocation but whether the four layers see a common, complete price vector. They almost never do today.

Information rents and the screening cost of routing. The router is not just choosing; it is screening. A type-θ user knows her own V but the router does not. Mechanism design [29, 21] then implies that the cost of truthful self-selection is an *information rent* paid to high-value types: with type distribution $F(\theta)$ and the increasing-hazard property, the optimal menu prices the marginal type at virtual valuation $V(\theta) - \frac{1-F(\theta)}{f(\theta)}\, V'(\theta)$ rather than at $V(\theta)$. The wedge $\frac{1-F}{f}\, V'$ does not appear in any naive cost–quality dashboard. Two empirical implications follow. Even an optimally designed router will downgrade a non-trivial fraction of high-V requests on purpose: the rent is the price of incentive compatibility, not a bug. And the rent grows in user heterogeneity, which is why a router tuned on a uniform benchmark systematically fails on long-tail traffic.

General equilibrium across tenants. A single request faces four prices, but multi-tenant deployments must clear them simultaneously. A competitive equilibrium [28, 42] is a price vector $\mathbf{p}^*$ such that each tenant's demand $\mathbf{z}_x(\mathbf{p}^*)$ solves its own Equation 1, the operator's supply maximizes profit given the production frontier, and markets clear, $\sum_x \mathbf{z}_x(\mathbf{p}^*) = \mathbf{Y}(\mathbf{p}^*)$. The first welfare theorem then guarantees that the equilibrium allocation is Pareto efficient *across tenants*, internalizing the queueing externality that flat per-token pricing cannot. The closest production analogues are priority queues with admission control [1, 13], equivalent to a degenerate equilibrium in which only one constraint is priced.

The Knightian limit of ρ. A separate caveat applies to $\rho\Delta R_i$ itself: it captures expected-value risk, but agentic actions are often novel and their consequence distribution unknown—a Knightian regime [19]. The framework is not changed, but the functional form of $\rho\Delta R_i$ should switch from expectation to a coherent risk measure (e.g., CVaR or max-min over an ambiguity set) on rare, high-consequence actions.

What the theory does *not* claim. We deliberately stop at a first-order condition. Equation 1 is a one-step rule; we do not claim that summing it across tasks gives a complete macroeconomic theory of AI, nor that token allocation is the only relevant economic primitive (data, energy, and labor matter too). We use marginalism as a *lens*: it should produce sharp predictions for system design and identify shared structure across what otherwise look like unrelated engineering problems.

3 One Request, Four Layers

We now follow the developer's request from §1 through each layer of the stack and show that each layer is solving Equation 1 at a different price. The narrative deliberately preserves the request's identity: the same task picks up new prices as it descends.

3.1 Demand: Routing as a Screening Mechanism

The first decision is which model answers. Naively one would route by "best quality per dollar." That intuition is wrong in the same way that posting a single price is wrong in a market with heterogeneous buyers [42].

A request has a hidden type $\theta = (V, d, r, \lambda)$: task value, difficulty, risk sensitivity, latency sensitivity. The router observes only x, a noisy signal of θ. Its problem is the screening problem of Spence [41] and Mirrlees [29]: design a mapping $m^*(x)$ such that the chosen model maximizes Equation 1 restricted to the model index,

$$m^*(x) \;=\; \arg\max_{m \in \mathcal{M}} \Big[\, V(x)\, \widehat{q}_m(x) \;-\; c_m \;-\; \lambda\, l_m(x) \;-\; \rho\, r_m(x) \Big]. \tag{4}$$

Recent routers [8, 30, 17] estimate $\widehat{q}_m(x)$ from preference data or cascades. That estimation is doing economic work: it converts a flat "model market" into a differentiated market in which each request is matched to the cheapest model that preserves expected utility. Akerlof [3] showed that hidden quality on the seller side can collapse a market; routing exposes the symmetric problem on the buyer side, where hidden *difficulty* causes mis-matched models. Both directions are observed in production [30, 17].

