TOKENOMETRICA: TRACE-GROUNDED ACCOUNTING AND SCALING LAWS OF AGENT TOKEN DEMAND

Anonymous authors
Paper under double-blind review

ABSTRACT

LLM agents are metered and billed in tokens, yet a token total says neither where the tokens went nor how they will grow as tasks get longer. We introduce *Tokenometrica*, a framework that treats every agent run as an observable metered trace and partitions each model call, through an exact accounting identity, into new prompt, replayed context, completion, reasoning, tool-schema, tool-argument and tool-observation tokens. Token-demand functions over families of traces are stated with ordinary O/Θ in named scale variables under an explicit quality floor. We prove a *replay law*: under full-history replay a K-round loop costs $\Theta(K^2)$ tokens, with a closed-form depth beyond which replay dominates. Under random termination, expected token demand is finite if and only if $\mathbb{E}[M^2] < \infty$, and the bill inherits half the tail index of the interaction count. We also give an exact log-mean Divisia identity that attributes an empirical scaling exponent to mechanisms. We apply Tokenometrica to 26,437 public trajectories from τ-bench, τ^2-bench and SWE-agent, covering nine LLMs including a reasoning model; the reconstructed meters match provider-reported usage within 0.6% median error. Replayed context and repeatedly exposed tool schemas account for 89–96% of tokens in every system. Model-generated tokens are under 3% for non-reasoning agents, and reasoning adds only 5.3% for o4-mini because it is never replayed. On SWE-agent the token elasticity in rounds rises from 1.06 to 1.69 as runs lengthen, and replay explains 93% of it. Replay-aware forecasts cut the error of interaction-count extrapolation by about half. Live τ-bench runs confirm the architectural saving that counterfactual accounting predicts, but expose a behavioral cost: with less in context an agent takes more calls, which replay makes expensive, so masking keeps half its predicted saving and every tool-loading variant raised the bill. A 15-round circuit breaker makes each resolved SWE-agent issue 3.2× cheaper.

1 INTRODUCTION

Every model interaction of a deployed AI system is metered in tokens: prompts, completions, reasoning, and the tool descriptions, arguments and results that cross the model boundary. Agents multiply this meter. A single SWE-bench run consumes about three orders of magnitude more tokens than a code-chat exchange (Bai et al., 2026), multi-agent research systems use roughly $15\times$ the tokens of chat (Hadfield et al., 2025), and cost-aware evaluation has become a standard request for agent benchmarks (Kapoor et al., 2025; 2026; Erol et al., 2025). Yet the quantity usually reported, a token total or a dollar figure per task, answers only *how much*. It does not answer *where* the tokens were spent, *how* spending grows as tasks get longer, or *which* architectural change would lower the growth order rather than just the constant.

Practitioners have started to name token-growth regimes. The "Big-T" ladder $T(1)$, $T(\log n)$, $T(n)$, $T(nk)$, $T(nka)$, $T(\infty)$ classifies workloads by how their token use scales with usage, hidden multipliers and agent depth (Neff, 2026). By its author's own account the ladder is informal: it conflates scale variables (requests, rounds, fan-out), names no statistic of a random run, treats context replay as a constant multiplier k, and has no measurement procedure. Recent measurement studies document symptoms: input-dominated cost and high run-to-run variance (Bai et al., 2026), a "token snowball" and expensive failures (Fan et al., 2025), inflation from tool definitions (Ding et al., 2025), and stage- or role-level breakdowns (Salim et al., 2026; Wang et al., 2025b). None of them connects per-call accounting to asymptotic growth laws, or growth laws to the mechanisms that produce them.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

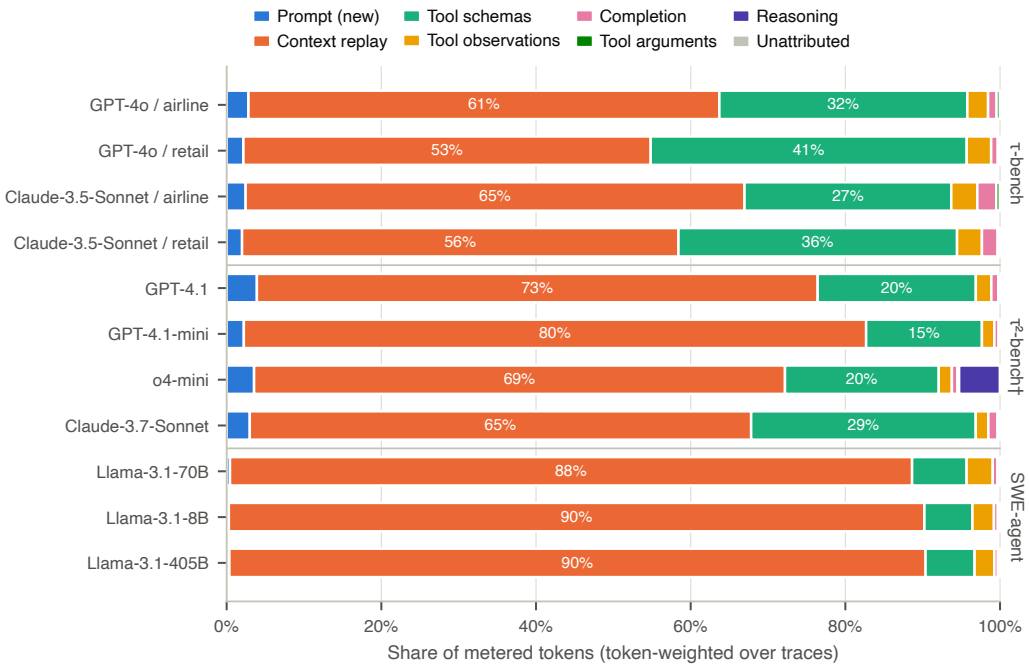

Figure 1: **Where agent tokens go.** Token-weighted composition of 26,437 traces. τ^2-bench (†) is provider-metered and attributed with a calibrated tokenizer (Appendix C). Model-generated tokens (completion and tool arguments) are under 3% for every non-reasoning agent. o4-mini's reasoning (violet) is 5.7× its visible output but only 5.3% of its bill. The rest goes to replayed context and repeatedly exposed tool schemas.

Tokenometrica. We build the missing link on one methodological principle: *a single trace measures; a family of traces scales.* For each run we reconstruct an observable metered trace and split every model interaction into disjoint token components. The split obeys an accounting identity with no double counting, and it isolates replayed context and tool-schema exposure from genuinely new information. Big-T becomes a token-demand function $T^s_\pi(\mathbf{x})$ of a named scale vector $\mathbf{x}$ under a named statistic s (mean, quantile, worst case) and quality floor q_0. Its growth is stated with ordinary O/Θ, and the trace decomposition identifies the mechanism that produces it (Section 2).

Contributions. (1) *Framework.* We give a provider-neutral accounting identity for metered traces, attribution diagnostics, token-demand functionals with an explicit quality floor, and a reporting standard (Section 2, Appendix D). (2) *Theory.* We prove a replay law with a closed-form crossover depth (Theorem 1). We also prove a stopping-time theorem tying expected, quantile and tail token demand to the moments and tail index of the number of interactions (Theorem 2); it splits Big-T's $T(\infty)$ into distinct, testable notions. Finally, we give an exact decomposition of any empirical scaling exponent into mechanism contributions (Proposition 1) and an algebra that separates order reduction from coefficient reduction (Proposition 2). (3) *Measurement.* We analyze 26,437 public trajectories (623,728 model interactions) from two tool-use benchmarks and a coding-agent corpus. The accounting pipeline is verified against brute-force prompt reconstruction and validated against provider-reported usage (Section 4). (4) *Design.* Counterfactual and budget accounting quantify whether tool loading, composition, masking, caching and circuit breakers change the order of token growth or only its coefficient, and pre-registered live runs split realized changes into that architectural saving and a behavioral cost (Section 4.5).

2 THE TOKENOMETRICA FRAMEWORK

2.1 METERED TRACES AND THE ACCOUNTING IDENTITY

An execution of workload W under architecture (or policy) π produces an observable trace $\xi = \big((I_1, \dots, I_M), E_\xi\big)$. Here I_i is the i-th model interaction, one metered call of a generative model, and E_ξ holds optional causal links (delegation, retry, merge). Each interaction is a record $I_i = (t_i, e_i, m_i, \ell_i, \mathrm{pa}_i, \mathbf{z}_i, \mu_i)$: order, event type, model, logical agent, parents, token vector and auxiliary metadata (cache flags, latency, status, termination cause, outcome). Only $\mathbf{z}_i$ is counted; everything else explains it.

Definition 1 (Accounting vector and identity)**.** The token vector of interaction i is $\mathbf{z}_i = (P_i^{\mathrm{new}}, P_i^{\mathrm{rep}}, C_i, R_i, U_i^{\mathrm{sch}}, U_i^{\mathrm{arg}}, U_i^{\mathrm{obs}})$, where P_i^{new} is context first admitted to the model at i (user turns, system prompt on first use), P_i^{rep} is previously seen context re-sent at i (earlier observations and outputs, but never tool schemas, which are charged to U_i^{sch} at every interaction), C_i is visible completion, R_i is reasoning metered separately from C_i, U_i^{sch} counts the tool-schema or command-documentation tokens exposed at i, U_i^{arg} counts model-generated tool arguments, and U_i^{obs} counts tool results returned to the model for the first time at i. The components partition the provider meter: $P_i^{\mathrm{new}} + P_i^{\mathrm{rep}} + U_i^{\mathrm{sch}} + U_i^{\mathrm{obs}} = X_i^{\mathrm{in}}$ and $C_i + R_i + U_i^{\mathrm{arg}} = X_i^{\mathrm{out}}$, where X_i^{out} is the total metered output, including any reasoning tokens the provider bills but does not return. The *token expenditure* of the trace is

$$\mathrm{Tok}(\xi) = \sum_{i=1}^{M} \tau_i, \qquad \tau_i = \mathbf{1}^\top \mathbf{z}_i. \tag{1}$$

Every token of every request is counted exactly once in that request. An observation is U^{obs} when first returned and P^{rep} whenever it is re-sent. Backend work (database scans, sandbox execution) is not a token expenditure; only the token-bearing model/tool interface is. From equation 1 we read diagnostics: component shares $s_c = T_c/\mathrm{Tok}$ with $T_c = \sum_i z_{i,c}$; the replay share $\rho_{\mathrm{replay}} = T_{P^{\mathrm{rep}}}/\mathrm{Tok}$; the schema share $\rho_{\mathrm{schema}} = T_{U^{\mathrm{sch}}}/\mathrm{Tok}$; the output share $\rho_{\mathrm{out}} = (T_C + T_R + T_{U^{\mathrm{arg}}})/\mathrm{Tok}$; retry and per-agent profiles $T_v = \sum_{i:g(i)=v} \tau_i$ for any trace label g; and *downstream amplification* $\Gamma_i = (\tau_i + \sum_{j \in \mathrm{Desc}(i)} \tau_j)/\tau_i$, which flags cheap calls (planning, delegation) that trigger expensive cascades.

Reconstruction from logs. Hosted chat APIs are stateless: the input of I_i is the system prompt, the exposed tool schemas and every preceding message, so a message log determines equation 1 up to the tokenizer and schema rendering. Provider usage fields should be used when logged; public logs rarely contain them, so we reconstruct the meter and verify it against explicit prompt reconstruction and, where available, the provider meter (Appendix B). Algorithm 1 states the procedure, and Table 6 decomposes one real trace call by call (Appendix G).

2.2 TOKEN DEMAND AND BIG-T

Definition 2 (Token-demand function)**.** Let $\{W_{\mathbf{x}}\}$ be a workload family indexed by a scale vector $\mathbf{x}$ (e.g. requests N, source size S, model-mediated rounds K, agentic expansion A), and let $\xi_{\pi,\mathbf{x}} \sim \mathbb{P}_{\pi,\mathbf{x}}$. For a statistic $s \in \{\mathbb{E}, Q_q, \mathrm{ess\,sup}\}$, the token-demand function is $T_\pi^s(\mathbf{x}) = s\big(\mathrm{Tok}(\xi_{\pi,\mathbf{x}})\big)$. A ***Big-T*** *statement* is $T_\pi^s(\mathbf{x}) = \Theta(f(\mathbf{x}))$ together with the named scale variables, the statistic, a quality floor $Q_\pi(\mathbf{x}) = \mathbb{E}[q(\xi_{\pi,\mathbf{x}})] \geq q_0$ for a task success or utility score q, and the trace mechanism that produces f.

