DOES MULTI-AGENT COLLABORATION PAY WHEN AGENTS SERVE USERS? TESTING AGENT SCALING IN DUAL-CONTROL SETTINGS

Anonymous authors
Paper under double-blind review

ABSTRACT

Multi-agent systems (MAS) built from language models are increasingly deployed on the premise that collaboration among agents outperforms a single-agent system (SAS), yet when collaboration pays remains poorly understood. The most comprehensive study to date answers with a capability-saturation principle: coordination helps while the underlying SAS is weak and hurts once it is strong, with the crossing near 45% single-agent accuracy. That principle was derived on autonomous benchmarks, where no user takes part. We test it where agents increasingly operate, in dual-control settings, where the agent serves a user under a written policy and both act on a shared environment. On the airline, retail and telecom domains of τ^2-bench we compare a SAS with a centralized MAS built from the same model (an orchestrator, a database agent and a policy advisor) for six models from three families (four, from two families, in the primary analysis), over about 7,000 simulated conversations, with each system held to one completion-token ledger per episode. Under a fixed simulated user no cell shows a MAS benefit: the MAS is worse in 12 of 13 valid cells and tied in the thirteenth, a cell near 45% where the benchmark's default simulator yields a 10-point MAS gain. Where the principle predicts a benefit, the MAS loses by 13 and 22 points, and removing the budget ceiling brings it at best level with the SAS at 5.5 times its median token spend. A capability index measured on held-out domains shows no detectable dependence of the gain on capability (slope -0.41, 95% CI $[-1.51, +0.68]$), although this interval is too wide to exclude a steep decline; the fitted gain is negative across the observed range, with an interval that excludes zero only from an index of 60% upward. The mechanism that would justify a MAS here is not detectable: the MAS shows no measurable reduction in wrongful database writes and converts other failures into budget exhaustion and inaction. For agents that serve people, these results make the SAS the default and place the burden of proof on multi-agent designs. We release all trajectories, budget ledgers, prompts and code.

1 INTRODUCTION

Language-model agents are moving from research prototypes into deployed products, and one design question now precedes almost every deployment: should a task be handled by a single-agent system (SAS) or by a multi-agent system (MAS) whose agents plan, check and specialize? Multi-agent frameworks have multiplied, and industrial reports describe large gains from orchestrating several agents, at many times the tokens of a single one (Anthropic, 2025). Controlled studies that hold compute fixed often find the SAS as strong or stronger (Jwalapuram et al., 2026; Tran & Kiela, 2026). The choice is consequential, since a MAS multiplies model calls, latency and points of failure.

Kim et al. (2026) gave the most systematic answer to date, and with it the beginnings of a science of agent scaling. Across 260 configurations spanning six benchmarks, five architectures and nine models, the capability of the underlying SAS was the strongest predictor of what coordination adds: a MAS pays while the SAS is weak and costs once it is strong, with the fitted crossing near 45% single-agent accuracy. In a validation on 16 model-by-benchmark cells the principle predicted the direction of the MAS effect in 15, and it offers practitioners a simple rule for choosing an architecture.

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.

Every benchmark behind this principle is autonomous: the agent receives a task and acts alone. Many agents now reaching users operate in a different regime, which Barres et al. (2025) call dual control. The agent converses with a person who supplies information late, changes goals and asks for what the policy forbids, and both act on a shared environment. This regime bears on a MAS in three ways. The conversation has one user-facing locus, so most generic MAS architectures have no deployable analogue. Input tokens dominate the cost, so what is held constant when comparing systems decides the outcome. And the user is simulated by a model whose choice alone can move scores as much as the per-cell effects reported for the principle (Seshadri et al., 2026; Zhou et al., 2026). Whether principles derived without a user hold once a user takes part is of both scientific and practical weight.

We present such a test on τ^2-bench (Barres et al., 2025), the standard dual-control benchmark, where the reward is the state of a database or device when the conversation ends. We compare the benchmark's stock SAS with a centralized MAS built from the same model: an orchestrator that plans and talks to the customer, a database agent that executes its reads and writes, and a policy advisor (Figure 1). Six models from three families on three domains give 18 cells, of which 16 were run, for about 7,000 simulated conversations in all. Within a cell both systems run the same 50 tasks with three seeds, the MAS under a per-episode completion-token ceiling calibrated from the SAS's own spend. We read the result in one of three ways: the principle *transfers* if the gain declines with capability and crosses zero near 45%, *shifts* if it crosses elsewhere, and *disappears* if no dependence is detected. Mechanical validity checks separate a MAS that does not help from one that is broken, and the primary test uses a capability index measured on held-out domains, which removes the negative slope that a gain regressed on its own baseline shows by construction.

The principle does not survive the move. Under a fixed simulated user no cell shows a MAS benefit: the MAS is worse than the SAS in 12 of 13 valid cells and tied in the thirteenth, the cell nearest 45%, where only a change of simulator produces a 10-point gain (§4.6). On the two cells where the principle predicts a benefit, the MAS loses by 13 and 22 points, and with no budget ceiling at all it at best draws level at 5.5 times the SAS's median token spend (§4.3). The slope test detects no dependence of the gain on capability, although its interval cannot exclude a steep decline, and the fitted gain is negative across the observed range, significantly so from an index of 60% upward (§4.4). The mechanism that motivates a MAS in this setting, intercepting wrongful database writes, is not detectable: the MAS does not measurably reduce them and converts other failures into budget exhaustion and inaction (§4.5).

Contributions. (1) The first test of the capability-saturation principle in the dual-control regime, where agents serve users. (2) A compute-matching protocol for multi-turn agent systems: one per-episode completion-token ledger for all agents of a system, calibrated per cell. (3) An analysis, supported by a lemma, that separates capability dependence from the coupling between a gain and its own baseline. (4) Evidence on why the MAS fails: a taxonomy separating write-type from word-type violations, per-cell validity checks, and a no-ceiling arm that rules out the budget as the sole cause. (5) Release of all trajectories, ledgers, prompts and code.

2 Background and related work

The capability-saturation principle. Kim et al. (2026) compared a SAS with four generic MAS architectures (independent, centralized, decentralized, hybrid) across six autonomous benchmarks and three model families at three tiers, holding prompts, tools and a per-system reasoning-token budget constant. In their linear model the SAS baseline is the strongest predictor of the MAS gain (coefficient 0.250); its interaction with the number of agents (-0.236) does not survive cluster-robust correction, and the ratio of the two places the crossing near 45%. Its sign validation (15 of 16 cells on SWE-bench Verified and Terminal-Bench) used 20 tasks per cell, with per-cell intervals of about 20 points. The threshold is global, with no benchmark-specific parameter, so a new setting needs only its own SAS baselines, and the τ-bench family is named in that paper as untested.

Compute-matched comparisons. Most positive results for MAS are not compute-matched (Kapoor et al., 2025; 2024). Under a shared token budget self-consistency often beats debate and reflection (Wang et al., 2024); under a fixed thinking budget a SAS is at least as information-efficient as any multi-agent decomposition and was best or tied at every budget but the smallest (Tran & Kiela, 2026); automatically generated MAS underperform self-consistency at up to ten times the cost (Jwalapuram

et al., 2026). Several results agree with the principle, with the benefit of a MAS shrinking as models improve (Gao et al., 2025; Li et al., 2026; Yang et al., 2025; Yu et al., 2026). These protocols hold different quantities constant, and none was built for multi-turn episodes in which a simulated user adds tokens the agent does not control and input dominates the total.

Conversational evaluation and the simulated user. τ-bench (Yao et al., 2024) scores an agent that converses with a simulated user under a domain policy by the final database state; τ^2-bench (Barres et al., 2025) adds a telecom domain in which both parties hold tools, and the user alone costs GPT-4.1 18 points there. Human participants succeeded on 45.2% of τ-bench retail tasks against 67.8% with a simulated user, and swapping the simulator model alone moved agent success from 67.0% to 75.9% (Seshadri et al., 2026); simulators are over-cooperative and inflationary (Zhou et al., 2026). A per-cell MAS effect of the size Kim et al. (2026) report is within what simulator choice alone can produce, so we fix one simulator and test a second (R2).

Verification of state-changing actions. On τ-bench traces each deviation in a state-changing action lowers the odds of success by up to 92% on airline and 96% on retail, while errors in read-only actions matter little (Cuadron et al., 2025), and dialogue-grounded verifiers add 6 to 12 points of pass^4 on τ^2 airline (Kang et al., 2026). These domain-specific designs act on the wrongful-write class that our taxonomy separates from verbal violations. Comparisons of generic architectures on τ-bench-style tasks are scarce and neither compute-matched nor varied over capability (LangChain, 2025); Table 13 (Appendix O) positions the closest work.

3 Experimental design

3.1 Setting, systems and unit

Tasks. We use τ^2-bench v1.0.1 at a pinned commit with the evaluator and domain data unmodified (Appendix B). The reward checks the final database state and required communication on airline, the database state and natural-language assertions graded by a pinned judge on retail, and the state of the customer's device on telecom, where the customer runs tools too. Airline uses its 50 base tasks; retail and telecom use fixed 50-task subsets drawn by one seeded shuffle. Every cell runs every task with three seeds (150 episodes per condition). One simulator, `gpt-5.2` at low reasoning effort, plays the customer everywhere; the benchmark's default `gpt-4.1` is used only in R2.

Notation. A cell $i = (m, d)$ pairs model m with domain d and its task set $\mathcal{T}_d$ ($|\mathcal{T}_d| = 50$). For system $A \in \{\text{SAS}, \text{MAS}\}$, task t and seed $s \in \{1, \dots, S\}$ ($S = 3$), let $R^A_{t,s} \in \{0, 1\}$ be the episode reward. Then

$$P_A(i) = \frac{1}{|\mathcal{T}_d|} \sum_{t \in \mathcal{T}_d} \bar{R}^A_t, \qquad \bar{R}^A_t = \frac{1}{S} \sum_{s=1}^{S} R^A_{t,s}, \qquad \Delta_i = P_{\text{MAS}}(i) - P_{\text{SAS}}(i). \tag{1}$$

The unit is the per-task mean $\bar{R}^A_t$, and $P_A(i)$ is pass^1, the axis on which the principle is stated: it predicts $\operatorname{sign} \Delta_i = \operatorname{sign}\big(x^* - P_{\text{SAS}}(i)\big)$ with $x^* \approx 0.45$. Intervals are task-cluster bootstraps (tasks resampled with their episodes; seed 300), and Δ_i is reported in points.