In our running example, the router must guess whether the failing-test query is shallow (cheap-model territory) or deep (frontier territory) from a few hundred characters of prompt. If it guesses shallow and the bug is a subtle race condition, the agent will burn tokens later trying to compensate, the developer will eventually re-issue the request to a stronger model, and the system will pay for both attempts. If it guesses deep and the bug is a forgotten import, the operator overpays by a factor of five (using our worked numbers from §2). The cost of routing error is therefore not symmetric: the misallocation propagates downstream and is amortized by every layer below. Strategic users know this, and a sophisticated developer can perturb x to obtain a stronger model—an LLM analogue of Spence's costly signaling. A revenue-equivalent design would charge a premium for higher tiers and let users self-select via an incentive-compatible menu [42],

$$V_k\, q_{m_k} - p_k \;\geq\; V_k\, q_{m_{k'}} - p_{k'}, \qquad \forall\, k, k'. \tag{5}$$

Few production routers do this; most attempt to infer θ silently.

Position. Routers should be evaluated not by accuracy or cost alone but by *regret* relative to Equation 4: the gap between the chosen model's risk-adjusted utility and the ex-post optimal model. They should publish either the regret bound or the menu (Eq. 5). Existing benchmarks [17] approximate the former only loosely and almost never report risk components.

3.2 Action: Agents as Principal–Agent Contracts

The chosen model now enters the agent loop. The router has answered "which model"; the agent must answer "what should it do." This is where the same token's price changes again, because tokens used to summarize a file and tokens used to commit a patch carry different consequences [45, 37, 44].

The autonomy contract. Let $a \in [0, 1]$ denote autonomy (0 = always ask, 1 = act freely) and t be the token budget. The user's expected utility is

$$U(a, t) \;=\; V\, p(a, t) \;-\; C(t) \;-\; R(a, t) \;-\; H(a), \tag{6}$$

where $p(a, t)$ is success probability, $C(t)$ is the token cost, $R(a, t)$ is the expected loss from autonomous mistakes, and $H(a)$ is the human-oversight cost (decreasing in a). The interior optimality condition is the principal–agent first-order condition [29, 16, 21]:

$$V\,\frac{\partial p}{\partial a} \;=\; \frac{\partial R}{\partial a} \;+\; \frac{\partial H}{\partial a}. \tag{7}$$

Autonomy expands until the marginal value of saved human labor equals the marginal increase in risk plus the marginal change in oversight cost. Because $\partial R/\partial a$ is heavily right-skewed—small probability of a catastrophic action—risk-neutral budgeting badly under-prices autonomy.

Token allocation *within* the agent. Once a is set, the agent still has a team-production problem [4]. In our example, the agent must split tokens among reading the repo (T_r), planning the patch (T_p), editing (T_e), and running the test (T_v):

$$Y \;=\; F(T_r, T_p, T_e, T_v, H_{\text{review}}). \tag{8}$$

At the optimum, marginal products are equalized: $\partial Y/\partial T_r = \partial Y/\partial T_p = \partial Y/\partial T_e = \partial Y/\partial T_v$. This contradicts the heuristic of "minimize tokens." Reading and verification tokens are *complements* to editing tokens [39, 26, 24, 14]: the marginal product of an edit token is small without context and verification, and the marginal product of additional reasoning tokens is itself task-dependent—a fact that signals such as model certainty [14] can be used to estimate online. Empirically, agents that skimp on T_v produce cheaper but lower-quality patches and shift cost downstream onto H_{review}. The team-production view also explains why imitating only the editing step from a strong model rarely transfers: the chain of complements upstream and downstream is what produces Y, and a partial copy is not Pareto-improving.

Reversibility and option value. Because action risk is partly irreversible, autonomy decisions also carry option value, in the sense of the real-options literature [12]. Asking the user for confirmation preserves the option to act later; acting immediately destroys it. Equation 1 should therefore include an additional term $\rho_{\text{irr}}\Delta R_i^{\text{irr}}$ for the unrecoverable component of risk. This is why "read" and "draft" tokens flow freely while "commit" and "send" deserve a discrete oversight check.

Position. Agentic systems should publish an *autonomy schedule*—a mapping from action class to required confirmation level (read $\rightarrow$ free, draft $\rightarrow$ free, commit $\rightarrow$ confirm, deploy/transfer $\rightarrow$ multi-party). It is the LLM equivalent of an authorization matrix and is the natural artifact of Equation 7. Current agentic benchmarks [25] measure success rate but rarely measure $R(a, t)$, which we argue is the binding economic constraint.

