Apollo-1 Benchmark Results

Unparalleled reliability for workflow execution at a fraction of the cost

τ²-bench is a public benchmark for conversational agents from Sierra. Its airline domain tests whether an agent can handle airline service requests using tools while following the airline's policies, including booking changes and cancellations. Each of its 50 tasks is run four times against a simulated customer.

Benchmark τ²-bench v1.0.1, airline Scored 200 conversations, first attempt only, no retries

Run 28 October 2026

TASK COMPLETION

90.5%

of 200 conversations (181) completed correctly on the first attempt

Next best shown: **Claude Opus 5.5* at 86.5%**

CONSISTENCY

88.0%

of tasks completed correctly in all four trials

Next best shown: **Claude Opus 5.5* at 78.0%**

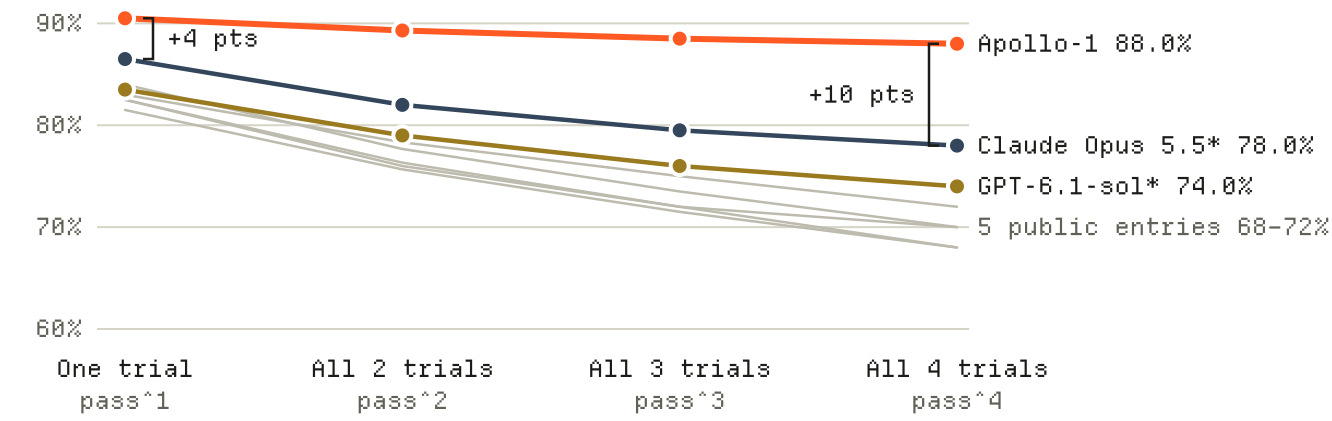
COST

$0.013

agent cost per conversation

5.4× less than GPT-6.1-sol*, **30× less** than Claude Opus 5.5*

The gap widens with every repeat



Apollo-1 loses 2.5 points between one trial and four. With the same simulated customer, Claude Opus 5.5 and GPT-6.1-sol lose 8.5 and 9.5; the leading public entries lose 11 to 15. Axis starts at 60%. * Run by AUI on τ²-bench's official agent with the same simulated customer as Apollo-1. The grey public entries used a different simulated customer; see How we tested.

Getting the work done, and knowing when not to

The benchmark mixes two kinds of request. In 27 of the 50 tasks, the right outcome changes the booking: the agent must book, cancel or modify a reservation. In the other 23, the right outcome is to refuse, inform or decline, and change nothing.

89 / 108 82%

Booking changes completed

Conversations won where the right outcome was to book, cancel or modify a reservation.

92 / 92 100%

Refusals and answers handled

Conversations won where the right outcome was to refuse, inform or decline, changing nothing.

48 / 50 96%

Same result every trial

Tasks with the same result, pass or fail, in all four trials: 44 passed every time and 4 failed every time.

Each of Apollo-1's 170 database writes came after an explicit confirmation from the customer. All 19 losses were in tasks that change the booking.