SAS. The benchmark's stock agent (S1 in the released code): its default system prompt with the domain policy inserted verbatim, native function calling and the highest reasoning effort. This is the SAS over which the principle is stated and the agent behind public leaderboard scores (Appendix B).

MAS. The centralized architecture adapted to one conversational thread (M1 in the released code; Figure 1b), with three agents built from the same model. The *orchestrator* alone talks to the customer and keeps a task ledger and a progress ledger after Magentic-One (Fourney et al., 2024); the *database agent* alone calls tools, with up to $r = 4$ consecutive calls per read assignment and one call per write, executed as ordered without veto; the *policy advisor* reasons with a fresh context and returns a finding with the quoted policy lines. The benchmark is half-duplex, so each orchestrator turn is a *plan* call without tools and, if the advisor was asked, an *act* call; a turn with a tool call thus costs at least two calls against one in the SAS, the coordination cost the budget charges. Role blocks describe workflow

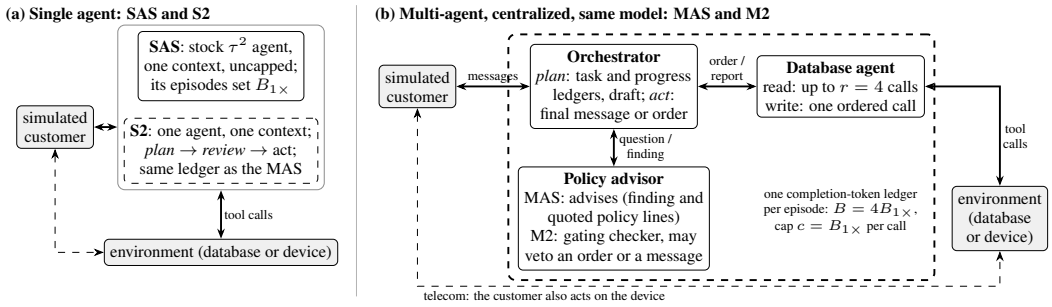

Figure 1: The systems. (a) Single-agent systems: the stock SAS, whose uncapped episodes calibrate the budget, and S2, one agent that runs the MAS's plan-and-review cycle in one context under the same ledger. (b) The centralized MAS built from the same model: only the orchestrator talks to the customer, only the database agent calls tools, and all calls draw on one completion-token ledger (Eq. 2); in M2 the policy advisor becomes a checker that can veto an order or a message (outlined, not evaluated here). On telecom the customer also acts on the device (dashed).

only and follow the unchanged base prompt; every prompt is hashed (Appendix B). The other generic architectures would need an adaptation that could itself explain the result (Appendix C).

3.2 Budget protocol

In the pilot the SAS read 57 input tokens for every token it wrote, so matching total tokens would mostly charge a MAS for duplicated prompts. The comparison is therefore matched on *system-wide completion tokens*: κ_j counts the completion tokens of call j by any agent of the system, including hidden reasoning, tool-call arguments and inter-agent messages. Other tokens are reported as deployment accounts, and the meter matches provider billing within 1% (Appendix E).

Budgets are calibrated per cell. The SAS first runs the cell uncapped; these episodes, $\mathcal{E}_{\text{SAS}}(i)$, are the SAS arm and the calibration. For an episode e of the MAS,

$$C_e = \sum_{j \in e} \kappa_j, \qquad B_{1\times}(i) = Q_{0.75}\{C_{e'} : e' \in \mathcal{E}_{\text{SAS}}(i)\}, \qquad B = kB_{1\times}, \tag{2}$$

and call j may write at most $m_j = \min(B - C_{<j}, c)$ tokens, where $C_{<j}$ is the spend before it, so all agents draw on one ledger (Appendix C). $k = 4$ is the smallest multiple that left every SAS episode unconstrained in the calibration runs (largest $3.85B_{1\times}$). The per-call cap $c = B_{1\times}$ touches 0 to 1% of SAS episodes. Once $B - C_{<j} < w = \max(0.15B_{1\times}, 2\,\text{median}_{\text{SAS}})$, a fixed notice tells the system to conclude with the customer; once $B - C_{<j} < a_{\min}$ (the median SAS call), the episode ends with the label *budget exhausted*. Exhausted episodes are scored as they stand and kept apart from inaction; a truncated tool call is charged and never executed. The ledger held in every MAS episode except three on retail × `gemini-3.1-flash-lite`, where the provider returned up to 1.8% more tokens than the limit sent.

Two properties of the protocol matter for reading the results. First, the SAS runs uncapped while the MAS runs under the ceiling, so the asymmetry lies mainly in the notice, and R1 re-runs the SAS capped to size it. Second, the original limit (about 100 model calls per system, no token cap, against a realised 7.2 calls for the SAS and 27.7 for the centralized MAS) is looser than ours, so the grid at $4B_{1\times}$ is a *cost-bounded arm*, and a MAS with no ceiling forms a secondary *uncapped arm* on the three cells at or below 45% (Table 6).

3.3 Outcome measures

We use three measures, each blind to something the others catch. The *primary* measure is the official reward, which on airline and telecom misses violations that never reach a write, such as promising a forbidden action. The *secondary* measure is episode-level compliance with every natural-language assertion under the official judge. The *tertiary* measure is the pass rate on tasks whose gold trajectory contains a write, because the database comparison rewards passivity on refusal tasks. Failures are

classed as wrongful write (WW, visible to the primary measure), inaction (IN), budget exhausted (BX), judge-flagged pass (FP, invisible to it; airline only, since telecom tasks carry no natural-language assertions) and, for the MAS, a write executed differently from its order (Appendix H).

3.4 Hypotheses, validity checks and power

Two hypotheses are primary; everything else is secondary. Table 5 (Appendix A) gives each test, its reading rule and the result in one row.

H1. The primary model regresses the gain on a capability index,

$$\Delta_i = \alpha + \beta x_i + \gamma_{f(i)} + \delta_{d(i)} + \varepsilon_i, \qquad x_i = \frac{1}{2}\sum_{d' \neq d} P_{\text{SAS}}(m, d'), \tag{3}$$

with family and domain fixed effects $\gamma_{f(i)}$ and $\delta_{d(i)}$, fitted by weighted least squares with weights $\omega_i = \widehat{\text{Var}}(\Delta_i)^{-1}$ from the task-cluster bootstrap. The index x_i, the model's SAS pass[1] on its other two domains, shares no episode with Δ_i; Lemma 1 shows why this matters. H1 holds if $\hat{\beta} < 0$ with a 95% CI excluding zero, with the p-value from 10,000 permutations of Δ_i within family. The crossing at the mean fixed effect, $x_0 = -(\alpha + \bar{\gamma} + \bar{\delta})/\beta$, with a task-cluster bootstrap CI, gives the reading: the principle *transfers* if H1 holds and the CI of x_0 covers 0.45, *shifts* if H1 holds and it does not, and *disappears* if H1 fails. Split-seed, logit and relative-gain forms are reported beside it, and the naive form $x_i = P_{\text{SAS}}(i)$ descriptively.

Lemma 1 (Coupling). *Let $P_{\text{SAS}}(i) = p_i + e_i$ and $\Delta_i = g + \beta p_i + u_i - e_i$, with latent capability p_i and independent mean-zero sampling errors e_i, u_i (variances σ_p^2, σ_e^2, σ_u^2). The least-squares slope of Δ_i on $P_{\text{SAS}}(i)$ converges to $\lambda\beta - (1-\lambda)$, where $\lambda = \sigma_p^2/(\sigma_p^2 + \sigma_e^2)$; on an index $x_i = p_i + e_i'$ built from other episodes, with error variance $\sigma_{e'}^2$, it converges to $\lambda'\beta$, where $\lambda' = \sigma_p^2/(\sigma_p^2 + \sigma_{e'}^2)$.*

The naive slope is therefore negative even when $\beta = 0$, the coupling artefact of change-score regression (Tu & Gilthorpe, 2007), while an out-of-domain index can hide a slope but never create one (proof and the fixed-effect case in Appendix K). The sign match, the bridge to the original 15 of 16, counts cells whose SAS CI does not straddle 45% and is secondary, since a sign is a discontinuous metric that can manufacture thresholds (Schaeffer et al., 2023).

H2. Stated as rates, because the share FP/(FP + WW) rises mechanically with the pass rate. H2a, on the SAS: WW/failures declines with the index while FP/passes does not. H2b, within cells: the paired change in WW, always read with the changes in IN, BX and FP, so that a drop in WW bought by passivity or exhaustion is read as one failure traded for another.

Validity (G1). A null H1 may mean the MAS is broken (Cemri et al., 2025), so every MAS cell is checked from logs, never from reward: calls cut at the cap ($\leq 5\%$), tokens in cut calls ($\leq 10\%$), no turn with two cut calls outside the final exhaustion turn, and no JSON or cut text sent to the customer; we also report how often write orders were executed exactly as issued. Exhaustion is an outcome, never a validity failure; only bug-class fixes are allowed, and a family that still fails leaves H1.

Robustness and power. R1 re-runs one cell below and one above 45% at other budgets with the SAS capped; R2 re-runs two airline cells, the straddling cell and one above 45%, with the benchmark's default simulator. A simulation gives power 0.94 at $\beta = -0.30$ and 0.74 at -0.20 for an 18-cell grid, so a null is read as $|\beta| < 0.20$; even at $\beta = -0.30$ the sign match reaches the original 15 of 16 in only 4% of worlds (Appendix J).

4 Results

4.1 The grid as run

Scope. Of the 18 cells, 16 were run; the airline and retail `qwen3.5-4b` cells were omitted for compute time.

Cells below 45%. Two cells are clearly below 45%, one straddles and thirteen are clearly above (Table 1). The weakest open model, `qwen3.5-4b` served at 4-bit, scores 94.7% on telecom, above

Table 1: **No cell shows a MAS benefit.** The grid in the cost-bounded arm ($B = 4B_{1\times}$, $c = B_{1\times}$, $r = 4$; 150 episodes per system), pass[1] in percent with task-cluster bootstrap CIs. Exh.: MAS episodes ending budget exhausted; util.: median share of the ceiling spent; fin./exh.: pass rates of MAS episodes that finished and that were exhausted; tokens: median MAS completion tokens per episode over the SAS median (the ceiling bounds this ratio at B over the SAS median, 5.0 to 6.3 across cells). †: fails G1, sensitivity only; SAS values computed by a collaborator.