3.3 Supply: Serving as Production

Each token the agent commands must be physically produced. Modern stacks separate prefill and decode [34, 46], page KV cache [20], and chunk requests [1]. Speculative decoding [22] adds a verifier stage. These are exactly the moves a microeconomic theorist would predict in a multi-stage production system with heterogeneous resources [4].

Let G_p, G_d, K, N denote prefill GPU capacity, decode GPU capacity, KV-cache storage/bandwidth, and interconnect bandwidth. Token output is $Y_{\text{tok}} = F(G_p, G_d, K, N)$ and latency $L = L_p(G_p) + L_d(G_d) + L_K(K, N)$. The cost-minimizing producer satisfies the equimarginal condition,

$$\frac{\partial L/\partial G_p}{\partial C/\partial G_p} \;=\; \frac{\partial L/\partial G_d}{\partial C/\partial G_d} \;=\; \frac{\partial L/\partial K}{\partial C/\partial K}, \tag{9}$$

i.e., latency reduction per dollar should be equalized across resources. Patel et al. [34] and Zhong et al. [46] show empirically that pre-disaggregation systems were systematically off this frontier.

In our example, the agent's plan-then-edit-then-test loop stresses the supply layer in characteristic patterns. Reading the repo is prefill-heavy. Generating the patch is decode-heavy. Running the test produces a long error log—prefill again. None of these is the same token, economically. A request that occupies a long-context KV cache imposes a queueing externality on every other tenant—a textbook congestion externality [35, 43]. The first-best policy is congestion pricing: charge each request the marginal external delay it imposes. Most production APIs charge a flat per-token rate, which under-prices long-context, decode-heavy traffic and over-prices short prompts. Recent schedulers that learn to rank requests by predicted output length [13] are an early step in this direction: they transform an unpriced FCFS queue into something closer to a priority discipline that internalizes the queueing externality, even if the implied prices are not surfaced to the upstream router or agent.

The serving layer also reveals why the previous two layers' decisions cannot be evaluated in isolation. The router that selected the frontier model at the demand layer has implicitly committed the supply layer to higher prefill cost; the agent that chose "read the whole repo before planning" has implicitly committed it to higher KV-cache pressure. If the supply layer's prices ΔC_i are not visible upstream, the demand and action layers will optimize as if compute were free, and the supply layer will absorb the externality as queueing delay. This is the operational mechanism by which *one* layer's local optimum becomes *another* layer's congestion problem.

Speculative decoding as outsourced labor. Speculative decoding is a make-or-buy decision [9]: a cheap draft model produces candidate tokens that the expensive model verifies. The arrangement is profitable when the verifier's marginal cost of accepting a draft is strictly less than its marginal cost of generating from scratch. The acceptance rate α plays the role of an internal transfer price; small drops in α flip the make-or-buy calculus. Adversarially long contexts—where α falls—should disable speculation rather than merely slow it. This is the textbook Coasean prediction: integration dominates the market when transaction costs are high.

Position. Serving systems should expose, log, and ideally bill against *shadow prices* for prefill, decode, and KV resources. These shadow prices are the operational manifestation of ΔC_i in Equation 1 and are a prerequisite for upstream layers (the router and the agent) to make correct decisions.

3.4 Capital: Caches and RL Training as Investment

After the developer's test passes, two streams of tokens persist. The KV blocks for the repo prefix and the test logs may be cached for the next request; the trace itself may be added to the next post-training run. Both are *capital*—past tokens that lower the marginal cost or raise the marginal quality of future tokens.

Caches and memory as inventory. Let S_t denote the stock of cached or memorized content (KV blocks [20], retrieval embeddings [23], or agent notes [33]). Its dynamics are

$$S_{t+1} \;=\; (1 - \delta_S)\, S_t \;+\; I_t, \tag{10}$$

where I_t is investment in new cache writes and δ_S captures distribution drift, schema change, and stale knowledge. The optimal-investment rule equates the marginal cost of writing to the discounted expected savings on future inference. Most production systems implement S_t but rarely measure δ_S, so cache hit rate is reported as an accounting metric rather than an economic one. Reusing a cached prefix when the new task value $V(x')$ differs from the original $V(x)$ is a quality externality; the correction is to track provenance and reuse only when expected reuse value clears an explicit threshold derived from Equation 1.