How it compares

SYSTEM	ONE TRIAL PASS^1	ALL FOUR TRIALS PASS^4	POINTS LOST 1 → 4	COST PER CONVERSATION AND MULTIPLE OF APOLLO-1
Apollo-1 (AUI)	90.5%	88.0%	2.5	$0.0130
Claude Opus 5.5 (xhigh)*	86.5%	78.0%	8.5	$0.390 30×
GPT-6.1-sol (max)*	83.5%	74.0%	9.5	$0.0706 5.4×
GPT-5.2 (high)	83.0%	72.0%	11.0	$0.114 8.8×
Claude Opus 4.5	84.0%	70.0%	14.0	$0.399 31×
GLM-5	82.5%	70.0%	12.5	Not published
Gemini 3 Flash	82.5%	68.0%	14.5	Not published
Qwen3.5-397B-A17B	81.5%	68.0%	13.5	Not published
Gemini 3 Pro	80.5%	66.0%	14.5	Not published
Claude Sonnet 4.5	72.0%	48.0%	24.0	$0.296 23×
GPT-5.2 (no reasoning)	52.5%	22.0%	30.5	$0.0537 4.1×

Apollo-1 completed all four trials of a task more often than any other system in the table completed one. Sorted by pass^4. * Run by AUI on τ²-bench's official agent with the same simulated customer as Apollo-1 (gpt-6-luna). Other rows are from the public leaderboard's submission files, read 20 October 2026, which use GPT-5.2 as the simulated customer. Costs exclude the simulated customer; public entries' costs are as their submitters published them. Multiples compare each cost with Apollo-1's.

Why the results hold steady

An LLM agent receives the policy as a system prompt and is trusted to follow it. Apollo-1 instead receives the policy as a program. Language models read what the customer says and word the reply; a symbolic core decides what the policy allows, the same way every time. A booking changes only after the customer confirms the exact change.

The same facts lead to the same decision, so results barely move from trial to trial. The design also keeps cost down: the logic that decides runs on the CPU and is cheap, and the language work goes to a low-cost model. The reliability comes from the architecture, not a larger model.

Where it fell short

We read all 19 failed conversations. They come from 6 of the 50 tasks, grouped here by the cause the report assigns each task.



6 **Agent behaviour or error**, such as booking a new trip without cancelling the old one, or giving a wrong answer about a change fee.

8 **Policy interpretation**: the benchmark's answer key reads a rule differently from Apollo-1.

5 **Sequencing or the simulated customer**, which declined a two-step change after seeing the first step's price, or ended a call as it gave the go-ahead.

Each loss is explained in the technical report.

How we tested

BENCHMARK AND SCORING

τ²-bench v1.0.1, airline domain: 50 tasks, four trials each, 200 conversations. Only the first attempt is scored; nothing was retried or rerun, and all 200 completed with no turn or run errors. The tasks, policy, tools, database, simulated-customer prompt and evaluator are the benchmark's own.

WHAT IS CUSTOM

The agent is Apollo-1 running an airline program written only from the benchmark's policy and tool definitions, with no task-specific rules. The public airline tasks were used in development to find failures and, with other domains, to calibrate some Type-1 thresholds; a program change was accepted only if it traced to a policy sentence. AUI trained or fine-tuned no model on τ-bench tasks, rewards or evaluation data for this submission. It is a custom submission under the leaderboard's rules.

COMPARISONS

Claude Opus 5.5 (effort xhigh) and GPT-6.1-sol (reasoning max), marked *, were run by AUI with τ²-bench's official agent, prompts, tools, tasks and evaluator, and with gpt-6-luna as the simulated customer, as in Apollo-1's run. Other results are the public leaderboard's published submissions, which use GPT-5.2 as the simulated customer.

COST AND EVIDENCE

For Apollo-1 and the two models AUI ran, cost counts every agent-side model call at the providers' list prices. Other costs are as their submitters published them. The simulated customer's cost is excluded throughout. All 200 trajectories are in the submission package, and every loss is explained in the report below.

Let's prove it on your work

Choose the task, agree the success criteria, and run a scoped evaluation on your own policies, systems and cases.

contact@aui.io

Apollo-1 on τ²-bench Airline

Apollo-1 is AUI's neuro-symbolic agent runtime. This report describes what was evaluated, how the system works at a high level, and what the 200 scored airline conversations show: accuracy, consistency across trials, cost, latency and every loss.