Model	Domain	SAS	SAS 95% CI	MAS	Gain [95% CI]	Exh.	Util.	Fin./exh.	Tokens
`gpt-5-nano`	airline	42.7	[31.3, 54.0]$^{\circ}$	45.3	$+2.7$ [-6.7, $+12.7$]	23%	0.50	50% / 31%	2.9×
	retail	68.0	[58.7, 77.3]	44.7	-23.3 [-32.7, -13.3]	16%	0.63	52% / 8%	3.1×
	telecom	32.7	[22.7, 43.3]$^{\downarrow}$	19.3	-13.3 [-20.7, -6.0]	15%	0.54	20% / 13%	2.8×
`gpt-5-mini`	airline	74.7	[64.7, 84.0]	58.0	-16.7 [-27.3, -6.7]	34%	0.83	68% / 39%	5.2×
	retail	76.0	[66.7, 84.0]	48.0	-28.0 [-40.0, -17.3]	43%	0.90	68% / 22%	4.9×
	telecom	77.3	[68.0, 86.0]	66.0	-11.3 [-18.7, -4.0]	18%	0.70	79% / 7%	4.2×
`gemini-3.1-flash-lite`	airline	78.7	[68.0, 88.7]	62.7	-16.0 [-24.0, -8.0]	31%	0.71	85% / 13%	4.1×
	retail	77.3	[67.3, 86.7]	41.3	-36.0 [-47.3, -24.0]	58%	1.00	75% / 17%	5.2×
	telecom	34.0	[24.0, 44.7]$^{\downarrow}$	12.0	-22.0 [-30.7, -12.7]	45%	0.96	20% / 1%	5.8×
`gemini-3-flash`	airline	80.7	[70.7, 90.0]	55.3	-25.3 [-36.7, -14.0]	51%	1.00	74% / 37%	6.1×
	retail	80.0	[70.7, 88.7]	25.3	-54.7 [-66.0, -44.0]	67%	1.00	63% / 7%	5.1×
	telecom	91.3	[86.0, 96.0]	10.7	-80.7 [-87.3, -73.3]	68%	1.00	33% / 0%	5.4×
`qwen3.5-4b`	telecom	94.7	[90.7, 98.0]	23.3	-71.3 [-80.0, -62.7]	63%	1.00	54% / 5%	5.3×
`qwen3.5-9b`†	airline	79.3	[71.3, 86.7]	61.3	-18.0 [-28.7, -6.7]	12%	–	–	–
	retail	76.0	[66.7, 84.7]	49.3	-26.7 (no paired CI)	10%	0.51	55% / 0%	–
	telecom	95.3	[91.3, 98.7]	33.3	-62.0 (no paired CI)	33%	0.93	48% / 4%	–

$^{\downarrow}$ SAS CI entirely below 45%; $^{\circ}$ SAS CI straddles 45%; every other cell is entirely above. Airline and retail `qwen3.5-4b` were not run.

every closed model (`qwen3.5-9b` scores 95.3%), and adds no low cell. The principle's distinctive prediction, a gain below 45%, is therefore tested on two cells.

Validity. The twelve OpenAI and Google MAS cells and telecom × `qwen3.5-4b` pass the bars on calls cut, tokens in cut calls, turns with two cut calls and JSON or cut text sent to the customer (Appendix G). Write orders were executed exactly as issued in 11 to 46% of cases per cell (28% overall): the orchestrator plans without the tool schemas, which only the database agent holds, so many orders name a tool or an argument that does not exist; the database agent then called the valid tool or declined and reported back (Appendix G). We read this as part of what dividing the work costs in this architecture, and it limits what our results say about designs that share tool schemas (§5). `qwen3.5-4b` failed to parse 56% of its plan calls; excluding its cell leaves the pattern of every result unchanged (sign match 9 of 11, binomial $p = 0.065$; pooled WW change -0.7 points, CI $[-2.9, +1.3]$). The three `qwen3.5-9b` cells fail, all on turns with two cut calls and two on tokens in cut calls; a larger cap for one family would change the protocol, so they enter only a sensitivity. The `qwen3.5-4b` cell has no leave-one-domain-out index, so the primary fit uses 12 cells.

4.2 The grid: no cell shows a MAS benefit

Table 1 gives the grid before any regression. The MAS is below the SAS in 12 of the 13 valid cells, with a gain CI excluding zero in each, and ties in the straddling cell; the three invalid cells show the same sign. Within families the loss grows with the strength of the SAS, and the largest losses fall on the two cells where the SAS exceeds 90%. Scored against the SAS on the same task, exhausted MAS episodes carry 74% of the mean loss over the 13 valid cells (-22.4 of -30.5 points, an exact split; Appendix I). In ten cells the loss runs mainly through the budget, 60 to 90% of it via exhaustion: 18 to 68% of MAS episodes end exhausted and pass at 0 to 39%. On the three `gpt-5-nano` cells it runs through decisions: on retail and telecom, finished episodes carry 73% of the loss, and on telecom the MAS hands 120 customers to a human agent against 79. Per task, the MAS is better on 43 of 650 task-cell pairs and worse on 330, and no task improves for more than two of the four closed models.

4.3 Does the principle pick the right architecture?

Figure 2a and Table 2 test the principle on the axis it was stated on, each cell's own SAS pass[1]. No slope is drawn on this axis (Lemma 1). Of the 13 valid cells, 12 have a SAS CI that does not straddle 45%, and all 12 have a gain CI excluding zero. The principle predicts the observed sign in

Table 2: **Where the principle predicts a MAS benefit, the MAS loses under the ceiling and at best ties without one.** Top: the 45% rule applied to each valid cell's own SAS pass[1]. Bottom: the three cells at or below 45%, with and without a ceiling (uncapped arm). Index: SAS pass[1] on the model's other two domains; tokens: median MAS completion tokens per episode over the SAS median, without a ceiling.

SAS 95% CI relative to 45%	Cells	Principle predicts	Observed	Sign matches
entirely above	10	MAS loses	10 losses	10 of 10
entirely below	2	MAS gains	2 losses	0 of 2
straddles	1	(uninformative)	1 tie	–
all informative cells	12			10 of 12 (binomial $p = 0.039$)

Cell	SAS [95% CI]	Index	Gain at 4× [95% CI]	Gain uncapped [95% CI]	Tokens
telecom × `gemini-3.1-flash-lite`	34.0 [24.0, 44.7]	78.0	−22.0 [−30.7, −12.7]	+0.7 [−8.7, +9.3]	5.5×
telecom × `gpt-5-nano`	32.7 [22.7, 43.3]	55.3	−13.3 [−20.7, −6.0]	−10.0 [−18.7, −2.0]	2.6×
airline × `gpt-5-nano`	42.7 [31.3, 54.0]	50.3	+2.7 [−6.7, +12.7]	+1.3 [−7.3, +10.0]	2.7×

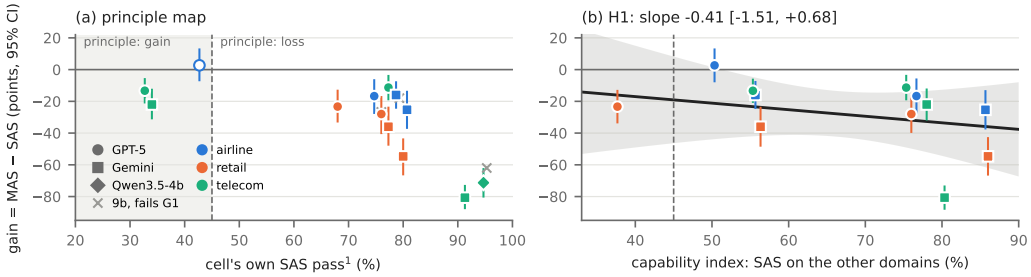

Figure 2: (a) Principle map: each valid cell's own SAS pass[1] against its gain with the paired task-cluster 95% interval; the principle predicts a gain left of the dashed 45% line and a loss right of it; the hollow marker is the one cell whose interval covers zero; grey crosses are the three `qwen3.5-9b` cells that fail validity. No regression is drawn on this axis. (b) H1: the same gains against the leave-one-domain-out index for the 12 cells of the primary fit, with the weighted fit at the mean family and domain effect and its 95% band. The two telecom cells left of 45% in (a) move right of it. Colour marks the domain and shape the family.

10 of 12, and where it errs matters more than how often. The ten matches are the cells above 45%, where any statement of the form "the SAS wins" would also be right; both cells below 45%, the only cells on which the principle says something distinctive, lose where it predicts a gain. Sweeping the threshold from 35 to 55% leaves this pattern unchanged. Removing the ceiling does not rescue the prediction (Table 2, bottom): on telecom × `gemini-3.1-flash-lite` it brings the MAS level with the SAS at 5.5 times the SAS's median token spend, and on the two `gpt-5-nano` cells it changes nothing. Neither this count nor the original 15 of 16 is powered to establish an accuracy, so our primary reading rests on the slope.

4.4 H1: NO DETECTABLE CAPABILITY DEPENDENCE

Table 3 and Figure 2b give the primary test. On the 12 cells with a leave-one-domain-out index (two families, two tiers, three domains; 7 residual degrees of freedom) the slope of gain on the index is -0.41 (95% CI $[-1.51, +0.68]$; permutation $p = 0.43$ two-sided, 0.21 one-sided), so H1 is not supported and, by our reading rule, the fitted line has no crossing. Every other form agrees in sign and detects no slope, including the naive form, which is biased toward one. No family-by-index interaction is detected (CI $[-3.07, +1.58]$).

This null cannot be read as $|\beta| < 0.20$, the bound our power analysis targeted. The cells scatter around the line far more than task sampling allows: the weighted residual variance is 18.6 times its sampling expectation. What the data do fix is the level: the fitted gain is negative across the observed

Table 3: **The gain shows no detectable dependence on capability, and its fitted level is negative across the observed range.** Left: the H1 slope (points of gain per point of index) under each form; the primary form carries the test (minimum detectable slope in our power analysis 0.20). [a]Other units. Right: the primary fit at the mean family and domain effect.

Form	Cells	Slope [95% CI]	Perm. p
leave-one-domain-out index (primary)	12	-0.41 [-1.51, $+0.68$]	0.43
split seeds	12	-0.44 [-0.95, $+0.07$]	–
logit gain[a]	12	-1.61 [-4.96, $+1.75$]	–
relative gain[a]	12	-0.35 [-1.40, $+0.70$]	–
primary with the `qwen3.5-9b` cells	15	-0.28 [-1.26, $+0.69$]	0.59
own SAS score (naive, biased)	12	-0.55 [-1.16, $+0.06$]	–

Index	Fitted gain [95% CI]
37.7% (lowest cell)	-16 [-50, $+18$]
45%	-19 [-46, $+8$]
60%	-25 [-41, -9]
86% (highest cell)	-36 [-62, -10]
weighted mean, 12 cells	-28 [-44, -12]

Table 4: **The MAS does not detectably reduce wrongful writes and converts other failures into exhaustion and inaction.** Pooled over the 13 valid cells (1,950 episodes per system); paired change per episode in points, cell-cluster bootstrap CI; classes assigned in the order WW, BX, IN, other. FP (airline only; 600 episodes per system) counts judge flags among passes; "all flags" adds omission flags, which were not adjudicated by hand, and is an upper bound.