RL post-training as token investment. Reasoning-oriented post-training [32, 5, 11, 31] consumes tokens that no end-user reads: rollouts, reward computations, KL-regularized updates [38, 36, 2]. These tokens are not consumption but *investment* in future model capability. The right analogy is the neoclassical capital-accumulation model [40]. With A_t the model capability and R_t, V_t, U_t tokens spent on rollouts, verification, and updates,

$$A_{t+1} \;=\; A_t \;+\; g(R_t, V_t, U_t) \;-\; \delta\, A_t, \tag{11}$$

the optimal allocation equalizes marginal capability gain per token spent across modes: $\frac{\partial g/\partial R_t}{\kappa_R} = \frac{\partial g/\partial V_t}{\kappa_V} = \frac{\partial g/\partial U_t}{\kappa_U}$, where $\kappa_R, \kappa_V, \kappa_U$ are the per-token shadow prices introduced in Eq. 12. Equation 11 embeds a Bellman problem; with discount $\beta \in (0, 1)$ and per-token shadow prices $\kappa_{(\cdot)}$,

$$W(A_t) \;=\; \max_{R_t, V_t, U_t} \; \big\{\, \pi(A_t) - \kappa_R R_t - \kappa_V V_t - \kappa_U U_t + \beta\, \mathbb{E}\, W(A_{t+1}) \,\big\}. \tag{12}$$

SFT, DPO [36], and online RL [38, 11] are token-investment assets with different risk–return profiles: SFT is low-variance imitation, DPO is preference-bounded, online RL is high-variance exploration whose returns depend on verifier quality [10, 24]. Verifier tokens are risk capital—cutting them is identical to cutting risk capital in a financial firm: it lowers measured cost and raises tail risk [2]. The cost structure of online RL is itself shaped by the supply layer: speculative rollouts that share a tree-structured cache across trajectories [7] lower κ_R by amortizing prefix computation, which under Equation 12 should shift the optimal mix toward more rollout tokens and away from purely imitation-based investment. In agentic post-training, the situation is further complicated because the same trace produces tool calls, plans, and final answers; cleanly separating those concerns at the pipeline level [47] is what allows the planner to assign distinct shadow prices $\kappa_R, \kappa_V, \kappa_U$ to the components of an otherwise monolithic "RL token."

Portfolio frontier and closing the loop. SFT, DPO, and online RL form a portfolio of token-investment assets: SFT is low-variance, short-payback imitation; online RL is high-variance, longer-payback exploration; DPO sits between, and verifier tokens act as risk capital lowering the variance of every other asset's return. The Markowitz logic [27] predicts that the efficient frontier is a mix, not a corner—which is why "all-RL" or "all-SFT" pipelines typically underperform mixed schedules, and why aggressive verifier cuts tighten short-term budgets but blow up long-run learning curves. After this trace—now potentially training data—the same token has passed through all four layers, priced in dollars, risk, latency, and discounted future capability respectively. The four prices were never identical and were never visible to a single optimizer; they had to be reconciled by Equation 1. Agentic AI is one allocation problem, not four.

Position. Caches and RL pipelines should be reported with a depreciation rate, a hit-rate decomposition by $V(x)$, and a marginal-capability-per-investment-token estimate. Without these, "cache hit rate" and "rollout volume" are accounting metrics rather than economic ones.

4 The Cost of Local Optimization

We have followed one request through four layers and seen that each layer's mechanism is a different reading of Equation 1. We now turn from synthesis to diagnosis. The unified view sharpens what counts as a failure: a system fails not when it is slow or expensive in absolute terms, but when its allocation deviates predictably from Equation 1. The seven failure modes in Table 2 are not independent observations across heterogeneous systems; they are the corner cases of Equation 1 when one of the four prices (V, ΔC_i, λ, ρ) is held at zero or at infinity by a layer that does not see it.