The symbol T names the resource; O, Ω, Θ keep their standard meaning. Naming the variables matters: with k, a fixed, $\Theta(nk) = \Theta(nka) = \Theta(n)$, so the rungs are distinct only as multivariate statements. Architectures are compared only above a common floor, with benchmark $T^\star(\mathbf{x}; q_0) = \inf_{\pi: Q_\pi(\mathbf{x}) \geq q_0} T_\pi^{\mathbb{E}}(\mathbf{x})$. Growth is estimated through *token elasticities* $\alpha_j = \partial \log T / \partial \log x_j$, or between two scales $\hat{\alpha}(x_0, x_1) = \log(\hat{T}(x_1)/\hat{T}(x_0))/\log(x_1/x_0)$. A $T(nk)$ claim is falsifiable: it predicts $\alpha_N \approx \alpha_K \approx 1$.

3 THEORY: WHAT GENERATES TOKEN GROWTH

Consider a sequential model loop with M interactions. Write $\tau_i = a_i + \mathbb{I}[i \geq 2]c_0 + \sum_{j<i} h_j^{(i)}$. Here a_i is the non-replayed part of interaction i (new prompt, schemas, new observations, outputs), c_0 is a fixed prefix re-sent from the second call on, and $h_j^{(i)}$ is the portion of the history item admitted at interaction j that is re-sent at i. *Full replay* means $h_j^{(i)} = h_j$; an *observation window* w replaces items older than w by bounded stubs. We assume only bounded increments, $0 < a_{\min} \leq a_i \leq a_{\max}$ and $0 \leq h_{\min} \leq h_j \leq h_{\max}$, and require no independence or stationarity. Proofs are in Appendix A.

Theorem 1 (Replay law). *Under full replay, every trace satisfies* $a_{\min}M + c_0(M-1) + h_{\min}\binom{M}{2} \leq \mathrm{Tok}(\xi) \leq a_{\max}M + c_0(M-1) + h_{\max}\binom{M}{2}$. *Hence N tasks of K rounds cost* $T_\pi(N,K) = \Theta(NK^2)$ *when* $h_{\min} > 0$. *Under an observation window with empty stubs,* $T_\pi(N,K) = \Theta(NK)$. *With constant increments* $\bar{a}, \bar{\delta}$ *the replay share is* $\rho_{\mathrm{replay}}(M) = (M-1)/(M-1+2\bar{a}/\bar{\delta})$, *so replay dominates the bill beyond the crossover depth* $M^\times = 1 + 2\bar{a}/\bar{\delta}$.

The informal $T(nk)$ label thus hides a second factor of K whenever history is replayed, and the crossover says *when* it matters: large per-call constants $\bar{a}$ (big catalogs, long policies) delay it, verbose observations (large $\bar{\delta}$) bring it forward.

Theorem 2 (Random termination). *Let M be any (possibly trajectory-dependent) stopping time. Under full replay with* $h_{\min} > 0$*: (i)* $\mathbb{E}\,\mathrm{Tok} = \Theta(\mathbb{E}[M^2])$, *so* $\mathbb{E}\,\mathrm{Tok} < \infty$ *iff* $\mathbb{E}[M^2] < \infty$*; (ii)* $T_\pi^{(q)} = \Theta(Q_q(M)^2)$ *for every quantile level q; (iii) if* $\Pr(M > m)$ *is regularly varying with tail index* $\kappa > 0$, *then* $\log \Pr(\mathrm{Tok} > t)/\log t \to -\kappa/2$. *Under an observation window with empty stubs (or any hard cap on replayed context) the same statements hold with M^2 replaced by M and $\kappa/2$ by κ; a stub of size $s > 0$ leaves a quadratic term with coefficient at most s.*

Corollary 1 (Two kinds of $T(\infty)$). *If* $\kappa \in (1,2)$, *a replaying agent has finite expected length but infinite expected token demand. A per-task circuit breaker* $M \leq M_{\max}$ *bounds the worst case; bounding replay changes the tail class.*

Proposition 1 (Exact attribution of scaling exponents). *Let* $T = \sum_c T_c$ *over components and* $L(u,v) = (u-v)/\log(u/v)$. *For any two scales* $x_0 < x_1$, $\hat{\alpha}(x_0,x_1) = \sum_c \omega_c \hat{\alpha}_c(x_0,x_1)$ *with* $\omega_c = L(T_c(x_1),T_c(x_0))/L(T(x_1),T(x_0))$, *with no residual. In the limit* $\alpha_j = \sum_c s_c \alpha_{c,j}$, *and the total exponent converges to that of the fastest-growing mechanism.*

This is the log-mean Divisia index of index-decomposition analysis (Ang, 2005), used here to answer "which mechanism makes this agent super-linear?" with an exact identity rather than a regression.

Proposition 2 (Algebra of token-metered architectures). *(i) Sequential composition is additive, so* $T_{W_1;W_2} = \Theta(\max\{T_{W_1}, T_{W_2}\})$. *(ii) A delegation tree with branching $b \geq 2$ and depth d has* $A = (b^{d+1}-1)/(b-1) = \Theta(b^d)$ *model-bearing agents, so* $T = \Theta(NKb^d)$ *when each node's K interactions have bounded footprint (a node that replays its own history costs* $\Theta(K^2)$*); depth alone does not determine A. (iii) Pricing, prompt caching and routing multiply per-token weights within* $[p_{\min}, p_{\max}]$, $p_{\min} > 0$, *so* $\mathrm{Cost} = \Theta(\mathrm{Tok})$*: they change the coefficient, never the order (free cached input,* $p_{\min} = 0$, *is the one exception). (iv) Loading an index of size ι plus k schemas of mean size $\bar{\sigma}$, instead of all J, saves the fraction* $\rho_{\mathrm{schema}}(1 - (\iota + k\bar{\sigma})/(J\bar{\sigma}))$. *(v) Merging a run of r tool calls into one composed call removes $r-1$ interactions, each with input at least* $c_0 + J\bar{\sigma}$ *plus replayed history.*

Table 13 in Appendix N maps every rung of the Big-T ladder to formal sufficient conditions under these results.

4 MEASURING TOKEN DEMAND IN REAL AGENT TRACES

Data. We use three public corpora with success labels and full message logs. (a) *τ-bench* historical trajectories (Yao et al., 2025): 1,980 conversational tool-use runs by GPT-4o and Claude-3.5-Sonnet in the airline (J=14 tools, 1,336 metered schema tokens per call) and retail (J=16, 1,840) domains, with a simulated user. The accounting boundary is the agent model; simulator tokens are excluded. (b) *SWE-agent* trajectories (Nebius, 2024; Yang et al., 2024): 20,009 runs on 918 GitHub issues by SWE-agent driven by Llama-3.1-based 8B, 70B and 405B action models (3 of 12 released shards), with resolution labels and exit statuses. SWE-agent uses a text command interface whose documentation

Table 1: **Token accounting by system.** Means over traces; shares are token-weighted. ρ_{out}: completion + reasoning + arguments. †Provider-metered (Appendix C). Tok/succ.: total tokens divided by the number of successful runs. Task-clustered 95% CIs for the pooled shares are given in the text.

Benchmark	Agent model / domain	n	Succ.	$\bar{M}$	$\overline{\text{Tok}}$	ρ_{replay}	ρ_{schema}	ρ_{out}	Tok/succ.
τ-bench	GPT-4o / airline	200	0.42	12.3	51.1k	0.61	0.32	0.015	121.7k
τ-bench	GPT-4o / retail	460	0.60	14.3	64.3k	0.53	0.41	0.011	106.3k
τ-bench	Claude-3.5-Sonnet / airline	400	0.46	11.7	58.4k	0.65	0.27	0.029	126.9k
τ-bench	Claude-3.5-Sonnet / retail	920	0.69	13.7	70.1k	0.56	0.36	0.023	101.2k
τ^2-bench†	GPT-4.1	1,112	0.54	13.4	88.5k	0.73	0.20	0.011	162.4k
τ^2-bench†	GPT-4.1-mini	1,112	0.54	19.0	159.2k	0.80	0.15	0.007	294.1k
τ^2-bench†	o4-mini	1,112	0.57	14.4	98.2k	0.69	0.20	0.062	171.6k
τ^2-bench†	Claude-3.7-Sonnet	1,112	0.62	15.3	137.9k	0.65	0.29	0.015	224.1k
SWE-agent	Llama-3.1-70B	18,592	0.16	25.8	293.4k	0.88	0.07	0.009	1.85M
SWE-agent	Llama-3.1-8B	1,092	0.21	34.7	448.5k	0.90	0.06	0.008	2.13M
SWE-agent	Llama-3.1-405B	325	0.34	33.9	428.5k	0.90	0.06	0.007	1.25M

lives in the system prompt; we attribute it to U^{sch}. (c) τ^2-*bench* (Barres et al., 2025): 4,448 runs, namely 4 trials of 278 tasks for each of GPT-4.1, GPT-4.1-mini, o4-mini (high reasoning effort) and Claude-3.7-Sonnet, in the airline, retail and dual-control telecom domains. Its logs include provider-reported prompt and completion tokens for all 69,092 agent calls. Throughout we tokenize with `o200k_base` and charge 4 formatting tokens per message. Corpora (a) and (b) contain no reasoning models, so $R \equiv 0$ there.

Validation. (i) The vectorized accounting matches explicit prompt-by-prompt reconstruction exactly on every tested trace, under full replay and masking (Appendix B). (ii) Tool schemas are billed in the provider's own rendering: calibrating live τ-bench calls against OpenAI's meter prices the catalogs at 1,336 and 1,840 tokens per call, not the 2,436 and 2,853 of their JSON text, and we use the metered values (Appendix B). Re-tokenizing with `cl100k_base` moves the replay and schema shares by under 0.2 points. (iii) SWE-agent final-context lengths (9.2k successful, 16.0k failed) are within 10% of the dataset card's Llama-3-tokenizer figures (8.4k, 15.2k). (iv) On τ^2-bench, regressing each call's metered input on our reconstructed history with a per-domain intercept gives $R^2 \geq 0.9999$ and median error at most 0.2% for the OpenAI agents ($\hat{\beta} \approx 1.01$–1.02); Claude's tokenizer yields $\hat{\beta} = 1.203$ times as many tokens ($R^2 = 0.9969$, median error 0.6%), so we attribute the provider meter directly (Appendix C). (v) Varying each of the five accounting constants (message and call overheads, stub size, k, program overhead) keeps the replay and schema shares within one percentage point of their defaults and leaves the ranking of the counterfactuals unchanged (Appendix H). Confidence intervals come from a task-clustered bootstrap (1,000 resamples).