Outcome	SAS	MAS	Change per episode [95% CI]	Cells down / up
pass	1,362	768	-30.5 [-43.6, -18.6]	12 / 1
wrongful write (WW)	55	43	-0.6 [-2.6, $+1.2$]	5 / 5
budget exhausted (BX)	0	666	$+34.2$ [$+24.5$, $+44.3$]	0 / 13
inaction (IN)	126	268	$+7.3$ [$+3.7$, $+11.8$]	1 / 10
other failure	407	205	-10.4 [-17.0, -4.3]	10 / 3
FP, commission (airline)	27 of 415 passes	17 of 332 passes	-1.7 (4 clusters, no CI)	4 / 0
FP, all flags (airline)	58 of 415 passes	57 of 332 passes	-0.2 (4 clusters, no CI)	2 / 2

index range and its interval excludes zero from 60% upward. The index also relocates the two cells the principle is distinctive about (Table 2, index 78.0 and 55.3%), so their low scores are a domain effect. The one cell whose index falls below 45%, retail × `gpt-5-nano` (37.7%), loses 23.3 points.

4.5 H2: WHAT COORDINATION CHANGES ABOUT FAILURE

A wrongful write is possible only where the gold trajectory has no write: 24 of 50 tasks on airline, 16 on telecom, 3 on retail. H2a fails on its wrongful-write half: across the 12 SAS cells WW/failures does not decline with the index (slope -0.15, 95% CI $[-0.64, +0.34]$, permutation $p = 0.78$); the rate is high only for `gpt-5-nano` on airline (33% of failures). Under the reading grid for H2b the MAS shows no detectable change in wrongful writes (Table 4), so we find no evidence that the architecture acts on the failures it targets. The failures it did change lie elsewhere: exhaustion and inaction rise, and other failures fall. The one cell where the MAS halves wrongful writes, airline × `gpt-5-nano` (28 to 14), is the tie of the grid, and there inaction and exhaustion rise by 35 episodes. On airline, the judge flags a commission, an action or offer the policy forbids, in 6.5% of SAS passes and 5.1% of MAS passes; about half of the pooled drop follows from the 83 fewer passes of the MAS, and with omission flags included the two systems are level. Commission flags fall with capability (15.6 to 2.5% of passes), against H2a's second half, on four cells of one domain.

4.6 ROBUSTNESS, SENSITIVITY AND DEPLOYMENT COST

R1 re-ran both `gemini-3.1-flash-lite` cells at 1× with the SAS capped too. Capping the SAS at its own 75th percentile costs it 4.0 and 6.7 points (telecom, airline), which sizes the asymmetry of §3.2. The gain is negative at both budgets (-27.3 and -24.0 at 1×, -22.0 and -16.0 at 4×), and its change between them has an interval covering zero on both cells (Appendix A), so no budget dependence is claimed; the larger budget cuts exhausted MAS episodes from 98% to 45 and 31% and lifts the MAS by 9 to 15 points, without closing the gap. R2 re-ran two airline cells with the benchmark's default simulator, `gpt-4.1`. On `gemini-3.1-flash-lite` the gain barely moves (-19.3 against -16.0; change -3.3, CI $[-11.3, +6.0]$). On `gpt-5-nano`, the straddling cell, the SAS falls to 37.3% (CI $[30.0, 45.3]$) and the MAS gains $+10.0$ (CI $[+1.3, +19.3]$), the

only gain interval above zero in the study; the change between simulators, $+7.3$ (CI $[-3.3, +18.0]$), covers zero, and it is one contrast among many with no multiplicity adjustment. In dollars the MAS costs 1.4 to 3.1 times the SAS per cell (2.3 to 4.0 times for the agent calls alone), and the SAS is more accurate and cheaper in 11 of the 12 priced cells, all but the straddling cell (Table 12). A strong SAS (S2) that runs the MAS's plan-and-review cycle in one context under the same ceiling (Appendix L) separates a loss from dividing the work from a loss from the scaffolding. On telecom × `gemini-3.1-flash-lite`, S2 scores 36.0% against 34.0% for the SAS ($+2.0$, CI $[-7.3, +12.0]$) and 12.0% for the MAS (MAS − S2 $= -24.0$, CI $[-33.3, -15.3]$): by the reading rule of Appendix L, the loss on this cell comes from dividing the work among agents.

5 Discussion and limitations

What the evidence says. The capability-saturation principle makes two claims: coordination helps while the SAS is weak, and its value declines as the SAS grows stronger. In dual control the first claim fails in the two cells that clearly test it; the one gain in the study appears on the straddling cell under a second simulator. The second claim is left undecided: the slope test detects no dependence, and its interval is too wide to exclude a steep decline, because model-by-domain effects dominate the variation between cells. The mechanism that motivates a MAS here, catching wrongful writes, shows no detectable effect. For systems that serve users under a written policy, the SAS was the better choice at every capability level we measured under a fixed simulator, and at a fraction of the cost. A MAS design for such settings therefore needs compute-matched evidence from the setting itself, since results from autonomous benchmarks do not carry over.

Why dual control penalises coordination. Control is shared with the user and the benchmark is half-duplex, so every tool call passes through the user-facing agent's turn and the orchestrator plans before each assignment. The user supplies information turn by turn, so the task cannot be decomposed in advance, and decomposable tasks are where coordination drew its largest benefits in the original study (Kim et al., 2026). The reward is a state at the end of a long conversation, so a MAS that spends its ceiling early fails outright.

Limitations. (1) One architecture is tested, and its orchestrator plans without the tool schemas that only the database agent holds, so 72% of its write orders were not executed as written; a null below 45% is stronger evidence against transfer than the original best-of-four validation gives, and a null above 45% is weaker. (2) Only two cells lie clearly below 45%, so the principle's distinctive prediction rests on two cells and the slope is identified mostly above the threshold. (3) The two `qwen3.5-4b` cells that were not run might have sat below 45% (Appendix N). (4) The user is a model, and simulators are more cooperative than people, so where a cell sits relative to 45% may differ with human customers; swapping the simulator already turns the straddling cell's tie into a 10-point MAS gain (R2). (5) The official reward is blind to verbal violations on airline and telecom; we measure this blind spot on airline only, commission flags are classified by a fixed text rule, and the judge was not checked against human labels. (6) The models postdate the benchmark's release and may have seen it in training, which inflates SAS and MAS scores alike. Appendix N lists more.

6 Conclusion

Scaling principles for agent systems have so far been derived without a user. We tested the most prominent of them in the dual-control regime: a centralized MAS built from the same model showed no benefit over the stock SAS in any of 13 valid cells under a fixed simulated user, lost where the principle predicts a benefit, drew level at best without a budget ceiling, and did not detectably reduce the wrongful writes it was designed to catch. For agents that serve users, the burden of proof lies with the multi-agent design. We release every trajectory, ledger, prompt and our code so that the test can be rerun on the next architecture, benchmark and principle.

AI use statement

In this work we used generative AI tools (large language model assistants) for literature search and summarisation; for critiquing the research design and proposing revisions to it, some of which

the authors adopted (for example, the zero-crossing location test); for writing and reviewing the experiment runner, budget layer, agent and analysis code; for drafting and editing text; and for computing and checking summary statistics from the run records. The research question and every design decision are the authors'. The language models under study are also the experimental subjects, and their outputs are the data. We have reviewed all AI-assisted work: AI-assisted code was read and tested by the authors (the released test suite of 60 offline tests against the pinned benchmark, and the meter and billing checks of Appendix E); every number in the text and tables was checked by the authors against the run manifests and results files where held; AI-drafted text was revised by the authors, and the literature it cites was verified against the primary sources. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

Ethics statement

The study involves no human participants: every customer is a language-model simulator and every database is the benchmark's synthetic data, so the released trajectories contain no personal information. The agents under study act under written policies, and the work measures, among other things, how often they violate those policies in words or in writes; the findings bear on the deployment of language-model agents in customer-facing roles and are reported with the limitations of simulated users stated (§5). Total spend on commercial model APIs for the runs reported here was about US$453 at list prices (Appendix M); the smallest open-weight model was run on a laptop. The authors declare no conflict of interest and no sponsorship from any model provider.

Reproducibility statement

The benchmark is pinned to one commit with the evaluator and domain data unmodified, and the task subsets, root seed, model snapshots, routes, reasoning settings and simulator are recorded with hashes (Appendix B). Every prompt (the base prompt, the three role blocks, the wrap-up notice and the exhaustion message) is released with its SHA-256, and every run writes a manifest carrying those hashes together with the calibration values it ran under (Appendix B, Table 7). The budget protocol is specified rule by rule with the mechanism and test that implement each (Appendix C). The power simulation, all trajectories with per-call token records and budget ledgers, the calibration files, and the code for the agents, runners and analysis are included in the anonymised supplementary material submitted with this paper and will be released publicly on acceptance.

References

Anthropic. (2025). *How we built our multi-agent research system.* Engineering blog. Retrieved from `https://www.anthropic.com/engineering/multi-agent-research-system`

Barres, V., Dong, H., Ray, S., Si, X., & Narasimhan, K. (2025). τ^2-bench: Evaluating conversational agents in a dual-control environment. *arXiv preprint arXiv:2506.07982*.

Cemri, M., et al. (2025). Why do multi-agent LLM systems fail? *arXiv preprint arXiv:2503.13657*.

Cuadron, A., Yu, P., Liu, Y., & Gupta, A. (2025). SABER: Small actions, big errors: Safeguarding mutating steps in LLM agents. *arXiv preprint arXiv:2512.07850*.

Fourney, A., et al. (2024). Magentic-one: A generalist multi-agent system for solving complex tasks. *arXiv preprint arXiv:2411.04468*.

Gao, M., et al. (2025). Single-agent or multi-agent systems? Why not both? *arXiv preprint arXiv:2505.18286*.

Jwalapuram, P., et al. (2026). The illusion of multi-agent advantage. *arXiv preprint arXiv:2606.13003*.

Kang, S., Yu, T., & Hwang, S. J. (2026). PolicyGuard: A dialogue-grounded sub-agent verifier for policy adherence in LLM agents. *arXiv preprint arXiv:2606.29225*.