Why the same failure recurs. Heterogeneous teams—router authors [8, 30], agent authors [45, 44], serving authors [20, 46], RL authors [11]—repeatedly under-price the same quantity. The structural reason is that the four prices in Equation 1 sit at different layers: V is exposed to the user, ΔC_i to the operator, λ to the SLA, and ρ to the safety team. Locally rational decisions—"my router minimizes cost," "my serving stack maximizes throughput," "my agent maximizes success rate"—compose into globally irrational allocations. The prescription is not better local optimizers but a shared accounting object [42].

Table 2: Marginal token allocation predicts a small set of recurring failure modes across heterogeneous LLM systems. Each row is a violation of Equation 1 or Equation 2.

Failure mode	Allocation violated	Where observed
Over-routing	Marginal $V\Delta Q_m < \Delta C_m$ for chosen m	Frontier-default deployments
Under-routing	$V\Delta Q_m \gg \Delta C_m$ ignored	Cost-minimizing routers
Over-delegation	$\partial R/\partial a$ exceeds $V\ \partial p/\partial a$	Auto-execute coding/email agents
Under-verification	$V\Delta Q_v - \rho\Delta R_v$ positive but $T_v = 0$	Skip-the-tests pipelines
Serving congestion	$\lambda\Delta L_i$ un-priced in ΔC_i	Flat-rate inference APIs
Stale RL rollouts	δA_t exceeds $g(\cdot)$ at the margin	Long async PPO loops
Cache misuse	Reused KV with mismatched $V(x)$	Naive prefix-cache reuse

Equilibrium across tenants. In multi-tenant deployments the failures interact. A heavy-context tenant raises λ for everyone via congestion; an aggressive autonomy tenant raises ρ via reputational risk; a high-volume RL tenant raises ΔC on inference capacity. The right object is a competitive equilibrium in which shadow prices clear across tenants [28], not a single-tenant optimization. Few production systems run such an equilibrium today; we view this as the next layer of the design problem.

Diagnosis vs. dashboard. A practical implication is that current dashboards measure the wrong things. "Tokens per dollar" is the average compute productivity; "p95 latency" is the supply-layer congestion summary; "win rate" is the demand-layer quality summary. None of them reads off Equation 1. A token-aware dashboard would instead report, per request, the realized vector $(V, \Delta C_i, \lambda\Delta L_i, \rho\Delta R_i)$ and the gap between realized and ex-post optimal allocation. This is harder to implement, but it is the only metric the framework treats as informative: every other dashboard captures a marginal slice and risks Goodhart's-law optimization at the layer that owns it.

Empirical predictions. The framework is falsifiable at the system-design level. Three predictions follow directly. First, holding V fixed, raising the agent's verifier budget T_v should monotonically reduce realized risk $R(a, t)$ until the marginal product of T_v matches its marginal cost; agents whose verifier budget is below this point should reliably under-perform on high-ρ tasks. Second, the same router that minimizes operator cost should display systematic regret on long-tail high-V requests, identifiable from logs by the gap between achieved and ex-post optimal model. Third, multi-tenant serving stacks that flat-price tokens should observe quality regressions correlated with the volume of long-context traffic, even when none of the regressing tenants used long contexts themselves—the characteristic fingerprint of an unpriced congestion externality. Each of these predictions can be checked against existing production traces.

5 Alternative Views

Token economics is a metaphor, not a theory. A reasonable critic will argue that "marginal," "screening," and "investment" are loose analogies. The analogies are formal, not rhetorical: each layer reduces to a first-order condition (Equations 1, 4, 7, 9, 11) testable on logs. The framework is falsifiable: a system that violates the relevant first-order condition should be Pareto-dominated by one that does not, and this can be checked empirically.

The right primitive is FLOPs, not tokens. Compute-optimal scaling work [15, 18] argues for FLOPs as the natural budget. We agree FLOPs are correct for pre-training. For agentic systems, however, the binding constraints are increasingly latency, action risk, and verifier quality—not raw FLOPs. A FLOP spent on prefill, on a verifier, and on a tool call is economically distinct, and tokens (not FLOPs) preserve that distinction.

Optimization, not economics, is the right frame. A well-known alternative is to treat all of this as constrained optimization or RL: write down the reward and let gradient descent allocate. We do not disagree about the implementation; we argue that economics provides the *specification*. Equilibrium concepts, screening, and externalities tell us *which* reward to optimize and *what counts*

as a market failure. Without that specification, one is free to optimize the wrong objective extremely efficiently—a recurring pattern when token cost is minimized while risk-adjusted utility falls.