System Apollo-1 (AUI) Domain airline · 50 tasks × 4 trials

Scoring first attempt only, no retries Run 2026-10-28 10:43 EDT

Contact contact@aui.io

What makes this submission custom

The agent under test is not one LLM behind the default τ-bench scaffold. It is the Apollo-1 runtime running an airline agent program, connected through τ²-bench's agent interface. Under the leaderboard's rules that makes the submission **custom**, and it is filed as such.

Everything on the benchmark side is unchanged:

- the tasks, the airline policy and the tool set;
- the tools' results and the database;
- the user-simulator prompt and orchestration;
- the evaluator.

The user simulator is `openai/gpt-6-luna` at temperature 1.0 (the model does not accept temperature 0), with its API-default reasoning effort.

Architecture

Apollo-1 is a neuro-symbolic system. Neural models carry the language and a symbolic core carries the decisions, inside one computation per turn. Every fact the agent uses declares where it comes from:

- the customer's own words, read by a neural model into the program's vocabulary;
- a tool or system of record, read symbolically;
- a calculation over other facts.

Each turn runs the same six stages.

Read	Establish	Check	Act	Reply	Seal
The customer's message is read into the program's typed vocabulary. NEURAL	Each needed fact is found in a fixed order: known, computed, read from a tool, or asked. SYMBOLIC	The policies guarding the requested action are evaluated against established facts. SYMBOLIC	A state change runs only after the customer confirms the exact change shown. SYMBOLIC	The decided outcome is phrased for the customer. NEURAL	Facts, rules, values and calls are recorded in the turn's trace. SYMBOLIC

Neural elements, and why they are interchangeable

The neural side is asked two kinds of question: what did the customer say, in the program's terms, and how should a decided outcome be worded? It is never asked whether a rule holds or what to do next. Because its job is that narrow, Apollo-1 is agnostic to the model behind it. Swapping the model changes how replies are phrased, not what the policy allows.

Inside the reasoning, Apollo-1 also uses **Type-1 models**: small, fast models that each answer one narrow, typed question and return a typed value the symbolic core then checks. Examples are which of the options shown the customer chose, or which event a stated date refers to. They never decide an action on their own.

This run used two neural models:

- `openai/gpt-6-luna` at high reasoning effort, for reading and phrasing;
- `typesafe/jev` as a Type-1 reader.

Where the time goes

Across 1,005 agent turns, 89.3% of reply time was spent in the general neural model and 6.4% in Type-1 calls. The symbolic core and tool I/O took the remaining 4.3% (Figure 6). The logic that decides is cheap; the cost and latency of the system are dominated by language.

From policy to program

An LLM agent receives the policy as a system prompt and is trusted to follow it. Apollo-1 instead receives a **program**: the policy translated into Apollo's abstraction layer, a folder of typed YAML declarations. The public documentation at docs.aui.io describes them.

- **program**: identity, voice and the clock that defines "today".
- **vocabulary**: the facts and records the agent can know and hold.
- **sources**: where facts come from and what changes the world. Here these are the benchmark's own tools.
- **derivations**: facts computed from other facts, such as fees, allowances, refund amounts and eligibility.
- **capabilities**: the jobs the agent can do. There are 14 for airline, including book, cancel, change flights, change cabin, add bags, edit passengers, compensation and transfer.
- **policies**: the rules that must hold before an action proceeds, each with its refusal wording.

The program is built only from the benchmark's airline policy and tool schemas. Every rule cites the policy section it encodes (`encodes`), and the policy text itself is carried as the program's reference corpus. Below are four policy sentences and the declarations they became, excerpted from the program used in this run. Only internal development comments are removed.

“Basic economy flights cannot be modified.”

POLICY

A rule guarding the change-flights capability. It is evaluated on the reservation's cabin as read from the booking tool, and refuses in its own words when it fails.

```
# policies/global.yaml
- id: basic-economy-flights-cannot-change
  intent: Basic economy flights cannot be modified (the cabin can still be
changed).
  encodes: '§3.2'
  protects: action
  severity: critical
  applies_to: [change-flights]
  when: reservation.cabin is known
  requires: reservation.cabin != basic_economy
  otherwise:
    refuse: >
      Reservation {reservation.id} is in basic economy, and basic economy flights
cannot be
      changed. The cabin can be changed, though, keeping the same flights.
```

“The agent can only cancel if the booking was made within the last 24 hours, the flight was cancelled by the airline, it is a business flight, or the user has travel insurance and the reason is covered.”