Kapoor, S., et al. (2025). Holistic agent leaderboard: The missing infrastructure for AI agent evaluation. *arXiv preprint arXiv:2510.11977*.

Kapoor, S., Stroebl, B., Siegel, Z. S., Nadgir, N., & Narayanan, A. (2024). AI agents that matter. *arXiv preprint arXiv:2407.01502*.

Kim, Y., et al. (2026). Capable language models can outgrow the benefits of collaboration. *Nature Machine Intelligence*, *8*, 1157–1172. (Preprint: Towards a Science of Scaling Agent Systems, arXiv:2512.08296. Code: Zenodo 10.5281/zenodo.20144433) doi: 10.1038/s42256-026-01268-y

LangChain. (2025, June). *Benchmarking multi-agent architectures.* Blog post. Retrieved from `https://www.langchain.com/blog/benchmarking-multi-agent-architectures`

Li, J., et al. (2026). Scaling behavior of single LLM-driven multi-agent systems. *arXiv preprint arXiv:2606.00655*.

Schaeffer, R., Miranda, B., & Koyejo, S. (2023). Are emergent abilities of large language models a mirage? In *Advances in Neural Information Processing Systems.* (arXiv:2304.15004)

Seshadri, P., Cahyawijaya, S., Odumakinde, A., Singh, S., & Goldfarb-Tarrant, S. (2026). Lost in simulation: LLM-simulated users are unreliable proxies for human users in agentic evaluations. *arXiv preprint arXiv:2601.17087*.

Tran, D., & Kiela, D. (2026). Single-agent LLMs outperform multi-agent systems on multi-hop reasoning under equal thinking token budgets. *arXiv preprint arXiv:2604.02460*.

Tu, Y.-K., & Gilthorpe, M. S. (2007). Revisiting the relation between change and initial value: a review and evaluation. *Statistics in Medicine*, *26*(2), 443–457. doi: 10.1002/sim.2538

Wang, J., et al. (2024). Reasoning in token economies: Budget-aware evaluation of LLM reasoning strategies. In *Proceedings of EMNLP.* (arXiv:2406.06461)

Yang, Y., et al. (2025). Revisiting multi-agent debate as test-time scaling: A systematic study of conditional effectiveness. *arXiv preprint arXiv:2505.22960*.

Yao, S., Shinn, N., Razavi, P., & Narasimhan, K. (2024). τ-bench: A benchmark for tool-agent-user interaction in real-world domains. *arXiv preprint arXiv:2406.12045*.

Yu, W., et al. (2026). When do multi-agent systems help? An information bottleneck perspective. *arXiv preprint arXiv:2607.16133*.

Zhou, X., et al. (2026). Mind the Sim2Real gap in user simulation for agentic tasks. *arXiv preprint arXiv:2603.11245*.

Zhu, Y., et al. (2025). Establishing best practices for building rigorous agentic benchmarks. *arXiv preprint arXiv:2507.02825*.

A TESTS AND RESULTS CARD

Table 5 collects every test in the paper with the rule used to read it and its result, so that each claim in the main text can be traced to one row.

Table 5: Tests and results. Each row gives a test, the rule used to read it and the result.

Row	Unit; contrast; test	Reading rule	Result
H1 slope (primary)	valid cells; gain on leave-one-domain-out index, family and domain effects, inverse-variance weights; permutation p	supported if slope < 0 with CI excluding zero; null read as $\|b\| < 0.20$	-0.41 $[-1.51, +0.68]$, $p = 0.43$; not supported; CI far wider than the minimum detectable slope of 0.20
H1 crossing	same fit; $x_0 = -\text{intercept}/\text{slope}$; bootstrap CI	transfers if CI covers 45%; shifts if not; no crossing if slope CI covers zero	no crossing (fitted gain < 0 over the whole index range)
Sign match	cells not straddling 45%; predicted vs observed sign; binomial vs 0.5	X/N beside 15/16; informative and within-noise cells apart	10 of 12 informative ($p = 0.039$); right on all 10 above 45%, wrong on both below
H2a (primary)	SAS cells; WW/failures and FP/passes on the index	WW/failures slope < 0, FP/passes slope not < 0	WW half: -0.15 $[-0.64, +0.34]$, $p = 0.78$; not supported; FP half (airline, 4 cells, descriptive): commission flags per pass fall from 15.6 to 2.5%
H2b (primary)	paired MAS$-$SAS WW, pooled, cluster CI	grid over WW, IN, BX, FP	WW -0.6 pts $[-2.6, +1.2]$: no detectable change; BX $+34$, IN $+7$ pts; FP commission 6.5 to 5.1% of passes
R1	2 cells, gain across budget levels	dependence only if its CI excludes zero	$\text{gain}_{4\times} - \text{gain}_{1\times}$ $+5.3$ $[-2.7, +14.0]$ and $+8.0$ $[-1.3, +18.0]$: not claimed
R2	2 airline cells, 2 simulators	simulator effect if the change in gain excludes zero	not detected: change $+7.3$ $[-3.3, +18.0]$ (`gpt-5-nano`, whose gain is $+10.0$ $[+1.3, +19.3]$ under `gpt-4.1`) and -3.3 $[-11.3, +6.0]$ (`gemini-3.1-flash-lite`)
G1	every MAS cell; mechanical bars	valid or protocol failure	13 of 16 valid; three `qwen3.5-9b` cells fail; write orders executed as issued in 11 to 46% of cases per valid cell
Cell coverage	SAS CI vs 45%	≥ 3 below and ≥ 3 above for a balanced test	2 below, 1 straddling, 13 above
Uncapped arm (sec.)	3 cells, no ceiling	(1) ceiling hid a benefit; (2) it did not	(2) met
S2 (ext.)	1 cell (telecom $\times$ `gemini-3.1-flash-lite`); strong SAS at $4\times$ against the SAS and the MAS	(1) S2 $-$ SAS CI < 0: scaffolding loses too; (2) S2 $-$ SAS covers zero or above and MAS $-$ S2 < 0: loss from the division; (3) mixed	(2): S2$-$SAS $+2.0$ $[-7.3, +12.0]$; MAS$-$S2 -24.0 $[-33.3, -15.3]$

B BENCHMARK RECORD

τ^2-bench v1.0.1, commit `c3398666e6559e3a063da3fc04b5acf7f941464e`; `git diff` against `src/tau2/evaluator/` and `data/tau2/domains/` is empty; the only dirty file is one version line in `uv.lock`. Our analysis code never imports the evaluator; its pass^k reimplementation was checked against the official `compute_metrics` at the pinned commit. Re-evaluating 300 baseline episodes with the assertion judge forced on reproduced every original reward (`reward_mismatches: []`). Reproduction gate: airline $\times$ `gemini-3-flash-preview` (effort high) with `gpt-5.2` (low) as the customer, 3 trials, seed 300, gave pass^1 80.7% against the public leaderboard's 82.5% for the same configuration (4 trials, seed 123, evaluated 2 March 2026); pass^3 74.0% against 72.0%. Run date 27 August 2026. Task subsets: airline all 50 base tasks (id-list SHA-256 `696d5a08...`); retail the first 50 of the 114 base ids after one `random.Random(300).shuffle` (`5047ad30...`); telecom likewise (`f9c34039...`); the id lists are in `task_subsets.json`. Reward basis per domain: airline DB and COMMUNICATE; retail DB and NL_ASSERTION; telecom ENV_ASSERTION (43 tasks) or ENV_ASSERTION and ACTION (7 tasks). Known task issues: one airline task is flagged by the judge in both baseline runs independently of the agent model (a task-level trap, reported separately); the benchmark family has documented impossible tasks with empty goal states (Zhu et al., 2025) and later community corrections (Cuadron et al., 2025), none of which are applied, so all SAS baselines are generated internally on the pinned version and no leaderboard number enters a test. Model snapshots and routes: `gpt-5-nano-2025-08-07` and `gpt-5-mini-2025-08-07` (OpenAI, Responses API; `reasoning_effort: high`; the API drops `seed`); `gemini-3.1-flash-lite` and `gemini-3-flash-preview` (Gemini API, `thinking_level: high`); `qwen3.5:4b` Q4_K_M, Ollama 0.34.2, digest `2a654d98e6fb`, context 131,072, shipped sampling defaults, served one request at a time on a laptop; `qwen/qwen3.5-9b` through a hosted router at temperature 0 with `reasoning.effort: high`, served build not pinned. Simulator `gpt-5.2-2025-12-11`, `reasoning_effort: low`. Max steps 200, max errors 10.

Infrastructure-error episodes are re-run once and reported; two were re-run in the `qwen3.5-4b` SAS cell (a tool-call parse rejection by the local server and one empty reply), and every MAS cell reported here finished with none.

Prompts and manifests. The budget protocol, the provenance record, the run configuration (`cells.json`, `budgets.json`, `task_subsets.json`) and the prompts are released with the code. Prompt hashes (SHA-256, file as loaded): base prompt template, the benchmark's default agent prompt at commit `c3398666` with the policy placeholder, `825bb492ecd9062046cc7de0050b31ae7887e3577b66b149883728e71a193da7`; MAS orchestrator role block v6, `1a26e997efe8928865330dd5ecc5409964402842aeac666239108e04b545e000`; MAS database-agent block v2, `367b9355faf9a0063ddaee28303e4e951ff1a032261556d5b379d2f1a76fffc8`; MAS policy-advisor block v4, `a350bd2cec205373fcf25403ed7606eb54b9a877b35bb0f9ebcfc3641dc06186`; wrap-up notice, `f48a23bee3300d0afd2fa6cbdbe7c77f6e069c10b3082572c7cee8a47feacb21`; exhaustion message, `f550c541217dd85534417b4d8262b66ab29ae723c289c9ce9009e1ecdc1e7089`. The uncapped arm uses the orchestrator block with its one allowance paragraph removed (block as sent, `54af48966c06cb4f70731158d4a2b5d129eb40ca1b3f1b6f88bb01d3e2f4b63d`). The S2 role block, v2 as sent after the first smoke, is `ba0af7c798387e5d6958949ac91caafe5b28f99dde6c7aadc26de211988e5973`; it was kept unchanged after a second smoke round (file SHA-256 `1345572de1dc2d86c76ca4d3c4b58a53403d2bb377d923c8dba41437207444f9`). Every manifest written by our runners records these digests, the benchmark commit, the `cells.json` hash, the task-id hash and the budget values in force; the collaborator's `qwen3.5-9b` manifests use a different schema without the last three.