4.1 Where do the tokens go?

Fig. 1 and Table 1 give the decomposition. Three regularities hold across every model and domain. *First, agents mostly read.* Model-generated tokens are 2.1% of the τ-bench meter, 1.1% for non-reasoning τ^2-bench agents and 0.9% for SWE-agent. Output-length interventions (terse responses, budgeted CoT) therefore act on a tiny share of the bill. Even o4-mini, emitting 5.7× as many reasoning as visible tokens, spends 5.3% of its bill on reasoning: reasoning is metered once, history at every call. *Second, the architecture sets the balance between the two input mechanisms.* On τ-bench, repeated exposure of the full tool catalog takes 35.1% of tokens (95% CI 34.2–36.1) and replay takes 57.4% (56.3–58.4). Together they nearly exhaust the budget, and the catalog alone costs over a third of every bill, the pattern that motivated the Big-T "composability" argument. On SWE-agent, replay alone is 88.4% (88.1–88.6), and the 802-token command documentation is only 7.0%. On the provider-metered τ^2-bench, replay takes 65–80% and schemas 15–29%. The metered input grows by 188–309 tokens per call, the linear growth that makes the bill quadratic (Theorem 1). Within each model, dollar cost correlates with Tok at $r \geq 0.947$, consistent with Proposition 2(iii). *Third, failures are expensive.* Failed runs consume $1.23\times$ (τ-bench) and $2.9\times$ (SWE-agent) the tokens of successful ones, and the 99th-percentile trace costs $24.5\times$ the median on SWE-agent. The first call has median

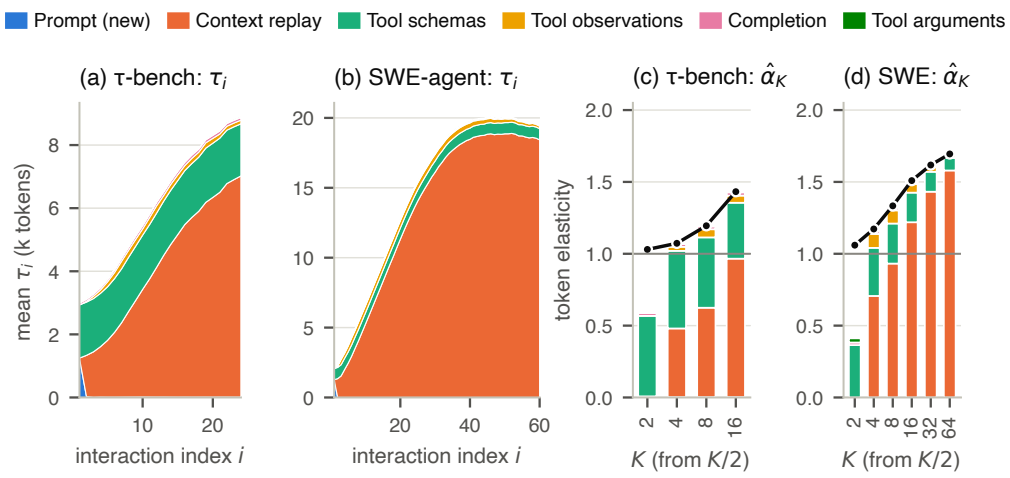

Figure 2: **Mechanism of growth.** (a,b) Mean tokens of the i-th interaction by component (runs with $M \geq i$). Schemas are a constant per call; replay grows linearly in i, which makes the prefix sum quadratic (Theorem 1). The SWE-agent plateau beyond $i \approx 40$ is the context-window cap, and partly survivorship: runs terminate on exhaustion, so only runs with small increments reach that depth. (c,d) Empirical token elasticity in rounds, $\hat{\alpha}_K$, between $K/2$ and K rounds on fixed cohorts (τ-bench: $M \geq 16$, n=561; SWE-agent: $M \geq 64$, n=1,196), decomposed exactly by Proposition 1 into component contributions (stacked) whose sum is the black line.

downstream amplification $\Gamma_1 = 19$ (τ-bench) and 49 (SWE-agent): entering the loop commits an order of magnitude more tokens than it costs.

4.2 How does token demand scale, and why?

A trace family indexed by depth is obtained exactly from the traces themselves. The prefix $T(k) = \sum_{i \leq k} \tau_i$ is the token demand of the same task truncated at k rounds, and a fixed cohort of long runs removes composition effects across k. Fig. 2(a,b) shows the mechanism directly. The per-call schema term is flat, while replay grows linearly in i with slope $\approx \bar{\delta}$. Fig. 2(c,d) shows the consequence. The elasticity $\hat{\alpha}_K$ is not a constant "k multiplier": it rises from 1.03 to 1.43 on τ-bench (95% CI of the last value 1.42–1.44) and from 1.06 to 1.69 on SWE-agent (1.68–1.71), climbing toward the asymptotic value 2 of Theorem 1, which is attained only far above the crossover and with no context cap in force. The exact decomposition attributes the growth. At the last doubling, replay contributes 0.96 of 1.43 on τ-bench (67%) and 1.58 of 1.69 on SWE-agent (93%; CI 1.57–1.59), while the schema term's contribution shrinks as its share is diluted. A single global slope would call τ-bench "$\approx$ linear" and miss its rising marginal cost per round.

The crossover explains the two regimes. We estimate Theorem 1's constants from the first 20 interactions, before any cap binds. For τ-bench this gives $\bar{a} = 2{,}070$ non-replayed tokens per call and $\bar{\delta} = 306$ history tokens per round; for SWE-agent, $\bar{a} = 1{,}423$ and $\bar{\delta} = 576$. Counting system-prompt replay, replay overtakes everything else at $M^{\times} = 11$ (τ-bench) and $M^{\times} = 6$ (SWE-agent) (Fig. 3a). The median run has $M = 13$ and $M = 17$ interactions respectively: τ-bench agents live just above their crossover, where catalog and history are of comparable size, and SWE agents live far above it, where history is almost everything. Although it is never fitted to the shares, the closed form tracks the binned replay share with mean absolute errors of 1.9 and 2.1 percentage points. The regime of an agent, and hence the right intervention, is predictable from two per-call constants measured in a handful of pilot calls. On the τ^2-bench provider meter, $\hat{\alpha}_K$ rises from 1.01 to 1.19 (replay: 79% of the last doubling) and $M^{\times} = 3$–6 against median runs of 12–15, so replay dominates that bill (Appendix I).

Table 2: **Predicting** Tok **from the number of interactions** M (5-fold, task-grouped cross-validation; relative-error least squares). R^2 on log scale.

	τ-bench			SWE-agent		
Model of Tok given M	MAPE	MdAPE	R^2	MAPE	MdAPE	R^2
Interaction count, cM ("$T(nk)$")	15.7%	13.2%	0.872	40.0%	38.7%	0.722
Free power law, $e^{\beta_0} M^{\beta_1}$	9.3%	7.3%	0.956	23.1%	18.3%	0.953
Replay law, $aM + b\binom{M}{2}$ (Thm. 1)	9.2%	7.9%	0.956	25.7%	22.8%	0.923

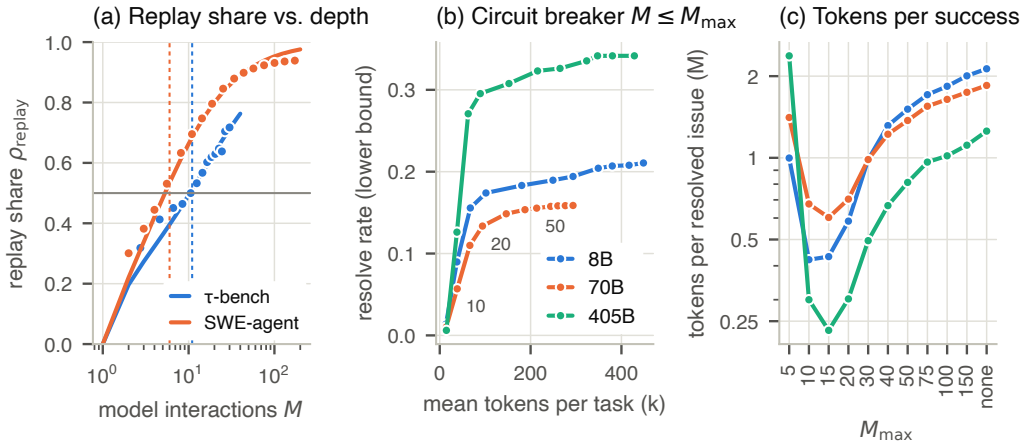

Figure 3: **Depth regimes and budgets.** (a) Replay share vs. depth: closed form of Theorem 1 (lines) against binned data (dots); dashed lines mark $M^{\times}$. (b) SWE-agent resolve rate vs. mean tokens under circuit breakers $M \le M_{\max}$ (labels on the 70B curve); runs exceeding the cap count as failures, so the resolve rate is a lower bound. (c) Tokens per resolved issue vs. $M_{\max}$.

4.3 Forecasting token bills

If the Big-T label were the whole story, cost would be proportional to the interaction count (Table 2). Super-linear laws beat the proportional model on both corpora: median APE 7.9% vs. 13.2% on τ-bench, and 22.8% vs. 38.7% on SWE-agent. A free power law fits about as well on τ-bench and slightly better on SWE-agent (18.3%). The context cap bends the longest runs toward linear growth (the windowed branch of Theorem 2), which a free exponent absorbs and the replay law trades for interpretable per-call parameters. Per-round increments vary widely between issues, so the best forecasts read them from the trace: estimating $\hat{\tau}_1$ and $\hat{\delta}$ from the *first three* interactions of a run and extrapolating with Theorem 1 gives median errors of 16.9% (τ-bench) and 31.8% (SWE-agent). Extrapolating the mean cost of those calls linearly gives 30.8% and 63.0%, and scaling the *last* observed call by M still gives 27.4% and 57.8%; both under-predict in 100% of runs, as they must whenever τ_i increases in i. Budget controllers that multiply "cost so far per step" by remaining steps are biased downward by construction. All three forecasts take the realized M as given, so they test the cost-per-step law, not stopping-time prediction. The gap persists for $k_0 \in \{2, 3, 5\}$ observed calls and on the τ^2-bench provider meter (9.3% vs. 21.3% at k_0=3; Appendix K).

4.4 Stopping, budgets and the quality floor

Because Tok grows super-linearly in M, the long runs that rarely succeed dominate the bill. On SWE-agent, 30% of runs end by exhausting the context window. They consume 74% of all tokens and resolve 2.7% of issues, against 24.0% for runs that submit normally. Circuit breakers act on exactly this mass (Fig. 3b,c). Capping at $M_{\max}$=15 removes 78% of tokens while retaining at least 70% of resolutions, which makes the cost per resolved issue 3.2× lower. A cap of 20 retains 84% at 2.7× lower cost per success. The right cap, the cheapest one whose lower-bound resolve rate stays above

Table 3: **Counterfactual architectures, decision sequence held fixed.** $\Delta\mathrm{Tok}$: change in total tokens. $\hat{\alpha}_M$: slope of $\log \mathrm{Tok}$ on $\log M$ across traces. Cost: price-weighted relative to baseline (input 1, output 4); $\mathrm{Cost}_{\mathrm{cache}}$ additionally prices the stable replayed prefix and schemas at 0.1.

Benchmark	Architecture π'	$\overline{\mathrm{Tok}}$	$\Delta\mathrm{Tok}$	$\bar{M}$	$\hat{\alpha}_M$	Cost	$\mathrm{Cost}_{\mathrm{cache}}$
τ-bench	Baseline (full replay, full tool catalog)	64.5k	–	13.3	1.39	1.00	0.22
τ-bench	Observation masking, w=1	54.5k	−15.4%	13.3	1.22	0.86	0.20
τ-bench	Observation masking, w=5	60.2k	−6.6%	13.3	1.29	0.94	0.21
τ-bench	Dynamic tool loading (oracle)	45.4k	−29.6%	13.3	1.52	0.72	0.19
τ-bench	Dynamic tool loading (index + top-3)	49.1k	−23.8%	13.3	1.47	0.78	0.19
τ-bench	Composed interface	47.9k	−25.7%	9.7	1.27	0.76	0.19
τ-bench	Composed + top-3 + mask w=5	34.9k	−45.8%	9.7	1.23	0.57	0.17
SWE-agent	Baseline (full replay, full tool catalog)	304.1k	–	26.4	1.61	1.00	0.17
SWE-agent	Observation masking, w=1	155.9k	−48.7%	26.4	1.32	0.53	0.12
SWE-agent	Observation masking, w=5	187.5k	−38.3%	26.4	1.34	0.63	0.13

q_0 (Section 2), is read off the trace family without new runs. Capping cumulative tokens instead performs almost identically (77% vs. 76% of resolutions at a quarter of the spend; Appendix J).

Theorem 2 also explains the tails. The context window, of size $c_{\max}$, acts as a hard cap on replayed history, so in the tail $\mathrm{Tok} \le (a_{\max} + c_{\max})M$ and the windowed branch applies. Consistently, the Hill estimates (Hill, 1975) for Tok (1.96) and M (2.06) coincide within noise rather than halving, as unbounded replay would give (simulated in Appendix F). Context limits are thus an implicit, and costly, circuit breaker: they stop runs only after the most expensive rounds have been paid for.

4.5 Counterfactual architecture accounting

Given a trace, we re-account the *same decision sequence* under an alternative architecture π'. *Dynamic tool loading* exposes a compact tool index plus either the schemas of the tools actually called (an oracle lower bound) or the top-3 schemas. *Composition* merges each run of consecutive tool calls into one structured program whose results return together, as in code-as-action and code-execution interfaces (Wang et al., 2024; Jones & Kelly, 2025). *Observation masking* replaces observations older than w rounds by a 12-token stub, and *caching* leaves tokens unchanged but reprices the stable prefix (Table 3). With decisions fixed, dynamic loading would remove 29.6% (oracle) or 23.8% (top-3) of τ-bench tokens, and composition 25.7% by cutting $\bar{M}$ from 13.3 to 9.7; together with masking (w=5) they would remove 45.8%. On SWE-agent, masking with w=1 or w=5 would remove 48.7% or 38.3%.