Centralized planners outperform marginal rules. A trainer could in principle solve a global plan over routing, agent policy, serving, and RL training jointly. A centralized planner is a valid algorithmic target, but it must still know which prices are being minimized against which constraints. Marginal allocation supplies that language and decomposes the joint problem into auditable subproblems.

This view will be obsolete when tokens are abolished, or tokens are mere billing artifacts. Two opposing critiques converge on the same point. Some argue that latent-space agents or continuous-action policies will make "tokens" an artifact; others argue that token billing has itself produced bad incentives (e.g., chain-of-thought becoming a billing strategy). The framework absorbs both: the load-bearing concept is *marginal allocation*, not the token, and a billing artifact decoupled from ΔC_i in Equation 1 is precisely the kind of Pigouvian distortion the framework is designed to diagnose.

6 Discussion

Implications for system design. Five design and evaluation principles follow directly from Equation 1. *Token-aware evaluation* should report the four prices (V, ΔC_i, λ, ρ) and the realized allocation per request, not only aggregate accuracy and dollar cost. *Risk-adjusted routing* should publish a regret bound against Equation 4 or an incentive-compatible menu (Eq. 5), not a cost–quality scatter plot. *Autonomy pricing* should make the action class explicit and price irreversible actions higher than reversible ones, in line with Equation 7. *Congestion-priced serving* should expose shadow prices for prefill, decode, and KV resources, so that upstream allocators can read them in real time and respond to the operator's binding constraints rather than to a flat per-token list price. *RL token budgeting* should equalize marginal capability gain across rollouts, verifiers, and updates (Eq. 12) and depreciate stale rollouts at the rate δ implied by drift, not at the rate implied by an arbitrary epoch boundary. None of these principles requires new mathematics beyond Section 2; what they require is a single, instrumented price vector visible to all four layers.

Limitations. We deliberately stop at a first-order condition; we make no claim that summing Equation 1 across tasks yields a complete macroeconomic theory of AI. Three limitations deserve explicit acknowledgement. First, our prices treat compute, latency, and risk as commensurable in dollar units; this is a simplification that breaks down when physical or regulatory constraints are absolute (energy caps, data-residency rules) and require lexicographic rather than scalar treatment. Second, the framework assumes that $V(x)$ is at least partially observable; tasks whose value is realized only after long horizons (research-grade scientific reasoning, multi-month software engineering) are poorly captured by a one-step marginal rule and may require a multi-period extension. Third, our welfare-theorem argument (§2) presumes convexity of the production frontier and absence of strategic gaming on either side; LLM markets violate both at the seams, and the gap between the idealized equilibrium and the implementable mechanism remains open.

7 Conclusion

We have argued that agentic AI systems should be designed and evaluated as marginal token allocation economies. The argument is built on three load-bearing claims. First, four ostensibly separate layers—routing, agent policy, serving, and post-training—are vertical slices of a single allocation problem characterized by Equation 1, with prices that are formally Lagrange multipliers of the joint feasibility set. Second, recurring failures across the stack (over-routing, over-delegation, under-verification, congestion, stale rollouts, cache misuse) are corner cases of that equation when one of the four prices is mis-set, and they are predictable rather than incidental. Third, a Pareto-efficient allocation across the four layers requires only that the layers see a common, complete price vector—a condition that current production stacks systematically fail. The prescription is not centralization; it is shared price discovery. Returning to the developer with a failing test, the request is not a single completion but a chain of allocations: model tier, action authority, serving resources, and future training value. Today's systems price these decisions separately, producing silent downgrades, runaway autonomy, latency spikes, and noisy learning signals. The next generation of agentic AI systems will not be defined only

by cheaper tokens or larger models, but by mechanisms that allocate marginal computation closest to the risk-adjusted equilibrium.

References

[1] Amey Agrawal, Nitin Kedia, Ashish Panwar, Jayashree Mohan, Nipun Kwatra, Bhargav S. Gulavani, Alexey Tumanov, and Ramachandran Ramjee. Taming throughput-latency tradeoff in llm inference with sarathi-serve, 2024. URL https://arxiv.org/abs/2403.02310.