C MAS CONTROL FLOW AND BUDGET PROTOCOL

Why centralized. The other generic architectures of Kim et al. (2026) need an adaptation that would itself explain the result: in an independent MAS each agent would hold its own conversation with the customer, decentralized debate between customer turns would be confounded by the debate protocol, and hybrid is centralized plus lateral messages.

Control flow. Figure 1 shows the two systems. At the start of an episode one ledger is created with limit $B = 4B_{1\times}$, per-call cap $c = B_{1\times}$, wrap-up threshold w and minimum allowance $a_{\min}$. When a customer message or a database-agent report arrives, the orchestrator makes a *plan* call without tools (max_tokens $= \min(\text{remaining}, c)$), returning a JSON object with the task-ledger and progress-ledger updates, an optional advisor question and a draft (customer message, read assignment or write assignment). If a question is present the advisor is called with a fresh context and its answer appended, and the orchestrator makes an *act* call that emits the final message or assignment; otherwise the draft is final. A customer message ends the turn. A read assignment hands control to the database agent, which calls tools by native function calling; tool results route to it until it reports or reaches $r = 4$ calls, at which point a report is forced. A write assignment is one call whose executed tool name and arguments are compared with the order and logged (exact, exact after type normalisation, argument mismatch, tool mismatch); a write attempted inside a read assignment is blocked. Every call reserves before dispatch and commits actual usage after; usage is charged before tool-call parsing, so a call cut mid-arguments is charged and its tool calls dropped. A plan that does not parse is retried once, then replaced by a forced customer message; a plan cut at c is not retried. When the remainder falls below w the wrap-up notice is appended once to the system message and only customer messages are allowed; when it falls below $a_{\min}$ the fixed exhaustion message is sent with the label `budget_exhausted` and the episode ends (the benchmark records `agent_stop`; the analysis reads the label). The simulator's calls are outside the ledger.

Rule-to-mechanism mapping. Each rule of §3.2 is implemented by one mechanism with a named offline test: one ledger per episode (`EpisodeBudget` in the agent state; every provider call passes through `metered_generate`); reservations bounded by the remainder; per-call limit $\min(\text{reservation}, c)$; wrap-up notice once, on a single system message; no tools after wrap-up; truncated calls charged and never executed; the exhaustion label distinct from inaction; provider overrun recorded; utilisation, wrap-up, exhaustion and truncation rates computable from every episode's snapshot; base-prompt injection with a hash check; MAS routing as above. The suite holds 60 tests at

Table 6: Relationship to prior budget protocols.

Protocol	Adopted	Deviation here
Kim et al. (2026)	per-system ceiling; coordination paid from the same pool; exhaustion is a legal termination	their ceiling is an iteration count with tokens matched only as a realised mean; here a per-episode token ceiling, enforced
Tran and Kiela (2026)	the SAS spends the whole budget in one trajectory; a MAS divides it	they keep planner and aggregator near budget-neutral, here every component is charged; they count thinking tokens, here completion tokens with reasoning-only as sensitivity; they split statically (B/k), here a shared pool with reservations, which is more generous to the MAS, so a null MAS result is conservative
Wang et al. (2024)	performance-at-tokens curves	their cap is input plus output per question; here input is reported as a deployment account outside the scientific budget
Kapoor et al. (2024)	accuracy-versus-USD frontier for deployment claims	not used for the main contrast, where compute and architecture would co-vary

the pinned commit on the benchmark's mock domain, including one full MAS episode, one capped SAS episode and one S2 episode through the benchmark's own runner. The uncapped arm sets no limit and no wrap-up, keeps c, and removes the orchestrator's allowance paragraph; a test checks that the prompt difference is exactly that paragraph.

D CALIBRATION TABLE

Table 7 gives, for every grid cell, the SAS spend that set the budget and the budget values the MAS ran under.

Table 7: Calibration per cell from the 150 uncapped SAS episodes: mean and 75th percentile of episode completion tokens ($B_{1\times}$), the ceiling $B = 4B_{1\times}$, the per-call cap as run ($c = B_{1\times}$), the SAS per-call 95th percentile for reference, and the results-file SHA-256 prefix. The wrap-up threshold is $w = \max(0.15B_{1\times}, 2\,\text{median})$ and the minimum allowance $a_{\min}$ = median of SAS per-call completion tokens (full values in `budgets.json`).

Domain	Model	SAS mean	$B_{1\times}$ (P_{75})	$B = 4B_{1\times}$	c	per-call P_{95}	SHA-256
airline	`gpt-5-nano`	40,839	51,297	205,188	51,297	11,566	`a0e882bf6ac0`
retail	`gpt-5-nano`	29,783	35,313	141,252	35,313	8,897	`03c84f443dc2`
telecom	`gpt-5-nano`	60,570	75,032	300,128	75,032	6,100	`6e0502f5595d`
airline	`gpt-5-mini`	20,808	27,477	109,908	27,477	8,312	`0993b01c8c63`
retail	`gpt-5-mini`	14,432	18,294	73,176	18,294	5,389	`e2bafb2beaa6`
telecom	`gpt-5-mini`	26,165	31,514	126,056	31,514	5,273	`d4be39972133`
airline	`gemini-3.1-flash-lite`	4,997	6,159	24,636	6,159	1,662	`3fc369bc1759`
retail	`gemini-3.1-flash-lite`	3,464	4,186	16,744	4,186	949	`0c9c773e4632`
telecom	`gemini-3.1-flash-lite`	5,272	7,123	28,492	7,123	1,326	`4c7b844471fc`
airline	`gemini-3-flash`	5,687	6,672	26,688	6,672	1,813	`e188ae2d1c16`
retail	`gemini-3-flash`	3,471	4,161	16,644	4,161	1,072	`836e2a05ad11`
telecom	`gemini-3-flash`	7,242	8,954	35,816	8,954	1,355	`8447013d0040`
telecom	`qwen3.5-4b`	4,335	5,630	22,520	5,630	608	`37957ae5a522`
airline	`qwen3.5-9b`	9,624	12,263	49,052	12,263	3,243	`856e06c713a6`
retail	`qwen3.5-9b`	5,367	6,370	25,480	6,370	1,808	`ce1270aa0d73`
telecom	`qwen3.5-9b`	4,514	5,758	23,032	5,758	772	`f1fbe345418f`

E METER VALIDITY

Checked on the gate run before any budget was set. Reasoning-token visibility: on all 1,614 Gemini agent calls carrying a details block, reasoning + text = completion exactly (698,493 reasoning and 144,363 text tokens; reasoning is 82% of output); `gpt-5.2` likewise (10,466 reasoning inside 59,538 completion). 123 calls (7%) carry no details block, covering 1.2% of output tokens, which the primary

meter counts and a reasoning-only meter would miss. Cached-input accounting: cached + text = prompt per call; recomputing cost at three rates reproduces the recorded totals to the cent for both providers, whereas flat input pricing would overstate the Gemini agent cost by 42%. Billing: recorded \$6.8699 against \$6.86 billed (Google, -0.1%) and \$1.6745 against \$1.69 (OpenAI, $+0.9\%$). Regrade determinism: 300 rewards reproduced exactly. Task-set identity: byte-identical task definitions and assertion lists across runs. Input-to-output ratio: about 60:1 for the non-reasoning pilot model and 16.9:1 for the reasoning gate run, so the completion-token meter diverges from total tokens far more for non-reasoning models; both are reported. On the local Ollama route the completion count includes thinking tokens and the thinking trace never reaches the customer; on the hosted router of the 9-billion model the per-call limit applied to visible text only (§4.1).

F CLASSIFICATION OF CELLS

The 16 measured grid cells are classified in Table 1 (SAS pass[1] with task-cluster CI; below, straddling or above 45%). Telecom separates the models most: the two low-tier closed models sit at 32.7 and 34.0%, `gpt-5-mini` at 77.3%, and `gemini-3-flash` and both open models at 91.3 to 95.3%.

G VALIDITY INDICATORS PER MAS CELL

Table 8: G1 indicators per MAS cell at $4\times$, computed from logs. Bars: calls cut at the cap $\leq 5\%$; tokens in cut calls $\leq 10\%$; no turn with two or more cut calls outside the final exhaustion turn (shown as exhaustion / other); no JSON or cut text sent to the customer. Reported indicators: episodes that consulted the advisor, plan calls that did not parse, forced customer messages, and the share of write orders executed exactly as issued. "–": not available from the collaborator's files.

Model	Domain	Cut calls	Cut tokens	Double-cut (exh. / other)	JSON, cut text	Advisor (of 150)	Plan-parse fail	Forced msgs	Orders as issued	Verdict
`gpt-5-nano`	airline	0.3%	0.3%	0 / 0	0	17	48 of 1,384 (3.5%)	31	22%	valid
	retail	0.4%	0.5%	0 / 0	0	13	15 of 1,334 (1.1%)	43	11%	valid
	telecom	0.3%	0.5%	0 / 0	0	9	30 of 1,861 (1.6%)	51	22%	valid
`gpt-5-mini`	airline	0.9%	0.9%	0 / 0	0	21	46 of 1,177 (3.9%)	40	46%	valid
	retail	1.3%	1.4%	0 / 0	0	7	58 of 1,210 (4.8%)	54	33%	valid
	telecom	0.5%	0.5%	0 / 0	0	8	32 of 1,685 (1.9%)	25	36%	valid
`gemini-3.1-flash-lite`	airline	1.8%	2.5%	0 / 0	0	93	37 of 1,009 (3.7%)	38	13%	valid
	retail	3.5%	6.7%	2 / 0	0	106	68 of 1,028 (6.6%)	78	15%	valid
	telecom	1.8%	2.0%	0 / 0	0	130	39 of 967 (4.0%)	39	25%	valid
`gemini-3-flash`	airline	1.9%	3.0%	1 / 0	0	147	48 of 979 (4.9%)	46	40%	valid
	retail	3.3%	5.8%	1 / 0	0	150	46 of 878 (5.2%)	42	19%	valid
	telecom	1.7%	1.5%	0 / 0	0[a]	150	66 of 1,102 (6.0%)	64	30%	valid
`qwen3.5-4b`	telecom	2.3%	8.0%	0 / 0	0[a]	14	1,386 of 2,467 (56.2%)	1,076	34%	valid
`qwen3.5-9b`	airline[b]	–	36%	14 (split –)	0	–	492 of 1,483 (33%)	–	–	fails
	retail	–	9.6% (13.2% excl. runaways)	1 / 4	0	59	352 of 1,414 (25%)	144	–	fails
	telecom	3.2% of plan calls	12.9%	5 / 6	0	52	292 of 2,138 (14%)	122	–	fails

[a] One `cut text` flag raised by the checker, inspected and found to be a false positive: a plan call with `finish_reason: length` and a complete JSON object; the customer message was complete. [b] Run by a collaborator on the same configuration at benchmark commit `8b57085` against the pinned `c339866`. Write orders: 2,771 closed write assignments over the 13 valid cells; 28% executed exactly as issued, 36% executed with a different tool or arguments (mostly a valid tool in place of an order that named a tool or argument that does not exist), 35% not executed and reported back, 0.6% not verifiable.