The table also separates *order* from *coefficient* (Proposition 2): prompt caching lowers price-weighted cost by 78% (τ-bench) and 83% (SWE-agent) but leaves every token and the exponent unchanged ($\hat{\alpha}_M = 1.39, 1.61$), while masking lowers $\hat{\alpha}_M$ itself (1.34 on SWE-agent at w=5); the cached prices assume a stable prefix, which per-call schema rotation would break (Appendix M). Removing the schema term *raises* the apparent exponent (1.52 for oracle loading): an observed exponent is a share-weighted mix of mechanism exponents (Proposition 1).

Live validation. Fixed decisions are an assumption, so we ran the architectures live on the 50 τ-bench airline tasks with pre-registered predictions (Appendix M), and split each arm's change exactly into a *behavior* term (its own trajectories billed under the base architecture) and an *architecture* term (the saving on those trajectories). The architecture term matches the counterfactual (masking w=5: −10.9% vs. −10.5% predicted; top-5: −14.6% vs. −15.5%), but the behavior term is always a cost: with less in context gpt-4.1-mini takes more calls, and each late call re-sends the whole history, so 0.8 extra calls per run cancel top-8 loading's schema saving (Theorem 1). Masking at w=5 saves −5.1% instead of −10.5%, and every loading variant raises the bill, from +9.2% (top-8) to +55.3% (top-3), because an agent missing a tool repeats an available call until the cap. Pass rates fell in every arm (base 0.40); at this scale only the top-3 and top-5 drops are clearly nonzero. Counterfactual accounting bounds what an architecture can save; realizing it requires an agent that does not spend the freed context on extra calls.

5 Related work

Cost-aware agent evaluation. Kapoor et al. (2025) and HAL (Kapoor et al., 2026) argue for reporting cost with accuracy, and cost-of-pass (Erol et al., 2025; Wang et al., 2025a) normalizes cost by success. Closest to us, Bai et al. (2026) characterize token totals of coding agents (input dominance, variance, unpredictability), SWE-Effi (Fan et al., 2025) names the token snowball and expensive failures, Ding et al. (2025) measure MCP-induced inflation, and Salim et al. (2026) and Wang et al. (2025b) attribute tokens to development stages and agent roles, respectively. Observability platforms such as ClawTrace (Yuan et al., 2026) record span-level token and dollar cost across tools and sub-agents. These works report *totals or partitions* (Table 11 compares coverage). Tokenometrica adds an exact mechanism-level identity, proved growth laws, exponent attribution and counterfactual accounting, and explains their observations: the snowball is Theorem 1, expensive failures Theorem 2. Neff (2026) proposed the Big-T vocabulary we formalize; Zhu (2026) motivates token-level measurement economically, and we supply the trace-level and asymptotic layer.

Inference scaling and reasoning length. Test-time scaling laws relate accuracy to inference compute (Snell et al., 2025; Brown et al., 2024; Wu et al., 2025), and work on overthinking and token budgets controls single-call reasoning length (Chen et al., 2025; Han et al., 2025; Muennighoff et al., 2025; Lee et al., 2025). We study the *metered bill* of multi-call traces, where reasoning length is a small share (Fig. 1); our fits follow the scaling-law tradition (Kaplan et al., 2020; Hoffmann et al., 2022) with its log–log caveats (Clauset et al., 2009).

Architectures that reduce mediation or context. ReWOO, LLMCompiler and CodeAct reduce model round trips (Xu et al., 2023; Kim et al., 2024; Wang et al., 2024). Tool retrieval and on-demand discovery bound schema exposure (Patil et al., 2024; Gan & Sun, 2025; Fei et al., 2025; Wu, 2025). Memory paging, compression and condensation bound replay (Packer et al., 2023; Jiang et al., 2023; Smith, 2025; Kang et al., 2026; Lindenbauer et al., 2025), and routing and cascades reprice tokens (Chen et al., 2024; Ong et al., 2025). Closest to our two live arms, AgentDiet prunes expired observations and SkillReducer compresses tool and skill descriptions (Xiao et al., 2026; Gao et al., 2026); our masking and retrieval arms are deliberately plain instances of those families, so the live savings we report are lower bounds on what tuned systems reach. Proposition 2 classifies these as order- or coefficient-changing; Section 4.5 measures them on a common basis.

6 Limitations and conclusion

Limitations. (i) For τ-bench and SWE-agent our meters are reconstructed from logs (within 0.6% median error where usage is logged), τ-bench schema costs are calibrated on OpenAI's tool rendering so Claude-3.5-Sonnet schema shares are approximate, and reasoning is identified only as metered output beyond visible output. (ii) Rounds are endogenous, so we scale depth within traces and read cross-trace exponents descriptively. (iii) Live validation covers one domain, one trial per task and one agent model; pass-rate intervals span about ± 14 points, so only the clearest quality losses are established, and the tool-search arm was designed after the pilot. (iv) Fan-out results (Proposition 2(ii)) are validated only in simulation (Appendix F); both corpora have compact observations, unlike web or GUI agents, whose larger $\bar{\delta}$ pulls $M^{\times}$ toward the first call. Budget-constrained live runs and multi-agent fan-out are the natural next tests.

Conclusion. Token totals say what happened; attribution, why; families of traces, how consumption grows; counterfactual accounting, what an architecture can save; live runs, what an agent keeps. We release the pipeline and reporting template (Appendix D) so token-demand claims can be stated, reproduced and falsified.

AI use statement

In this work, we used generative AI tools to help write and debug the analysis code, to search for and retrieve related literature, and to draft and edit the text of the paper. No experimental result, number, figure or table in this paper was produced by a generative AI tool: every quantity reported here is emitted by the released scripts from the raw public data. We have reviewed all AI-assisted work: the accounting code is covered by unit tests, including an exact brute-force check of the accounting identity against vectorized accounting on every tested trace; every citation was verified against its

primary source; and every claim and number was checked by the authors against the generated outputs. We take responsibility for the final content of this work, including all text, claims and artifacts produced with the aid of generative AI.

Ethics statement

This work analyzes publicly released agent trajectories on synthetic customer-service tasks and public open-source repositories; no human subjects or personal data are involved. Better token accounting can lower the cost and energy of deployed agents; it could also be used to optimize cost at the expense of quality, which is why every comparison is stated under an explicit quality floor.

Reproducibility statement

All data are public: the τ-bench historical trajectories, the τ^2-bench released results, and three shards of the SWE-agent trajectory release. The anonymous supplementary material contains the reconstruction and accounting library, the unit tests (including the brute-force identity check), the simulation code, and scripts that regenerate every figure, table and number in the paper from the raw data. Every number in the text is emitted by these scripts as a LaTeX macro. The live runs (Appendix M) additionally require an OpenAI API key; their logs, call-level provider usage, pre-registration and amendments are included, so all live numbers regenerate without new API calls. Proofs of all formal results are in Appendix A, and reconstruction conventions are in Appendix B.

References

B. W. Ang. The LMDI approach to decomposition analysis: a practical guide. *Energy Policy*, 33(7): 867–871, 2005. doi: 10.1016/j.enpol.2003.10.010.

Longju Bai, Zhemin Huang, Xingyao Wang, Jiao Sun, Rada Mihalcea, Erik Brynjolfsson, Alex Pentland, and Jiaxin Pei. How do AI agents spend your money? Analyzing and predicting token consumption in agentic coding tasks. *arXiv preprint arXiv:2604.22750*, 2026.

Victor Barres, Honghua Dong, Soham Ray, Xujie Si, and Karthik Narasimhan. τ^2-bench: Evaluating conversational agents in a dual-control environment. *arXiv preprint arXiv:2506.07982*, 2025.

Bradley Brown, Jordan Juravsky, Ryan Ehrlich, Ronald Clark, Quoc V. Le, Christopher Ré, and Azalia Mirhoseini. Large language monkeys: Scaling inference compute with repeated sampling. *arXiv preprint arXiv:2407.21787*, 2024.

Lingjiao Chen, Matei Zaharia, and James Zou. FrugalGPT: How to use large language models while reducing cost and improving performance. *Transactions on Machine Learning Research*, 2024.

Xingyu Chen, Jiahao Xu, Tian Liang, Zhiwei He, Jianhui Pang, Dian Yu, Linfeng Song, Qiuzhi Liu, Mengfei Zhou, Zhuosheng Zhang, Rui Wang, Zhaopeng Tu, Haitao Mi, and Dong Yu. Do NOT think that much for 2+3=? On the overthinking of long reasoning models. In *Proceedings of the 42nd International Conference on Machine Learning (ICML)*, volume 267 of *Proceedings of Machine Learning Research*, pp. 9487–9499, 2025.

Aaron Clauset, Cosma Rohilla Shalizi, and M. E. J. Newman. Power-law distributions in empirical data. *SIAM Review*, 51:661–703, 2009. doi: 10.1137/070710111.

Zihao Ding, Mufeng Zhu, and Yao Liu. Network and systems performance characterization of MCP-enabled LLM agents. *arXiv preprint arXiv:2511.07426*, 2025.

Mehmet Hamza Erol, Batu El, Mirac Suzgun, Mert Yuksekgonul, and James Zou. Cost-of-pass: An economic framework for evaluating language models. *arXiv preprint arXiv:2504.13359*, 2025.

Zhiyu Fan, Kirill Vasilevski, Dayi Lin, Boyuan Chen, Yihao Chen, Zhiqing Zhong, Jie M. Zhang, Pinjia He, and Ahmed E. Hassan. SWE-Effi: Re-evaluating software AI agent system effectiveness under resource constraints. *arXiv preprint arXiv:2509.09853*, 2025.

Xiang Fei, Xiawu Zheng, and Hao Feng. MCP-Zero: Active tool discovery for autonomous LLM agents. *arXiv preprint arXiv:2506.01056*, 2025.

Yuval Felendler, Parth A. Gandhi, Idan Habler, Yuval Elovici, and Asaf Shabtai. From tool orchestration to code execution: A study of MCP design choices. *arXiv preprint arXiv:2602.15945*, 2026.

Tiantian Gan and Qiyao Sun. RAG-MCP: Mitigating prompt bloat in LLM tool selection via retrieval-augmented generation. *arXiv preprint arXiv:2505.03275*, 2025.

Yudong Gao, Zongjie Li, Yuanyuan Yuan, Zimo Ji, Pingchuan Ma, and Shuai Wang. SkillReducer: Optimizing LLM agent skills for token efficiency. *arXiv preprint arXiv:2603.29919*, 2026.

Jeremy Hadfield, Barry Zhang, Kenneth Lien, Florian Scholz, Jeremy Fox, and Daniel Ford. How we built our multi-agent research system. Anthropic Engineering Blog, June 2025. URL `https://www.anthropic.com/engineering/multi-agent-research-system`. Published June 13, 2025. Accessed 2026-09-10.

Tingxu Han, Zhenting Wang, Chunrong Fang, Shiyu Zhao, Shiqing Ma, and Zhenyu Chen. Token-budget-aware LLM reasoning. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 24842–24855, 2025.

Bruce M. Hill. A simple general approach to inference about the tail of a distribution. *The Annals of Statistics*, 3(5):1163–1174, 1975. doi: 10.1214/aos/1176343247.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals, and Laurent Sifre. Training compute-optimal large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 35, 2022.

Huiqiang Jiang, Qianhui Wu, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. LLMLingua: Compressing prompts for accelerated inference of large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2023.

Adam Jones and Conor Kelly. Code execution with MCP: Building more efficient AI agents. Anthropic Engineering Blog, November 2025. URL `https://www.anthropic.com/engineering/code-execution-with-mcp`. Published November 4, 2025. Accessed 2026-09-10.

Minki Kang, Wei-Ning Chen, Dongge Han, Huseyin A. Inan, Lukas Wutschitz, Yanzhi Chen, Robert Sim, and Saravan Rajmohan. ACON: Optimizing context compression for long-horizon LLM agents. In *International Conference on Machine Learning (ICML)*, 2026.

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.

Sayash Kapoor, Benedikt Stroebl, Zachary S. Siegel, Nitya Nadgir, and Arvind Narayanan. AI agents that matter. *Transactions on Machine Learning Research*, 2025. arXiv:2407.01502 (2024).

Sayash Kapoor, Benedikt Stroebl, Peter Kirgis, Nitya Nadgir, Zachary S. Siegel, Boyi Wei, Tianci Xue, Ziru Chen, Felix Chen, Saiteja Utpala, Franck Ndzomga, Dheeraj Oruganty, Sophie Luskin, Kangheng Liu, Botao Yu, Amit Arora, Dongyoon Hahm, Harsh Trivedi, Huan Sun, Juyong Lee, Tengjun Jin, Yifan Mai, Yifei Zhou, Yuxuan Zhu, Rishi Bommasani, Daniel Kang, Dawn Song, Peter Henderson, Yu Su, Percy Liang, and Arvind Narayanan. Holistic agent leaderboard: The missing infrastructure for AI agent evaluation. In *International Conference on Learning Representations (ICLR)*, 2026.