[2] Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, Ahmet Üstün, and Sara Hooker. Back to basics: Revisiting reinforce-style optimization for learning from human feedback in llms. *Annual Meeting of the Association for Computational Linguistics*, 2024.

[3] George A Akerlof. The market for "lemons": Quality uncertainty and the market mechanism. *Quarterly Journal of Economics*, 84(3):488–500, 1970.

[4] Armen A Alchian and Harold Demsetz. Production, information costs, and economic organization. *The American Economic Review*, 62(5):777–795, 1972.

[5] Yuntao Bai et al. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*, 2022.

[6] Tom B Brown et al. Language models are few-shot learners. *Advances in Neural Information Processing Systems*, 2020.

[7] Chi-Chih Chang, Siqi Zhu, Zhichen Zeng, Haibin Lin, Jiaxuan You, Mohamed S. Abdelfattah, Ziheng Jiang, and Xuehai Qian. Srt: Accelerating reinforcement learning via speculative rollout with tree-structured cache, 2026. URL https://arxiv.org/abs/2601.09083.

[8] Lingjiao Chen, Matei Zaharia, and James Zou. Frugalgpt: How to use large language models while reducing cost and improving performance, 2023. URL https://arxiv.org/abs/2305.05176.

[9] Ronald H Coase. The nature of the firm. *Economica*, 4(16):386–405, 1937.

[10] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

[11] DeepSeek-AI. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

[12] Avinash K Dixit and Robert S Pindyck. *Investment Under Uncertainty*. Princeton University Press, 1994.

[13] Yichao Fu, Siqi Zhu, Runlong Su, Aurick Qiao, Ion Stoica, and Hao Zhang. Efficient llm scheduling by learning to rank, 2024. URL https://arxiv.org/abs/2408.15792.

[14] Yichao Fu, Junda Chen, Siqi Zhu, Zheyu Fu, Zhongdongming Dai, Yonghao Zhuang, Yian Ma, Aurick Qiao, Tajana Rosing, Ion Stoica, and Hao Zhang. Efficiently scaling llm reasoning with certaindex, 2025. URL https://arxiv.org/abs/2412.20993.

[15] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, et al. Training compute-optimal large language models. *Advances in Neural Information Processing Systems*, 2022.

[16] Bengt Holmström. Moral hazard and observability. *The Bell Journal of Economics*, pages 74–91, 1979.

[17] Qitian Jason Hu, Jacob Bieker, Xiuyu Li, Nan Jiang, Benjamin Keigwin, Gaurav Ranganath, Kurt Keutzer, and Shriyash Kaustubh Upadhyay. Routerbench: A benchmark for multi-llm routing system. In *arXiv preprint arXiv:2403.12031*, 2024.

[18] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

[19] Frank H Knight. *Risk, Uncertainty, and Profit*. Houghton Mifflin, 1921.

[20] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM Symposium on Operating Systems Principles (SOSP)*, 2023.

[21] Jean-Jacques Laffont and David Martimort. The theory of incentives: The principal-agent model. *Princeton University Press*, 2002.

[22] Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. In *International Conference on Machine Learning*, 2023.

[23] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 2020.

[24] Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. *International Conference on Learning Representations*, 2024.

[25] Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, et al. Agentbench: Evaluating llms as agents. In *International Conference on Learning Representations*, 2024.

[26] Aman Madaan et al. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems*, 2023.

[27] Harry Markowitz. Portfolio selection. *The Journal of Finance*, 7(1):77–91, 1952.

[28] Andreu Mas-Colell, Michael D Whinston, and Jerry R Green. *Microeconomic Theory*. Oxford University Press, 1995.

[29] James A Mirrlees. The optimal structure of incentives and authority within an organization. *The Bell Journal of Economics*, pages 105–131, 1976.

[30] Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E. Gonzalez, M Waleed Kadous, and Ion Stoica. Routellm: Learning to route llms with preference data, 2024. URL `https://arxiv.org/abs/2406.18665`.

[31] OpenAI. Openai o1 system card. *arXiv preprint arXiv:2412.16720*, 2024.