The read-call limit $r = 4$ was reached on 35 / 26 / 17% of read orders (airline / retail / telecom) for `gemini-3.1-flash-lite`, 50 / 36 / 51% for `gemini-3-flash`, 68 / 55 / 44% for `gpt-5-mini` and 30 / 72% for `gpt-5-nano` on retail / telecom. The 9-billion model's hosted route returned more than `max_tokens` on 17 (telecom) and 12 (retail) calls, of which 4 and 6 were runaway calls of about 131k reasoning tokens that exhausted the episode in one call, while the collaborator's retail replication had 6 such calls and no runaway, so the route's handling of the reasoning budget varies by run; pass[1] excluding those episodes is 34.2% [27.0, 42.3] on telecom and 51.4% [43.3, 59.4] on retail. Retail $\times$ `qwen3.5-9b` run twice with the same agent settings and budgets (ours at concurrency 4 on the pinned commit; the collaborator's at concurrency 1 on commit `8b57085`) gave 49.3% and 51.3%.

Read-call limit. The database agent's limit of $r = 4$ bound on 17 to 72% of read orders in the 11 cells where it was counted. Re-running two telecom cells at $r = 25$ left both gains inside their $r = 4$ intervals (-21.3 against -22.0; -77.3 against -80.7) and plan calls per episode flat, so the overhead sits in the orchestrator's planning.

H FAILURE TAXONOMY AND JUDGE RULES

Per episode and condition: *WW*, wrongful write, a write the gold trajectory does not contain (visible to the primary measure); *IN*, inaction, the gold trajectory requires a write and the agent made none; *BX*, budget exhausted, the episode ended on the ledger's exhaustion path (kept apart from IN; the inaction rate is reported with and without it); *FP*, a passing episode that the assertion judge flags for a policy violation in words (defined for airline, where the assertions are outside the official score; telecom tasks carry none and on retail they are inside the score); *order misexecuted* (MAS episodes only), a write whose executed tool name or arguments differ from the orchestrator's assignment, computed from logs; *other*. Write tools are read from the benchmark's own `WRITE` tags; only agent-side calls count (telecom customers call tools too); gold actions are filtered to the assistant requestor. The judge (`gpt-4.1-2025-04-14` at temperature 0, the benchmark's pinned default) is validated as an instrument: a full-population regrade flagged exactly the same 12 of 121 passing gate-run episodes as an earlier passing-only screen; assertion violations are about 6.5 times more frequent in failing than in passing gate-run episodes (65.5% against 9.9%), so the ungraded dimension is concordant with the graded one; commission-type flags are taken as-is, while omission-type flags ("did not cite the status as the basis") were not adjudicated by hand and enter only the upper-bound count; the judge was not checked against human labels. The measured blind spot of the primary measure on the two anchor runs is 15.0% of passing episodes at a 40% baseline and 9.9% at 80.7% (upper bounds). Anchor values for H2a from the two stock-prompt baseline runs: WW/failures 41% at a 40.0% baseline and 7% at 80.7%; FP/passes 15% and 10%.

As computed for H2. Our extraction script applies the definitions of `analyze_run.py` to every episode of the 13 valid cells: write tools from the benchmark's own `WRITE` tags, a task needs a write if its gold actions contain an assistant-side write, and writes are the agent-side write calls in the trajectory (for the MAS, the database agent's executed calls). Failing episodes are coded in order: WW (the gold trajectory has no write and the agent wrote), BX (the exhaustion label), IN (the gold needs a write and the agent made none), OTHER. The taxonomy reproduces our earlier count for the gate run (2 wrongful writes among 29 failures). Tasks with no gold write, where WW is possible: airline 24, telecom 16, retail 3 of 50. The telecom tasks carry no natural-language assertions, so FP is defined on airline only (retail's assertions are inside the official score). Table 9 gives the counts.

FP, as computed. `nl_screen.py` ran the benchmark's own assertion judge (`gpt-4.1`, temperature 0) on every passing episode of the four airline cells, SAS and MAS, on 26 September; the gate run's earlier screen supplies airline × `gemini-3-flash` SAS. Of 31 distinct unmet assertions, those whose text begins "Agent does not / should not / must not / should never / never / should refuse" are commission-type and count as FP as-is; the rest (duties to detect, verify, identify or communicate) are omission-type and enter only after hand adjudication, which has not been done, so the all-flags count is an upper bound. Per cell, commission / all flags among passes: `gpt-5-nano` 10 / 20 of 64 (SAS) and 9 / 22 of 68 (MAS); `gemini-3.1-flash-lite` 6 / 12 of 118 and 1 / 10 of 94; `gpt-5-mini` 8 / 14 of 112 and 6 / 17 of 87; `gemini-3-flash` 3 / 12 of 121 and 1 / 8 of 83. Pooled change per episode, commission -1.7 points and all flags -0.2 points; with four clusters no interval is reliable, and part of the commission drop reflects the MAS's fewer passes.

I LOSS DECOMPOSITION AND PER-TASK COUNTS

For each MAS episode on task t we take its reward minus $\bar{R}_t^{\mathrm{SAS}}$, the SAS pass rate on the same task, and average over the MAS episodes of the cell; the average equals the cell's gain. Grouping the terms by whether the episode ended on the exhaustion path splits the gain exactly into the part carried by exhausted episodes and the part carried by finished ones (Table 10). The two classes are outcomes of the MAS, so the split locates the loss and does not identify its cause. Per-task counts compare the MAS and SAS pass rates on each task over the three seeds. Of the 150 tasks (50 per domain), 33 improve under the MAS for one of the four OpenAI and Google models, 5 for two and none for three or more, while 18 get worse for all four.

Table 9: Failure taxonomy per valid cell (episodes out of 150): passes, wrongful writes, budget exhausted, inaction, other.

		SAS					MAS				
Model	Domain	PASS	WW	BX	IN	OTHER	PASS	WW	BX	IN	OTHER
gpt-5-nano	airline	64	28	0	10	48	68	14	23	22	23
	retail	102	0	0	16	32	67	0	22	23	38
	telecom	49	12	0	8	81	29	5	19	51	46
gpt-5-mini	airline	112	5	0	12	21	87	0	31	24	8
	retail	114	0	0	8	28	72	0	51	17	10
	telecom	116	1	0	10	23	99	2	24	10	15
gemini-3.1-flash-lite	airline	118	0	0	13	19	94	1	39	12	4
	retail	116	1	0	10	23	62	0	72	10	6
	telecom	51	0	0	18	81	18	5	62	42	23
gemini-3-flash	airline	121	2	0	11	16	83	0	48	17	2
	retail	120	0	0	8	22	38	0	94	16	2
	telecom	137	2	0	2	9	16	11	95	17	11
qwen3.5-4b	telecom	142	4	0	0	4	35	5	86	7	17
Total		1362	55	0	126	407	768	43	666	268	205

Table 10: Where the loss arises, per valid cell at $4\times$ (points). The exhausted and finished parts sum to the gain up to rounding; share: part of the loss carried by exhausted episodes, shown where the gain is negative. Tasks: number of the 50 tasks on which the MAS pass rate is above, equal to or below the SAS pass rate.

Model	Domain	Gain	Exhausted	Finished	Share	Tasks better / equal / worse
gpt-5-nano	airline	$+2.7$	$+2.0$	$+0.7$	–	13 / 26 / 11
	retail	-23.3	-6.2	-17.1	27%	2 / 23 / 25
	telecom	-13.3	-3.6	-9.8	27%	3 / 28 / 19
gpt-5-mini	airline	-16.7	-11.8	-4.9	71%	6 / 27 / 17
	retail	-28.0	-22.9	-5.1	82%	5 / 19 / 26
	telecom	-11.3	-8.7	-2.7	76%	5 / 25 / 20
gemini-3.1-flash-lite	airline	-16.0	-11.8	-4.2	74%	0 / 36 / 14
	retail	-36.0	-32.2	-3.8	90%	3 / 20 / 27
	telecom	-22.0	-13.1	-8.9	60%	4 / 23 / 23
gemini-3-flash	airline	-25.3	-17.6	-7.8	69%	2 / 30 / 18
	retail	-54.7	-48.7	-6.0	89%	0 / 15 / 35
	telecom	-80.7	-60.7	-20.0	75%	0 / 1 / 49
qwen3.5-4b	telecom	-71.3	-56.2	-15.1	79%	0 / 4 / 46
Mean (total for tasks)		-30.5	-22.4	-8.1	74%	43 / 277 / 330

J POWER ANALYSIS

Output of `power_sim.py` (400 worlds per setting, seed 1, 2 September 2026), task heterogeneity fitted to the pilot and gate runs (beta-binomial: pilot mean 0.400, $\kappa = 1.39$; gate mean 0.807, $\kappa = 0.42$; $\kappa = 0.91$ used). Power is the fraction of worlds in which the slope CI excludes zero on the negative side; the $b = 0$ column is the false-positive rate.

Table 11: Power of the H1 slope test by form of x and grid size (simulation).

x	Scenario	$b = 0$	-0.10	-0.20	-0.30	-0.40	acc. @-0.3	inform. @-0.3
naive	27 cells, 50 tasks, 3 seeds	0.19	0.78	1.00	1.00	1.00	0.84	0.39
naive	18 cells, 50 tasks, 3 seeds	0.20	0.67	0.99	1.00	1.00	0.87	0.49
split seed	27 cells, 50 tasks, 3 seeds	0.07	0.48	0.87	0.99	1.00	0.82	0.32
split seed	18 cells, 50 tasks, 3 seeds	0.07	0.35	0.80	0.97	1.00	0.86	0.42
leave-one-domain-out	27 cells, 50 tasks, 3 seeds	0.07	0.40	0.77	0.95	1.00	0.84	0.38
leave-one-domain-out	27 cells, 50 tasks, 5 seeds	0.10	0.51	0.93	0.99	1.00	0.86	0.47
leave-one-domain-out	18 cells, 50 tasks, 3 seeds	0.07	0.37	0.74	0.94	0.98	0.88	0.50

A replica of the original 16-cell validation (20 tasks, one run) has expected sign accuracy 0.74 under $b = -0.30$ and 0.60 under the null, and reaches 15 of 16 in 4% of true-effect worlds (1% under the null). H2a (WW/failures on the index): power 1.00 for the anchor effect, false-positive rate 0.03.