Sehoon Kim, Suhong Moon, Ryan Tabrizi, Nicholas Lee, Michael W. Mahoney, Kurt Keutzer, and Amir Gholami. An LLM compiler for parallel function calling. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, 2024.

Ayeong Lee, Ethan Che, and Tianyi Peng. How well do LLMs compress their own chain-of-thought? A token complexity approach. In *ES-FoMo III: 3rd Workshop on Efficient Systems for Foundation Models (ICML 2025 Workshop)*, 2025.

Tobias Lindenbauer, Igor Slinko, Ludwig Felder, Egor Bogomolov, and Yaroslav Zharov. The complexity trap: Simple observation masking is as efficient as LLM summarization for agent context management. In *Fourth Deep Learning for Code Workshop (DL4Code) at NeurIPS*, 2025. arXiv:2508.21433.

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. s1: Simple test-time scaling. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 20275–20321, 2025.

Nebius. SWE-agent trajectories. Hugging Face dataset, `https://huggingface.co/datasets/nebius/SWE-agent-trajectories`, 2024. CC-BY-4.0. Accessed 2026-09-10.

Dan Neff. Big-T notation paper: A framework for token efficiency. Tokenomics Foundation (Linux Foundation Projects), Working Draft, June 2026. URL `https://www.tokeneconomics.com/docs/projects/big-t/big-t-notation-paper/`. CC BY 4.0. Accessed 2026-09-10.

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 from preference data. In *International Conference on Learning Representations (ICLR)*, 2025.

Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G. Patil, Ion Stoica, and Joseph E. Gonzalez. MemGPT: Towards LLMs as operating systems. *arXiv preprint arXiv:2310.08560*, 2023.

Shishir G. Patil, Tianjun Zhang, Xin Wang, and Joseph E. Gonzalez. Gorilla: Large language model connected with massive APIs. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Mohamad Salim, Jasmine Latendresse, SayedHassan Khatoonabadi, and Emad Shihab. Tokenomics: Quantifying where tokens are used in agentic software engineering. In *International Conference on Mining Software Repositories (MSR '26)*, 2026.

Calvin Smith. OpenHands context condensensation for more efficient AI agents. OpenHands (All Hands AI) Blog, April 2025. URL `https://www.openhands.dev/blog/openhands-context-condensensation-for-more-efficient-ai-agents`. Accessed 2026-09-10.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling LLM test-time compute optimally can be more effective than scaling parameters for reasoning. In *International Conference on Learning Representations (ICLR)*, 2025.

Ningning Wang, Xavier Hu, Pai Liu, He Zhu, Yue Hou, Heyuan Huang, Shengyu Zhang, Jian Yang, Jiaheng Liu, Ge Zhang, Changwang Zhang, Jun Wang, Yuchen Eleanor Jiang, and Wangchunshu Zhou. Efficient agents: Building effective agents while reducing cost. *arXiv preprint arXiv:2508.02694*, 2025a.

Qian Wang, Zhenheng Tang, Zichen Jiang, Nuo Chen, Tianyu Wang, and Bingsheng He. AgentTaxo: Dissecting and benchmarking token distribution of LLM multi-agent systems. In *ICML 2025 Workshop on Multi-Agent Systems in the Era of Foundation Models: Opportunities, Challenges and Futures*, 2025b. URL `https://openreview.net/forum?id=0iLbiYYIpC`.

Xingyao Wang, Yangyi Chen, Lifan Yuan, Yizhe Zhang, Yunzhu Li, Hao Peng, and Heng Ji. Executable code actions elicit better LLM agents. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, 2024.

Bin Wu. Introducing advanced tool use on the Claude developer platform. Anthropic Engineering Blog, November 2025. URL `https://www.anthropic.com/engineering/advanced-tool-use`. Accessed 2026-09-10.

Yangzhen Wu, Zhiqing Sun, Shanda Li, Sean Welleck, and Yiming Yang. Inference scaling laws: An empirical analysis of compute-optimal inference for LLM problem-solving. In *International Conference on Learning Representations (ICLR)*, 2025.

Yuan-An Xiao, Pengfei Gao, Chao Peng, and Yingfei Xiong. Reducing cost of LLM agents with trajectory reduction. In *Proceedings of the ACM International Conference on the Foundations of Software Engineering (FSE)*, 2026. arXiv:2509.23586.

Binfeng Xu, Zhiyuan Peng, Bowen Lei, Subhabrata Mukherjee, Yuchen Liu, and Dongkuan Xu. ReWOO: Decoupling reasoning from observations for efficient augmented language models. *arXiv preprint arXiv:2305.18323*, 2023.

John Yang, Carlos E. Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. SWE-agent: Agent-computer interfaces enable automated software engineering. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. τ-bench: A benchmark for tool-agent-user interaction in real-world domains. In *International Conference on Learning Representations (ICLR)*, 2025.

Boqin Yuan, Yue Su, Renchu Song, Sen Yang, and Jing Qin. ClawTrace: Cost-aware tracing for LLM agent skill distillation. In *Agent Skills Workshop, ACM Conference on AI and Agentic Systems (CAIS)*, 2026. arXiv:2604.23853.

Quanyan Zhu. AI tokenomics: The economics of tokens, computation, and pricing in foundation models. *arXiv preprint arXiv:2606.24616*, 2026.

A PROOFS

Throughout, an execution is a sequential model loop with a (possibly random) number M of interactions. Write the token expenditure of interaction i as

$$\tau_i \;=\; a_i \;+\; \mathbb{I}[i \geq 2]\, c_0 \;+\; \sum_{j<i} h_j^{(i)}, \tag{2}$$

where a_i collects all tokens of interaction i that are *not* replayed history (new prompt, tool schemas, new observations, completion, arguments, reasoning), $c_0 \geq 0$ is a fixed prefix (system prompt) replayed from the second interaction on, and $h_j^{(i)}$ is the number of tokens of the history item admitted at interaction j that is re-sent at interaction i. Under *full replay* $h_j^{(i)} = h_j$ for every $i > j$; under an *observation window* w, $h_j^{(i)} \leq h_{\max}$ if $i - j \leq w$ and $h_j^{(i)} \leq s$ (a stub of bounded size, possibly 0) otherwise. We assume the bounded-increment condition

$$0 < a_{\min} \leq a_i \leq a_{\max}, \qquad 0 \leq h_{\min} \leq h_j \leq h_{\max} \tag{3}$$

almost surely. This is the only assumption used; no independence or stationarity is required.

A.1 PROOF OF THEOREM 1 (REPLAY LAW)

Under full replay, summing equation 2 over $i = 1, \ldots, M$ and exchanging the order of summation,

$$\mathrm{Tok}(\xi) = \sum_{i=1}^{M} a_i + (M-1)c_0 + \sum_{j=1}^{M-1} (M-j)\, h_j.$$

Using equation 3, $\sum_{j<M}(M-j) = M(M-1)/2$ gives

$$a_{\min} M + (M-1)c_0 + h_{\min}\tbinom{M}{2} \;\leq\; \mathrm{Tok}(\xi) \;\leq\; a_{\max} M + (M-1)c_0 + h_{\max}\tbinom{M}{2}.$$

For a family of N tasks each run for K rounds, Tok is additive over tasks (Proposition 2(i)), so $T_\pi(N,K) = \Theta(NK^2)$ whenever $h_{\min} > 0$; the linear rate $\Theta(NK)$ follows from the windowed bound derived next (and trivially when $h_{\max} = 0$). Under a window w, $\sum_{j<i} h_j^{(i)} \leq w h_{\max} + (i-1)s$; with $s = 0$ this is at most $w h_{\max}$, so $\mathrm{Tok} \leq (a_{\max} + c_0 + w h_{\max})M = \Theta(M)$, and with a nonzero stub the coefficient of $\binom{M}{2}$ drops from h_j to at most s. For the crossover statement, take $a_i \equiv \bar{a}$, $h_j \equiv \bar{\delta}$ and absorb c_0 into $\bar{a}$; the replay share of the total is

$$\rho_{\text{replay}}(M) = \frac{\bar{\delta}\binom{M}{2}}{\bar{a}M + \bar{\delta}\binom{M}{2}} = \frac{M-1}{M-1+2\bar{a}/\bar{\delta}},$$

which equals $1/2$ exactly at $M^\times = 1 + 2\bar{a}/\bar{\delta}$ and increases monotonically to 1. □

A.2 PROOF OF THEOREM 2 (STOPPING TIMES, QUANTILES, TAILS)

Let $f_{\min}(m) = a_{\min} m + h_{\min}\binom{m}{2}$ and $f_{\max}(m) = (a_{\max} + c_0)m + h_{\max}\binom{m}{2}$. The proof of Theorem 1 is pathwise, so $f_{\min}(M) \leq \mathrm{Tok} \leq f_{\max}(M)$ holds almost surely for any random M, including stopping times that depend on the trajectory.

(i) Moments. Taking expectations, $a_{\min}\mathbb{E}M + \frac{h_{\min}}{2}\mathbb{E}[M(M-1)] \leq \mathbb{E}\,\mathrm{Tok} \leq (a_{\max} + c_0)\mathbb{E}M + \frac{h_{\max}}{2}\mathbb{E}[M(M-1)]$. If $h_{\min} > 0$, $\mathbb{E}\,\mathrm{Tok} < \infty$ iff $\mathbb{E}M^2 < \infty$. Under a window with $s = 0$, $a_{\min}\mathbb{E}M \leq \mathbb{E}\,\mathrm{Tok} \leq (a_{\max} + c_0 + w h_{\max})\mathbb{E}M$, so $\mathbb{E}\,\mathrm{Tok} < \infty$ iff $\mathbb{E}M < \infty$.

(ii) Quantiles. $f_{\min}$ and $f_{\max}$ are strictly increasing on $\mathbb{N}$. Because $\mathrm{Tok} \leq f_{\max}(M)$, $\Pr(\mathrm{Tok} \leq f_{\max}(m)) \geq \Pr(M \leq m)$ for all m, so $Q_q(\mathrm{Tok}) \leq f_{\max}(Q_q(M))$; symmetrically $\mathrm{Tok} \geq f_{\min}(M)$ gives $Q_q(\mathrm{Tok}) \geq f_{\min}(Q_q(M))$. Hence $T_\pi^{(q)} = \Theta\big(Q_q(M)^2\big)$ under full replay and $\Theta(Q_q(M))$ under a window.