[32] Long Ouyang, Jeffrey Wu, Xu Jiang, et al. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 2022.

[33] Joon Sung Park, Joseph O'Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S Bernstein. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*, 2023.

[34] Pratyush Patel, Esha Choukse, Chaojie Zhang, Aashaka Shah, Íñigo Goiri, Saeed Maleki, and Ricardo Bianchini. Splitwise: Efficient generative llm inference using phase splitting. In *ACM/IEEE 51st Annual International Symposium on Computer Architecture (ISCA)*, 2024.

[35] Arthur Cecil Pigou. *The Economics of Welfare*. Macmillan, 1920.

[36] Rafael Rafailov, Archit Sharma, Eric Mitchell, et al. Direct preference optimization: Your language model is secretly a reward model. In *Advances in Neural Information Processing Systems*, 2023.

[37] Timo Schick, Janvi Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. In *Advances in Neural Information Processing Systems*, 2023.

[38] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. In *arXiv preprint arXiv:1707.06347*, 2017.

[39] Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, 2023.

[40] Robert M Solow. A contribution to the theory of economic growth. *The Quarterly Journal of Economics*, 70(1):65–94, 1956.

[41] Michael Spence. Job market signaling. *Quarterly Journal of Economics*, 87(3):355–374, 1973.

[42] Jean Tirole. *The Theory of Industrial Organization*. MIT Press, 1988.

[43] William S Vickrey. Congestion theory and transport investment. *The American Economic Review*, 59(2):251–260, 1969.

[44] Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. In *Transactions on Machine Learning Research*, 2024.

[45] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations*, 2023.

[46] Yinmin Zhong, Shengyu Liu, Junda Chen, Jianbo Hu, Yibo Zhu, Xuanzhe Liu, Xin Jin, and Hao Zhang. Distserve: Disaggregating prefill and decoding for goodput-optimized large language model serving. In *USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2024.

[47] Siqi Zhu and Jiaxuan You. Opentinker: Separating concerns in agentic reinforcement learning, 2026. URL `https://arxiv.org/abs/2601.07376`.

A Open Problems

The framework leaves a focused set of open problems. (1) *Estimation of ΔQ_i from logs* via causal inference / off-policy evaluation [32], with calibrated variance. (2) *Risk pricing*: an empirical proxy for $\rho \Delta R_i$ that incorporates the Knightian component of Section 2. (3) *Mechanism-design routing*: do incentive-compatible menus (Eq. 5) outperform silent routing under strategic users, and how should reasoning budgets be calibrated to per-task certainty signals [14]? (4) *Internal shadow prices*: serving APIs that expose prefill, decode, and KV shadow prices upstream, building on schedulers that already learn request-level priorities [13]. (5) *RL portfolios*: when SFT, DPO, and online RL—together with architectural variants such as speculative rollouts [7] and concern-separated agentic pipelines [47]—are treated as token-investment assets, what is the efficient frontier in the (variance, capability gain) plane? (6) *Distributed equilibrium*: can the multi-tenant equilibrium of Section 2 be implemented as a clearing protocol, or must it be approximated by admission control plus priority queueing, and how should caches report depreciation δ_S in Equation 10?

B Broader impact

Treating agentic AI as a token economy makes *who pays for what* explicit, which we view as a prerequisite for accountability. A user whose request is silently downgraded does not currently see the routing decision; a tenant whose latency degrades because of an unrelated long-context workload cannot identify the externality; a workforce whose tasks are delegated to an autonomous agent has no menu of oversight intensities to choose from. Instrumented prices make these decisions auditable, which is a public good. They are not, however, a substitute for governance: a mis-set ρ on irreversible actions can still cause harm at speed, and an information-rent-extracting router can still be unfair even if it is welfare-maximizing in expectation. The framework should be read as a tool for diagnosis and design, not as a normative claim that markets settle every question. In particular, we are not arguing that decentralized token markets will spontaneously solve agentic-AI design; the history of computation markets [9, 42] shows that decentralization without instrumented prices typically produces pathological equilibria, and current LLM markets—bundled pricing, opaque routing, unpriced congestion—are precisely such an environment. The argument is that agentic systems should be designed with the prices written down, so that internal optimization, external pricing, and human oversight are aligned to the same first-order condition.