Realised precision on the 12 cells of the primary fit: slope standard error 0.46 (95% CI half-width 1.09), because the weighted residual variance is 18.6 times its sampling expectation. The simulation drew cell gains on the line with task-level noise only, so it did not model the model-by-domain effects the grid shows (residual standard deviation 18 points).

K Proof of Lemma 1

Write $y_i = \Delta_i = g + \beta p_i + u_i - e_i$. For a single regressor z_i, the least-squares slope converges to $\mathrm{Cov}(y, z)/\mathrm{Var}(z)$. For the naive regressor $z_i = P_{\mathrm{SAS}}(i) = p_i + e_i$, independence gives $\mathrm{Cov}(y, z) = \beta\sigma_p^2 - \sigma_e^2$ and $\mathrm{Var}(z) = \sigma_p^2 + \sigma_e^2$, so the slope converges to

$$\frac{\beta\sigma_p^2 - \sigma_e^2}{\sigma_p^2 + \sigma_e^2} = \lambda\beta - (1 - \lambda).$$

For $z_i = x_i = p_i + e_i'$, the error e_i' is independent of (p_i, e_i, u_i), so $\mathrm{Cov}(y, z) = \beta\sigma_p^2$ and $\mathrm{Var}(z) = \sigma_p^2 + \sigma_{e'}^2$, and the slope converges to $\lambda'\beta$. With fixed effects, the Frisch–Waugh–Lovell theorem reduces the fit to a single-regressor fit on residuals after partialling out the family and domain indicators, and the argument applies to the residualised variables, with σ_p^2, σ_e^2 and $\sigma_{e'}^2$ read as within-group variances; weights rescale these variances without changing the signs. If the two systems' errors are correlated, as when both run the same tasks, with $\mathrm{Cov}(u_i, e_i) = \sigma_{ue}$, the naive limit becomes $\lambda\beta - (\sigma_e^2 - \sigma_{ue})/(\sigma_p^2 + \sigma_e^2)$, still shifted down whenever $\sigma_{ue} < \sigma_e^2$, while the out-of-domain limit is unchanged, because e_i' comes from episodes on other domains. □

L Extension: the strong SAS

S2, the strong SAS: one agent, one model, one context; every call sees the conversation as the SAS does under the base prompt plus one role block. Each turn is two calls. *Plan* (tools shown with `tool_choice` "none") returns a JSON update of a task ledger and a progress ledger, a policy check (the applicable policy lines quoted and what they require) and a draft of the next action; *act* (native tools) reviews the draft against the policy check and the conversation, revises it if needed, and acts. Budget identical to the MAS cell of the same model and domain ($4B_{1\times}$, $c = B_{1\times}$, w, $a_{\min}$, the same status line and notice); after the wrap-up notice a turn is one call without tools and without a plan, as in the SAS; an unparsable plan is retried once; an act reply that is JSON text is never sent (a message drafted inside it is used, else the act call is repeated once with a reminder). Comparators: the SAS is the calibration run, the MAS the $4\times$ grid cell, both reused. Outcomes per cell: MAS − S2 and S2 − SAS with task-cluster CIs, completion tokens and the S2/SAS ratio, exhaustion and utilisation, review outcomes (kept, revised, replaced), and the mechanical bars. Reading: (1) S2 − SAS with a CI below zero on a cell means plan-and-review scaffolding inside one context also loses at the same ceiling, so the grid loss there is not specific to dividing the work among agents; (2) S2 − SAS covering zero or above it while MAS − S2 lies below zero attributes the loss to the division; (3) any other pattern is reported as mixed. The first smoke (5 tasks × 1 trial per cell) passed every mechanical bar; three problems were fixed before the full run, after a second smoke round that also passed every bar (JSON act replies reaching a closing call; plans whose draft was a narrative of the plan where the next action belonged, fixed by one sentence in the block; a logging artefact), with reward read last and playing no part. Cost stop $75. M2 (Figure 1b), in which the policy advisor becomes a checker that can veto an order or a customer message, is outlined but not evaluated here.

Result. Telecom × `gemini-3.1-flash-lite` (150 episodes, no infrastructure errors, $14.04): S2 36.0% $[28.8, 43.9]$ against 34.0% for the SAS and 12.0% for the MAS at $4\times$; per task against the SAS, better on 15, equal on 23 and worse on 12. S2 exhausted 39 of 150 episodes (the MAS 67); its finished episodes passed 54 of 111 and its exhausted ones none. Its median completion tokens per episode were 5.4 times the SAS's (the MAS 5.8). The mechanical bars pass: 33 of 3,664 calls were cut at c, all act calls followed by a closing call, and no JSON or cut text reached the customer. Of 39 act replies that came back as JSON, 22 carried a usable message, 38 were repeated once, and 6 repeats that were JSON again became the holding line (6 of 855 customer messages). Over 1,807 turns the review kept the draft 779 times, revised arguments 135 times, changed the tool 419 times (the model drafts tool names that do not exist), reworded the message 306 times and changed the

action type 62 times. Reading (2) holds. Read with the uncapped arm for the same cell, where the MAS ties the SAS, what the division costs here is budget spent on coordination: under the same ceiling and a similar spend ratio, the single context finishes more episodes and passes more of those it finishes. S2 − SAS carries the asymmetry of an uncapped SAS comparator.

M COMPUTE AND COST

Episodes reported, 7,050 in all: the 16 SAS and 16 MAS grid cells (4,800); three uncapped MAS cells (450); two MAS cells at $r = 25$ (300); R1, a capped SAS and a $1\times$ MAS on two cells (600); R2, SAS and MAS on two airline cells with the second simulator (600); S2 on one cell (150); and a duplicate retail × `qwen3.5-9b` MAS run (150). Pilots and smoke runs are not counted. Commercial API spend recorded by the runner (agent plus simulator at list prices): six Gemini MAS cells about $88 and their SAS cells about $43; six GPT-5 MAS cells about $112 and their SAS cells about $50; the $r = 25$ check $39; the uncapped arm $44.13; the two hosted 9b MAS cells $8.48; the simulator for the local 4b cells $9.25; R1 $17.23; R2 $26.47; S2 $14.04 and its smoke $1.59; about $453 in all. The judge pass and the collaborator's runs are not included. Table 12 gives the recorded cost per cell.

Table 12: Recorded cost per cell in USD at list prices (agent plus simulator; tau2's own cost field). Ratio: MAS over SAS; agent only: the same ratio without the simulator. The local 4-billion model has no per-call price, so its cells show the simulator cost only.

Model	Domain	SAS	MAS at 4×	Ratio	Agent only
`gpt-5-nano`	airline	4.66	9.15	2.0	2.9
	retail	3.74	7.49	2.0	3.1
	telecom	8.71	12.57	1.4	2.5
`gpt-5-mini`	airline	8.77	27.54	3.1	3.9
	retail	6.82	20.83	3.1	4.0
	telecom	16.94	34.45	2.0	3.4
`gemini-3.1-flash-lite`	airline	4.53	8.10	1.8	2.3
	retail	4.21	7.37	1.8	2.3
	telecom	5.53	9.53	1.7	2.6
`gemini-3-flash`	airline	8.52	19.43	2.3	2.6
	retail	6.36	14.01	2.2	2.7
	telecom	14.29	29.47	2.1	2.9
`qwen3.5-4b` (simulator only)	telecom	5.09	4.16	–	–
`qwen3.5-9b`	retail	–	3.28	–	–
	telecom	–	5.20	–	–

The 4-billion model ran on one laptop (Apple M4 Pro, 48 GB): 150 SAS episodes at about 2.3 minutes each and 150 MAS episodes in about 30 hours at one request at a time.

N FURTHER LIMITATIONS

(1) The airline and retail `qwen3.5-4b` cells were not run for compute reasons; a collaborator's separate runs of that model on those domains, on a different serving stack, scored 39.7 and 36.3% with 45 to 48% of episodes ending in agent errors that our stack did not show on telecom, so whether those cells would sit below 45% on our stack is unknown. (2) The OpenAI snapshots are scheduled for retirement on 11 December 2026; after that date the released trajectories are the record. (3) The 56% plan-parse failure rate of `qwen3.5-4b` is a property of that model under our parser; a parser that completes a missing closing brace is an obvious post hoc variant and was not run. (4) A write followed by a compensating revert leaves the database unchanged and is invisible to the primary measure; three such candidates in the pilot run are listed for hand adjudication. (5) The reasoning-only sensitivity meter cannot split 1.2% of gate-run output tokens, whose calls carry no details block; the primary meter still counts them. (6) The judge model is pinned; the secondary measure depends on it. (7) The uncapped arm uses the three cells at or below 45% and the extension one of them, each with three seeds. In the uncapped arm, excluding the 6 telecom × `gpt-5-nano` episodes in which plan text reached the customer moves that cell's gain from -10.0 to -12.0. (8) The `qwen3.5-9b` SAS values were computed by a collaborator; the per-episode files needed for the paired intervals of two of its cells are not held by us. (9) The open-weight rungs run on non-standard

stacks: a local 4-bit server, and an unpinned hosted route that did not enforce the cap on reasoning tokens.

O POSITION AGAINST PRIOR WORK

Table 13 compares this study with the closest prior work on the properties that matter for testing an architecture-selection rule in a new setting.

Table 13: Position against the closest prior work. No prior study has all six properties at once. "Rule tested" means an existing selection rule is applied unchanged to a new setting.

Study	Architectures	Capability varied	Compute matched (what, level)	Dual-control	Sim.	Rule
Kim et al. (2026)	SAS + 4 generic MAS	3 families × 3 tiers	reasoning tokens, per system, realised mean	no	n/a	fitted
Tran and Kiela (2026)	SAS + 5 MAS variants	4 models	thinking tokens, per call, split	no	n/a	no
Jwalapuram et al. (2026)	CoT-SC vs automated MAS	several	USD, post hoc	no	n/a	no
Yu et al. (2026)	SAS vs MAS	3 scales	relay bandwidth (theory)	no	n/a	no
Gao et al. (2025)	SAS vs MAS	several	cost reported	no	n/a	no
LangChain (2025)	single, swarm, supervisor	1 model	no	yes	no	no
Kang et al. (2026)	agent vs agent + verifier	3 models	no	yes	no	no
This paper	SAS vs MAS (centralized)	3 families × 2 tiers	completion tokens, per episode, shared ledger	yes (3 domains)	fixed; second on 2 cells	yes