(iii) Tails. Suppose $\Pr(M > m) = m^{-\kappa} L(m)$ with L slowly varying. For large t,

$$\Pr\big(M > f_{\max}^{-1}(t)\big) \;\geq\; \Pr(\mathrm{Tok} > t) \;\geq\; \Pr\big(M > f_{\min}^{-1}(t)\big),$$

because $\mathrm{Tok} \le f_{\max}(M)$ makes $\{\mathrm{Tok} > t\} \subseteq \{M > f_{\max}^{-1}(t)\}$, and $\mathrm{Tok} \ge f_{\min}(M)$ makes $\{M > f_{\min}^{-1}(t)\} \subseteq \{\mathrm{Tok} > t\}$; and $f_{\min}^{-1}(t) \sim \sqrt{2t/h_{\min}}$, $f_{\max}^{-1}(t) \sim \sqrt{2t/h_{\max}}$. Both bounds are $t^{-\kappa/2}$ times a slowly varying function, so $\lim_{t\to\infty} \log \Pr(\mathrm{Tok} > t)/\log t = -\kappa/2$. Under a window the inverse maps are linear in t and the limit is $-\kappa$. In particular, for $\kappa \in (1, 2)$ the loop has finite expected length but infinite expected token demand under full replay. □

A.3 Proof of Proposition 1 (exponent decomposition)

Continuous form. If $T = \sum_c T_c$ with differentiable positive components, $\partial \log T/\partial \log x_j = \sum_c (T_c/T)\, \partial \log T_c/\partial \log x_j$, i.e. $\alpha_j = \sum_c s_c \alpha_{c,j}$. *Discrete form.* For $u, v > 0$ let $L(u, v) = (u - v)/(\log u - \log v)$ ($L(u, u) = u$). Then $u - v = L(u, v) \log(u/v)$ for every pair. Applying this to each component and to the total,

$$\log \frac{T(x_1)}{T(x_0)} = \frac{T(x_1) - T(x_0)}{L\big(T(x_1), T(x_0)\big)} = \sum_c \frac{L\big(T_c(x_1), T_c(x_0)\big)}{L\big(T(x_1), T(x_0)\big)} \log \frac{T_c(x_1)}{T_c(x_0)}.$$

Dividing by $\log(x_1/x_0)$ yields $\hat{\alpha} = \sum_c \omega_c \hat{\alpha}_c$ exactly, with no residual term. As $x_1/x_0 \to 1$ the weights converge to the shares s_c. If one component grows with the strictly largest exponent, its share tends to 1, so the total exponent converges to the exponent of the dominant mechanism. □

A.4 Proof of Proposition 2 (composition algebra)

(i) Concatenating traces concatenates interaction lists; Tok is a sum over interactions, hence additive; linearity of expectation gives additivity of $T^{\mathbb{E}}$, and $f + g = \Theta(\max\{f, g\})$ for nonnegative f, g. (ii) An agent tree where every non-leaf delegates to b children to depth d has $\sum_{\ell=0}^{d} b^{\ell} = (b^{d+1}-1)/(b-1)$ nodes; with K interactions of bounded size per node, $\mathrm{Tok} = \Theta(Kb^d)$ for $b \ge 2$ and $\Theta(Kd)$ for a chain. The bounded-footprint condition is what makes the per-node cost linear in K: a node that replays its own history costs $\Theta(K^2)$ by Theorem 1, giving $\Theta(NK^2b^d)$ overall. (iii) If every price weight $p_i \in [p_{\min}, p_{\max}]$ with $p_{\min} > 0$, then $p_{\min}\mathrm{Tok} \le \mathrm{Cost} \le p_{\max}\mathrm{Tok}$, so $\mathrm{Cost} = \Theta(\mathrm{Tok})$: re-pricing, prompt caching and model routing change the constant, never the order. Masking, composition and offloading instead change M or the replay increments h_j, and therefore can change the order by Theorem 1. (iv) Exposing a compact index of size ι and k schemas of mean size $\bar{\sigma}$ instead of all J schemas changes the schema term from $J\bar{\sigma}M$ to $(\iota + k\bar{\sigma})M$, a saving of the fraction $\rho_{\text{schema}}\big(1 - (\iota + k\bar{\sigma})/(J\bar{\sigma})\big)$ of the total, holding the decision sequence fixed. (v) Merging a run of r consecutive tool-calling interactions into one composed call removes $r - 1$ interactions; each removed interaction had input at least $c_0 + J\bar{\sigma}$ plus its replayed history, which lower-bounds the saving. □

B Trace reconstruction and accounting conventions

Protocol. All three corpora were generated with stateless chat APIs. The input of model interaction I_i is the system prompt, the exposed tool schemas, and every message preceding the i-th assistant message; its output is that assistant message. Every assistant message is one interaction. Messages after the final assistant message are never sent to the model and are not counted.

Component mapping (τ-bench). The system message (the domain policy) is P^{new} at i=1 and P^{rep} afterwards. The function specifications of all J domain tools are U^{sch} at every interaction, priced at their provider-metered cost (14 airline tools, 1,336 tokens; 16 retail tools, 1,840 tokens; see *Schema calibration* below). Simulated-user turns are P^{new} when first admitted. Tool results are U^{obs} when first admitted. The visible assistant text is C. Tool calls are U^{arg}: for GPT-4o, the function name plus the argument string plus 5 wrapper tokens per call; for Claude-3.5-Sonnet, the XML `<function_calls>` block, which the log stores inside the message content and which we move from C to U^{arg} to avoid double counting. Every message receives 4 formatting tokens (role tags and separators). Everything admitted or generated at interaction j becomes P^{rep} at every later interaction.

Component mapping (SWE-agent). The system prompt is split at its `COMMANDS:` and `RESPONSE FORMAT` markers. The command documentation (802 tokens) is U^{sch} at every interaction; the

remainder is treated like the τ-bench policy. The first user message (issue statement and instructions) is P^{new}. Every later user message is an environment observation (U^{obs} when first admitted); SWE-agent returns environment feedback in the user role, so this relabelling happens in the loader, before Algorithm 1, whose `obs` tag means environment feedback rather than a literal tool-role message. The last fenced code block of each assistant message is the command (U^{arg}); the remaining discussion is C.

Verification. For each trace we also rebuild every prompt explicitly, summing the tokens of every message sent at each call. The vectorized accounting (cumulative sums over interactions) agrees exactly with this brute-force reconstruction on 120 τ-bench traces from both agent models, under full replay and under masking with $w \in \{1, 5\}$. The LMDI decomposition reproduces the total exponent to machine precision. Composition preserves the multiset of tool calls and never increases the interaction count. These checks are shipped as unit tests.

Counterfactual definitions. *Observation masking (w):* at interaction i, an observation admitted at j with $i - j > w$ is replayed as a stub costing $\min(\text{original}, 12 + 4)$ tokens. *Dynamic tool loading:* the full catalog is replaced by an index (each tool's name and the first sentence of its description) plus either the schemas of the tools actually called at that interaction (*oracle*, a lower bound) or three schemas of mean size (*top-3*). *Composition:* each maximal run of consecutive assistant messages that call tools, with no intervening user turn, is merged into one interaction. That interaction emits all the calls as a single program (20 tokens of program overhead), and the observations are returned together afterwards. Sequentially dependent calls would require generated control flow, so a flat overhead makes the composition saving an upper bound. *Cost:* input tokens weigh 1 and output tokens 4. The cached variant prices the replayed prefix and schemas at 0.1, a common prompt-caching discount. These weights are illustrative, and Proposition 2(iii) makes clear that no choice of weights changes a growth class.

Schema calibration. Providers serialize function schemas into their own prompt format, so the token count of the JSON text (2,436 airline, 2,853 retail) is not what is billed. We ran the stock tool-calling agent live on five tasks per domain (GPT-4.1-mini, 172 calls) and fitted $X_i^{\text{in}} = F_d + \beta H_i$ exactly as in Appendix C. The fit is near-exact ($R^2 > 0.9999$, median error 0.06%, $\hat{\beta} \approx 1.01$), and the intercept F_d gives the metered per-call catalog cost used everywhere in the paper. Per-tool sizes are rescaled by the same per-domain ratio; the compact tool index is plain prompt text and is not rescaled. Using the JSON counts instead would overstate the τ-bench schema share by about 11 percentage points and move the crossover from 11 to 17 calls.

Tokenizer sensitivity. Re-running the τ-bench pipeline with `cl100k_base` yields replay and schema shares of 57.4% and 35.1%, against 57.4% and 35.1% with `o200k_base`.

Replay-law constants. From interactions $i \leq 20$ (before any context cap binds): $\bar{a}$ is the mean non-replayed cost per call, $\text{mean}_i(\tau_i - P_i^{\text{rep}})$, and $(c_0, \bar{\delta})$ are the intercept and slope of an ordinary least-squares fit of the mean replayed tokens $\overline{P^{\text{rep}}}_i$ on $i - 1$ over $2 \leq i \leq 20$. Tracking c_0 separately from $\bar{a}$, the closed form of Theorem 1 becomes $\rho_{\text{replay}}(M) = R(M)/(\bar{a}M + R(M))$ with $R(M) = c_0(M-1) + \bar{\delta}M(M-1)/2$, and $M^{\times}$ is the smallest integer with $\rho_{\text{replay}}(M) \geq 1/2$; absorbing c_0 into $\bar{a}$ instead gives the simpler $1 + 2\bar{a}/\bar{\delta}$ of Theorem 1, which is larger because it charges the system prompt to the non-replayed term. The online forecast estimates $\hat{\tau}_1 = \tau_1$ and $\hat{\delta} = (\tau_{k_0} - \tau_1)/(k_0 - 1)$ from the first k_0 interactions and predicts $M\hat{\tau}_1 + \hat{\delta}\binom{M}{2}$; the linear baselines predict $M \ \text{mean}_{i \leq k_0} \tau_i$ and $M\tau_{k_0}$.

Estimators. Shares are token-weighted ratios of sums; their confidence intervals use a cluster bootstrap over tasks (τ-bench task ids, SWE-agent instance ids). Tail indices use the Hill estimator (Hill, 1975) on the top 5% of order statistics; top 2% and 10% give the same qualitative picture. Prediction models are fitted by least squares on relative error, $\sum_n ((\hat{y}_n - y_n)/y_n)^2$, with task-grouped 5-fold cross-validation. The online forecast uses the first three interactions of runs with $M \geq 6$ and the realized M.

C τ^2-BENCH: PROVIDER-METERED ATTRIBUTION

Data. We use the 12 released result files of τ^2-bench for the standard LLM agent: 4 agent models, 3 domains, 4 trials per task, with GPT-4.1 as the user simulator. Each agent message logs the provider's

`prompt_tokens` and `completion_tokens`. The opening greeting is a fixed string that is never metered; we treat it as context. In the dual-control telecom domain, the user's own tool calls and results are not visible to the agent and are excluded. The agent's system prompt is the released template (instructions plus domain policy); tool definitions are not stored in the logs.

Calibration. For each agent model m and domain d we fit $X_i^{\text{in}} = F_{m,d} + \beta_m H_i + \varepsilon_i$ by least squares. Here H_i is the reconstructed (`o200k_base`) token count of all agent-visible messages before call i. $F_{m,d}$ is the fixed per-call prefix (system prompt, tool schemas, formatting), and β_m converts our tokenizer to the provider's. The schema term is then $U^{\text{sch}} = F_{m,d} - \beta_m S_d$, where S_d is the reconstructed system prompt. The replay, new-prompt and observation terms are the β_m-scaled reconstructed components. Any remaining input goes to an explicit unattributed term, so the components sum exactly to the provider meter. On the output side, reasoning for o4-mini is $R_i = X_i^{\text{out}} - \beta_m(C_i + U_i^{\text{arg}})$; this applies an input-fitted tokenizer ratio to output text, assuming the same compression and formatting density on both sides. For the non-reasoning models, the visible components are rescaled to X_i^{out}; the ratio of metered to visible output is 0.97 for GPT-4.1 and GPT-4.1-mini and 1.31 for Claude.

Table 4: τ^2-bench validation and composition. $\hat{\beta}$, R^2 and MdAPE describe the regression of the metered input on the reconstructed history. Shares are of the provider meter.

Agent	n	Succ.	$\bar{M}$	$\overline{\text{Tok}}$	$\hat{\beta}$	R^2	MdAPE	ρ_{replay}	ρ_{schema}	ρ_R
GPT-4.1	1,112	0.54	13.4	88.5k	1.011	0.9999	0.1%	0.73	0.20	0.000
GPT-4.1-mini	1,112	0.54	19.0	159.2k	1.014	>0.9999	0.2%	0.80	0.15	0.000
o4-mini	1,112	0.57	14.4	98.2k	1.023	0.9999	0.2%	0.69	0.20	0.053
Claude-3.7-Sonnet	1,112	0.62	15.3	137.9k	1.203	0.9969	0.6%	0.65	0.29	0.000

D TOKENOMETRICA REPORT CARDS

A Tokenometrica report is the tuple $\langle \mathcal{B}, \mathcal{S}_\xi, \mathbf{x}, s, q_0, T_\pi^s(\mathbf{x}) = \Theta(f), \mathcal{M} \rangle$: the accounting boundary, the observable trace schema, the scale variables, the statistic, the quality floor, the growth law, and the mechanism. Table 5 instantiates it for the two reconstructed corpora.

Table 5: Report cards for τ-bench and SWE-agent.

Field	τ-bench (GPT-4o, Claude-3.5-Sonnet)	SWE-agent (Llama-3.1-based 8B/70B/405B)
Boundary $\mathcal{B}$	agent model input + output; user simulator excluded; R not separately metered	agent model input + output; R not separately metered
Trace schema $\mathcal{S}_\xi$	e, m, P^{new}, P^{rep}, C, U^{sch}, U^{arg}, U^{obs}, reward	e, m, P^{new}, P^{rep}, C, U^{sch}, U^{arg}, U^{obs}, exit status, resolved
Scale $\mathbf{x}$	requests N; model-mediated rounds K	requests N; rounds K
Statistic s	mean; 95th percentile in released results	mean; tail index of Tok
Quality q_0	observed pass rate (59.7%)	observed resolve rate (16.5%)
Law	$\Theta(NK^2)$ with crossover $M^\times = 11$	$\Theta(NK^2)$ up to the context cap, then $\Theta(NK)$; $M^\times = 6$
Mechanism $\mathcal{M}$	schema exposure (constant per call) below $M^\times$, replay above	replay of observations

E A STYLIZED EXAMPLE, MEASURED

Discussions of the Big-T ladder typically argue from a stylized enterprise agent rather than from logs. The canonical instance is a five-round agent with 30 exposed tools: a 2,000-token system prompt, 3,600 schema tokens exposed at every call, 600 new tokens per round and a 300-token history increment. Its bill is $5 \times 3{,}600 = 18{,}000$ schema tokens, $4 \times 2{,}000 + 300\binom{5}{2} = 11{,}000$ replayed tokens and 5,000 new tokens: 34,000 in total, of which 52.9% is schema exposure and 32.4% replay.

Every one of those constants is assumed, not measured. A real 8-call GPT-4o run (Table 6) costs 25,997 tokens, with 41.1% going to schemas and 49.6% to replay: a smaller catalog, billed in a compact rendering, already tips a short run toward replay. Pooled over τ-bench, whose catalogs are about half as large and whose runs are longer, the measured shares are 35.1% and 57.4%. The same two mechanisms dominate the bill in the stylized case and in practice alike; only their balance differs, and it is set by the crossover $M^{\times}$ of Theorem 1.

F Validation on synthetic trace families

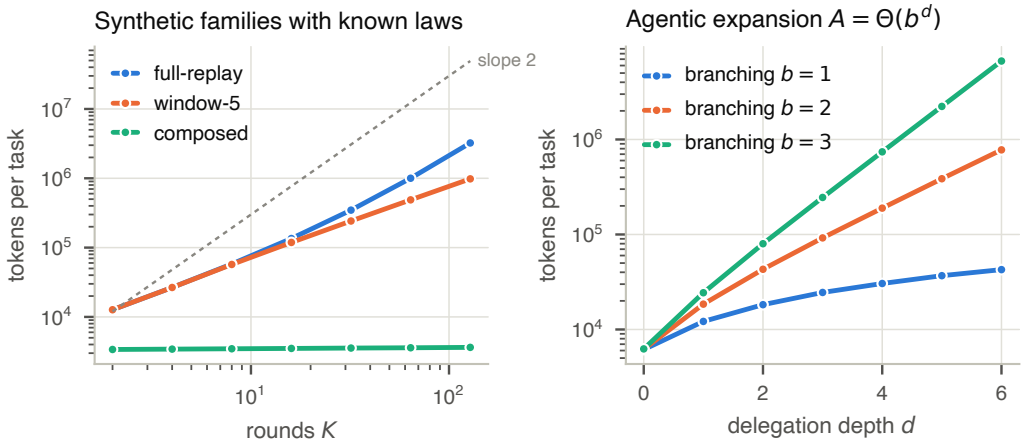

Figure 4: Synthetic trace families with known laws (mean and 5–95% band over 200 replicates). Left: full replay approaches slope 2; a window of 5 stays linear; a composed interface is flat in K. Right: agentic expansion grows geometrically in delegation depth for $b \geq 2$.

We simulate K-round loops with B=2,000 fixed prompt tokens, 3,600 schema tokens per call, V=600 new tokens per round (log-normal noise, CV 0.15), and a history increment of δ=300. These are the constants of the stylized example of Appendix E. *Exponents.* The local elasticity of full replay rises from 1.10 (K=2–8) to 1.61 (K=32–128), a window of 5 gives 1.05, and the composed interface gives 0.02. A single global slope (1.32 for full replay) would misclassify all three regimes. *Attribution.* Between K=8 and 64, Proposition 1 assigns 1.03 of the total 1.37 to replay, 0.29 to schemas and 0.05 to new tokens, which is the ground-truth mechanism. *Fan-out.* With branching $b \in \{2, 3\}$ the fitted growth rate of $\log \mathrm{Tok}$ in depth is 0.79 and 1.15, close to $\log b = 0.69$ and 1.10; the gap is the finite-depth correction in $(b^{d+1} - 1)/(b - 1)$. *Tails.* With Pareto stopping (κ=1.5, so $\mathbb{E}M < \infty$ and $\mathbb{E}M^2 = \infty$), the Hill estimate is 1.54 for M, 0.84 for Tok under full replay (theory: 0.75), and 1.54 under a window (theory: 1.5). This reproduces the halving predicted by Theorem 2. The empirical SWE-agent tail does not halve because the context window caps replay (Section 4.4).

G The accounting algorithm and a worked trace

Algorithm 1 states the accounting of Definition 1 for one trace under the stateless chat protocol. The implementation replaces the inner sum by cumulative sums over interactions ($O(M)$ per trace). It is checked against this direct form on every trace in the unit tests (Appendix B).

A worked trace. Table 6 applies the algorithm to one real run: the successful GPT-4o airline trace on τ-bench whose length is closest to 8 calls. It has $M = 8$ interactions and $\mathrm{Tok} = 25,997$. Every row sends the same 1,336-token tool catalog, and the replayed history grows by 164 tokens per call on average; individual increments vary with the size of each turn, and the P^{new} entries at $i > 1$ are simulated-user turns. Over the run, replay is 49.6% of the bill, schemas 41.1%, and model-generated output 1.7%. The first call triggers $\Gamma_1 = 9.8$ times its own cost. This is the trace-level picture behind Figs. 1 and 2.

Algorithm 1 Tokenometrica accounting of one trace (stateless chat protocol)

Require: messages $m_1, \ldots, m_L$ after the system prompt; system tokens S; schema tokens per call σ; window $w \in \mathbb{N} \cup \{\infty\}$; stub size s

$\mathcal{H} \leftarrow \emptyset$ ▷ history items (t, κ, j): tokens, kind, admission index
$i \leftarrow 0$; $\mathcal{N} \leftarrow \emptyset$ ▷ items admitted since the last call
for $\ell = 1, \ldots, L$ **do**
  if m_ℓ is an assistant message **then**
    $i \leftarrow i + 1$
    $P_i^{\text{new}} \leftarrow \sum_{(t,\text{user}) \in \mathcal{N}} t + \mathbb{I}[i = 1]\, S$; $U_i^{\text{obs}} \leftarrow \sum_{(t,\text{obs}) \in \mathcal{N}} t$; $U_i^{\text{sch}} \leftarrow \sigma$
    $P_i^{\text{rep}} \leftarrow \mathbb{I}[i > 1]\, S + \sum_{(t,\kappa,j) \in \mathcal{H}} \big(t \text{ if } \kappa \neq \text{obs or } i - j \leq w, \text{ else } \min(t, s)\big)$
    $C_i, U_i^{\text{arg}} \leftarrow$ tokens of the visible text and of the tool calls in m_ℓ
    $R_i \leftarrow X_i^{\text{out}} - C_i - U_i^{\text{arg}}$ if the provider meters output, else 0
    add every item of $\mathcal{N}$ and the output item $(C_i + U_i^{\text{arg}}, \text{out})$ to $\mathcal{H}$ with index i; $\mathcal{N} \leftarrow \emptyset$
  else
    add $(|m_\ell|,$ obs if m_ℓ is a tool result, else user$)$ to $\mathcal{N}$
  end if
end for
return $\mathbf{z}_i = (P_i^{\text{new}}, P_i^{\text{rep}}, C_i, R_i, U_i^{\text{sch}}, U_i^{\text{arg}}, U_i^{\text{obs}})$ for $i = 1, \ldots, M$

Table 6: A real τ-bench trace (GPT-4o, airline, successful), decomposed by Algorithm 1.

i	Event	P_i^{new}	P_i^{rep}	U_i^{sch}	U_i^{obs}	C_i	U_i^{arg}	τ_i
1	reply to user	1,279	0	1,336	0	36	0	2,651
2	call `get_user_details`	27	1,315	1,336	0	4	22	2,704
3	call `get_reservation_details`	0	1,368	1,336	273	4	18	2,999
4	reply to user	0	1,663	1,336	239	105	0	3,343
5	reply to user	32	2,007	1,336	0	61	0	3,436
6	reply to user	34	2,100	1,336	0	52	0	3,522
7	reply to user	32	2,186	1,336	0	48	0	3,602
8	call `transfer_to_human_agents`	43	2,266	1,336	0	4	91	3,740
Total		1,447	12,905	10,688	512	314	131	25,997

H Robustness to accounting conventions

The reconstruction uses five constants: 4 formatting tokens per message, 5 wrapper tokens per structured call, a 12-token masking stub, k=3 retrieved schemas, and 20 tokens of program overhead for composition. Table 7 varies each one while holding the others at their defaults. It uses all τ-bench runs and the first 3,000 SWE-agent runs, which is why the SWE-agent entries differ slightly from the full-corpus values of Table 3. Across all settings the replay and schema shares move by at most 0.5 percentage points. Composition removes 24.5–26.1% of τ-bench tokens. Masking with w=5 removes 6.2–6.8% on τ-bench and 32.3–38.5% on SWE-agent. Top-k loading removes 19.3–28.3% as k ranges over $\{1, 3, 5\}$. None of the qualitative conclusions of Section 4 depends on these constants.

I Provider-metered scaling on τ^2-bench

The scaling analysis of Section 4.2 used reconstructed meters. Fig. 5(a,b) repeats it on the provider meter of τ^2-bench, attributed as in Appendix C. Metered input grows linearly in the call index for all four agents. On the fixed cohort of runs with $M \geq 16$ (n=1,696), the elasticity $\hat{\alpha}_K$ rises from 1.01 to 1.19 (95% CI 1.19–1.20). At the last doubling, replay contributes 0.94 of it (79%). Table 8 gives the constants of Theorem 1 per agent. The crossover depth ranges from 3 to 6 calls, against median run lengths of 12–15. These agents therefore run well above their crossover, which is why replay takes 65–80% of their bill (Table 1). The closed-form replay share, never fitted to the shares, matches the binned data within 8.7 percentage points for every agent. The error is larger than on τ-bench because each fit pools three domains whose fixed per-call prefixes differ by a factor of about two (Appendix C).

Table 7: Sensitivity of shares and counterfactual reductions to the accounting constants. Defaults: message overhead / call overhead / stub / k / program overhead $= 4/5/12/3/20$ tokens.

	τ-bench (all 1,980 runs)					SWE-agent (3,000 runs)	
Setting	ρ_{replay}	ρ_{schema}	top-k	compose	mask w=5	ρ_{replay}	mask w=5
Default (4 / 5 / 12 / 3 / 20)	57.4	35.1	−23.8	−25.7	−6.6	88.4	−36.7
Message overhead 0	57.0	35.6	−24.1	−25.6	−6.7	88.2	−37.6
Message overhead 8	57.8	34.6	−23.5	−25.9	−6.5	88.6	−35.9
Call overhead 0	57.2	35.3	−23.9	−25.7	−6.6	88.4	−36.7
Call overhead 10	57.5	35.0	−23.7	−25.7	−6.6	88.4	−36.7
Stub 4 tokens	57.4	35.1	−23.8	−25.7	−6.8	88.4	−38.5
Stub 32 tokens	57.4	35.1	−23.8	−25.7	−6.2	88.4	−32.3
Top-k, k=1	57.4	35.1	−28.3	−25.7	−6.6	88.4	−36.7
Top-k, k=5	57.4	35.1	−19.3	−25.7	−6.6	88.4	−36.7
Program overhead 0	57.4	35.1	−23.8	−26.1	−6.6	88.4	−36.7
Program overhead 100	57.4	35.1	−23.8	−24.5	−6.6	88.4	−36.7

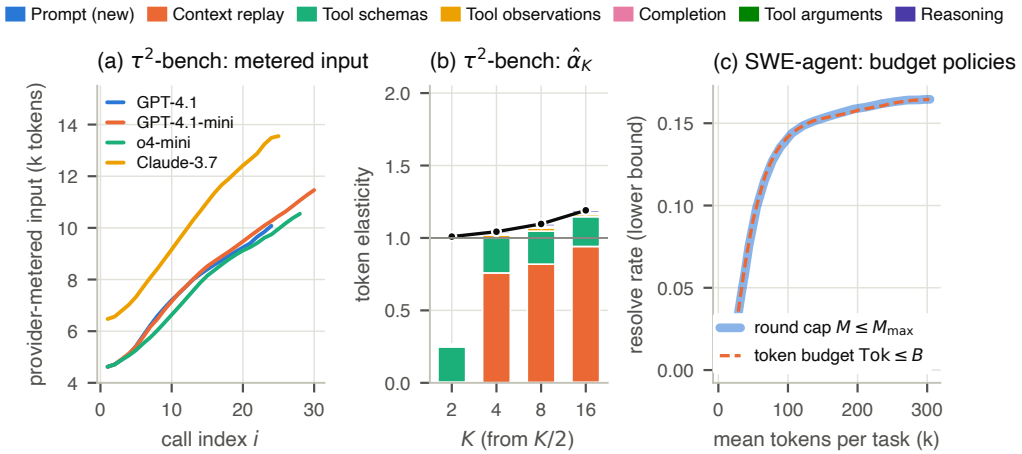

Figure 5: (a) Provider-metered input by call index on τ^2-bench (runs with $M \geq i$). (b) Exact attribution of the elasticity $\hat{\alpha}_K$ between $K/2$ and K on the provider meter. (c) SWE-agent resolve rate (lower bound) against mean tokens for a round cap and a cumulative-token budget.

J Budget policies

Section 4.4 capped the number of rounds. A deployment can instead cap cumulative tokens: the call that would push the bill past B is not issued. We re-account every SWE-agent run under both policies, again counting any run stopped before its logged end as a failure. Fig. 5(c) compares the two frontiers. At a quarter of the uncapped spend, a token budget keeps 77% of resolutions and a round cap 76%; at half the spend, 93% and 93%. The two frontiers are nearly indistinguishable, so the simpler round cap loses little. Table 9 reports the cheapest cap of each kind that keeps at least 90% of resolutions. Pooled over models, the round cap $M \leq 25$ spends 40% of the uncapped tokens, which makes each resolution 2.2× cheaper. The token budget Tok $\leq$ 229.5k spends 41%, for a 2.2× gain.

K Forecasting sensitivity

Table 10 repeats the online forecast of Section 4.3 with $k_0 \in \{2, 3, 5\}$ observed calls, and adds the provider-metered τ^2-bench. "Replay" extrapolates $\hat{\tau}_1$ and $\hat{\delta}$ with Theorem 1. "Linear" multiplies the mean cost of the first k_0 calls by M. On the provider meter with k_0=3, the replay forecast has

Table 8: Constants of the replay law on the τ^2-bench provider meter, estimated from the first 20 calls.

Agent	$\bar{a}$	c_0	$\bar{\delta}$	$M^{\times}$	median M	MAE (pp)
GPT-4.1	1,849	2,782	303	3	12	8.7
GPT-4.1-mini	1,807	2,678	314	3	14	5.8
o4-mini	2,175	2,593	288	4	14	5.6
Claude-3.7-Sonnet	3,186	3,001	380	6	15	6.2

Table 9: Cheapest caps that keep at least 90% of SWE-agent resolutions (lower bound). Tokens and resolutions are relative to the uncapped runs.

Model	Policy	Cap	Tokens kept	Resolutions kept	Tok/success gain
All	rounds	$M \leq 25$	40%	90%	2.2×
All	tokens	Tok $\leq$ 229.5k	41%	91%	2.2×
8B	rounds	$M \leq 41$	57%	90%	1.6×
8B	tokens	Tok $\leq$ 572.3k	64%	90%	1.4×
70B	rounds	$M \leq 25$	41%	91%	2.2×
70B	tokens	Tok $\leq$ 229.5k	42%	91%	2.2×
405B	rounds	$M \leq 25$	28%	90%	3.2×
405B	tokens	Tok $\leq$ 201.4k	28%	90%	3.3×

a median error of 9.3%, against 21.3% for the linear one. With k_0=5 on SWE-agent the errors are 27.0% and 63.7%. In every setting the linear forecast under-predicts the final bill in 100% of runs.

Table 10: Median absolute percentage error of online forecasts from the first k_0 calls, and the share of runs whose bill the linear forecast under-predicts.

	τ-bench			τ^2-bench (provider meter)			SWE-agent		
k_0	replay	linear	under	replay	linear	under	replay	linear	under
2	17.9%	31.7%	100%	15.9%	22.3%	100%	44.4%	64.6%	100%
3	16.9%	30.8%	100%	9.3%	21.3%	100%	31.8%	63.0%	100%
5	12.5%	28.8%	100%	5.9%	19.5%	100%	27.0%	63.7%	100%

L Coverage relative to the closest work

Partial entries reflect what each paper sets out to do. Bai et al. (2026) split totals into input and output. Salim et al. (2026) and AgentTaxo attribute tokens to development stages or agent roles, and AgentTaxo measures duplicated tokens. Ding et al. (2025) measure the inflation caused by tool definitions and history. SWE-Effi and Big-T name the growth regimes informally. Felendler et al. (2026) compare two tool-interface designs with live runs rather than by re-accounting logged traces. ClawTrace records per-span token, cache and dollar cost for observability; it measures where cost falls, while our object is how it grows and which mechanism drives it.

M Live validation of the counterfactuals

Design. We ran the stock τ-bench tool-calling agent and modified variants on the 50 airline test tasks, one trial each, with a GPT-4o user simulator. Only the architecture varies between arms: the task, the simulator, the reward function and a 30-call cap are held fixed. Two deviations from stock τ-bench apply to every arm, including the base. The user simulator runs at temperature 0, so that paired comparisons are not dominated by simulator sampling; stock τ-bench passes no temperature. Transient API errors (rate limits, timeouts, 5xx) are retried with backoff and never scored; any other

Table 11: What each line of work provides, based on our reading of the papers. ✓: provided; ○: partial or qualitative; –: not addressed.

Work	Per-call attribution	Replay vs. schema	Growth laws with proofs	Exponent attribution	Counterfactual re-accounting
Kapoor et al. (2025); HAL (Kapoor et al., 2026)	–	–	–	–	–
Bai et al. (2026)	○	–	–	–	–
SWE-Effi (Fan et al., 2025)	–	–	○	–	–
Ding et al. (2025)	○	○	–	–	–
Salim et al. (2026)	○	–	–	–	–
ClawTrace (Yuan et al., 2026)	✓	○	–	–	–
AgentTaxo (Wang et al., 2025b)	○	○	–	–	–
Felendler et al. (2026)	–	–	–	–	○
Big-T (Neff, 2026)	–	–	○	–	–
Tokenometrica (ours)	✓	✓	✓	✓	✓

exception counts as a failure. Schema costs use the provider-metered calibration of Appendix B, and every call's provider usage is logged, so each arm's token change is read from the meter.

Arms. *Masking* ($w \in \{1, 5\}$) sends tool results older than w calls as a 12-token stub while keeping the true history. *Retrieval loading* (top-k, $k \in \{3, 5, 8\}$) appends the compact tool index to the system prompt and exposes the k schemas whose descriptions are most similar (text-embedding-3-small) to the latest user turn and observation. *Tool search* appends the index and exposes a single `load_tools(names)` tool; requested schemas stay exposed for the episode, and each lookup is an ordinary metered call that never reaches the environment. All transformations were unit-tested offline against the accounting of Appendix G (3,180 call-level comparisons, exact) before any live call.

Pre-registration. Hypotheses, arms, margins and token predictions were written to a timestamped file before the first live call, and every later change was recorded as a dated amendment before the runs it affects. (1) The schema calibration replaced JSON token counts and re-derived every prediction. (2) A first pilot attempt without retries was invalidated by rate limiting, and a second by a retry wrapper that was not applied; crashed runs were re-run and never scored, and pass rates printed during those attempts were not used. (3) After the pilot falsified the headline predictions, the pre-registered retrieval sweep ($k = 5, 8$), masking at w=5 and the tool-search arm were added with fixed predictions; tool search is exploratory because it was designed after seeing the pilot. (4) The gpt-4.1 arms and the hypothesis that their behavior cost is smaller were registered before those runs.

Decomposition. For arm π' on the shared tasks, let $T_{\pi'}$ be its tokens as run, T_{base} the base tokens, and $\tilde{T}_{\pi'}$ the arm's own trajectories re-accounted under the base architecture (full catalog, full replay). Then

$$T_{\pi'} - T_{\text{base}} = \underbrace{(\tilde{T}_{\pi'} - T_{\text{base}})}_{\text{behavior}} + \underbrace{(T_{\pi'} - \tilde{T}_{\pi'})}_{\text{architecture}}$$

exactly. The fixed-decision counterfactual of Section 4.5 estimates only the architecture term, computed on the base trajectories. Holding decisions fixed, top-k loading saves $s = U^{\text{sch}} - (\iota + k\bar{\sigma})$ tokens on each of M calls, while an extra call at depth M costs about $\bar{a} + c_0 + \bar{\delta}M$ (Theorem 1); the saving is cancelled by $\Delta^* = sM/(\bar{a} + c_0 + \bar{\delta}M)$ extra calls, 0.8 for top-8 at the τ-bench constants.

Failure mode. When a needed tool is not exposed, gpt-4.1-mini repeats an exposed call, often with identical arguments, until the cap. In 12 of the 17 capped tool-search runs the repeated call was the same user lookup, and the tools the task required were never loaded. Retrieval shows the same pattern: on the base agent's own decisions, the retriever ranks the tool the agent used in the top 3 only 47% of the time (67% top-5, 91% top-8). The stock agent executes only the first tool call of each response; in a re-run of four looping tasks with emitted calls logged, two had dropped parallel calls and two did not, so truncation aggravates but does not explain the loops. It applies to every arm, including the base.

Scope. One domain, one trial per task, one agent model (gpt-4.1-mini), and a single retriever and prompt for tool search; pass-rate intervals span about ±14 points. Whether the behavior cost shrinks with a stronger agent is open; a pre-registered gpt-4.1 replication of the base, masking w=5, top-8 and tool-search arms is specified in the supplementary amendments but not yet run. For these agents, fixed-decision savings overstate the realized saving and can have the wrong sign; the results do not

Table 12: Live arms on τ-bench airline (50 tasks, one trial). Pass: task success rate; Δpass in percentage points vs. the same model's base. ΔTok metered: provider-reported tokens vs. base. Predicted: fixed-decision counterfactual from the base trajectories. Behavior and architecture sum to the realized accounted change (within 2.5 points of the metered change). Capped: runs reaching 30 calls. *Exploratory.

Architecture	Pass	Δpass (pp)	ΔTok metered	Predicted	Behavior	Architecture	Capped
gpt-4.1-mini: base pass rate 0.40, 61.4k metered tokens and 14.0 calls per run							
Masking, w=5	0.34	−6	−5.1	−10.5	+5.8	−10.9	2
Masking, w=1	0.26	−14	−11.3	−17.9	+9.3	−20.8	6
Retrieval, top-8	0.30	−10	+9.2	−8.9	+13.1	−4.1	5
Retrieval, top-5	0.22	−18	+17.7	−15.5	+31.7	−14.6	11
Retrieval, top-3	0.16	−24	+55.3	−19.8	+82.4	−29.5	19
Tool search*	0.28	−12	+45.3	−24.2	+76.4	−32.5	17

show that the architectures cannot pay with stronger agents, better retrieval or harnesses that execute parallel calls.

N THE BIG-T LADDER, FORMALIZED

Table 13 states each rung of the informal ladder as a claim about a named scale variable, with the result that governs it.

Table 13: **The Big-T ladder (Neff, 2026) made formal.** Each informal label becomes a statement about a named scale variable, with the result that governs it.

Label	Scale variable	Sufficient condition	Formal statement
$T(1)$	N or S	scale creates no new model-token path (finitely many cache misses; one-off code generation)	$T_\pi(x) = \Theta(1)$ in the named variable
$T(\log n)$	source size S	admitted context $g(S) = \Theta(\log S)$	$T_\pi(S) = \Theta(g(S))$; fixed-k retrieval gives $\Theta(1)$
$T(n)$	requests N	bounded footprint per request	$T_\pi(N) = \Theta(N)$
$T(nk)$	N, K	K rounds of bounded payload	$\Theta(NK)$; under full replay $\Theta(NK^2)$ (Theorem 1)
$T(nka)$	N, K, A	A model-bearing branches of K rounds, bounded footprint per branch	$\Theta(NKA)$; a delegation tree has $A = \Theta(b^d)$ (Proposition 2)
$T(\infty)$	stopping time M	no finite bound, or infinite mean	$T^{\mathrm{wc}}{=}\infty$ vs. $T^{\mathbb{E}}{=}\infty$; under replay $T^{\mathbb{E}}{<}\infty$ iff $\mathbb{E}M^2{<}\infty$ (Theorem 2)