DERIVATION + POLICY

Eligibility is computed from facts read from the tools and the customer's stated reason. The policy then requires it before the cancel capability may run.

```
# derivations.yaml
- id: cancel.reason_covered
  encodes: '§2.14'
  means: Whether the reason for cancelling is one travel insurance covers (health
or weather).
  value: cancel.reason in [health, weather]
- id: reservation.cancellable
  encodes: '§4.4'
  means: >-
    Whether the airline's rules let this reservation be cancelled: nothing on it
has flown, and it was booked
    within the last 24 hours, or the airline cancelled a flight on it, or it is in
business, or it is insured
    and the reason is one the insurance covers.
  value: >-
    reservation.any_flown == false and
    (reservation.booked_within_24h or reservation.has_cancelled_flight or
reservation.cabin == business
     or (reservation.insured and cancel.reason_covered))

# policies/global.yaml
- id: cancellation-rules
  intent: >
    A reservation can be cancelled only if it was booked within the last 24 hours,
the airline
    cancelled a flight on it, it is in business, or it is insured and the reason
is health or weather.
  encodes: '§4.4'
  protects: action
  severity: critical
  applies_to: [cancel-trip]
  when: reservation.any_flown == false and (cancel.reason is known or
reservation.cancellable is known)
  requires: reservation.cancellable == true
  otherwise:
    refuse: >
      Reservation {reservation.id} cannot be cancelled under the airline's rules:
it was not booked
      within the last 24 hours, no flight on it was cancelled by the airline, it
is not in business,
      and the reason is not one its insurance covers.
```

“Each reservation can use at most one travel certificate, at most one credit card, and at most three gift cards.”

POLICY OVER A TYPED PLAN

Checked on the payment plan of the exact booking before the customer is asked to confirm it. When it fails, the agent asks which payment method to use.

```yaml
# policies/global.yaml
- id: a-booking-pays-with-at-most-one-certificate-one-card-three-gift-cards
  intent: A booking can use at most one travel certificate, at most one credit
card and at most three gift cards.
  encodes: '§2.7'
  severity: critical
  applies_to: [book-trip]
  when: pay.lines is known
  requires: pay.certificates <= 1 and pay.credit_cards <= 1 and pay.gift_cards <=
3
  otherwise:
    refuse: A booking can use at most one travel certificate, one credit card and
three gift cards.
    ask: pay_line.payment_id
```

“Before taking any actions that update the booking database, you must list the action details and obtain explicit user confirmation (yes) to proceed.”

CAPABILITY + RUNTIME LAW

Every capability that changes the world declares what it reads back. The runtime runs its `changes` only after a confirmed yes to exactly what was shown.

```
# capabilities/cancel-trip/capability.yaml
id: cancel-trip
intent: Cancel a whole reservation, when the airline's rules allow it; the refund
goes to the original payment methods.
use_when: >
  The customer wants a reservation cancelled -- one, several, or all of their
upcoming ones.
offer: cancel a reservation
scope: per reservation
batch: true
needs: [user.id, reservation.id, cancel.reason, reservation.cancellable]
optional: [reservation.paid_total]
reads: [user_details, reservation_details, segment_now]
changes: [cancel]
confirm:
  show:
    - "Reservation {reservation.id}: {reservation.origin} to
{reservation.destination}, {reservation.flight_type}, {reservation.cabin},
{reservation.pax_said}"
    - "Cancelled in full; the refund goes to the original payment methods in 5 to
7 business days"
  assert_before: Nothing has been cancelled yet.
```

Section numbers in `encodes` refer to the airline policy's sections in the order they appear. Facts such as `reservation.cabin` or `reservation.booked_within_24h` are declared across the program's files. The vocabulary is one example; derivations, sources and capabilities declare others. Each declares its origin: a tool's result, a calculation, or the customer's words.

The variables these declarations use (`reservation.cabin`, `cancel.reason`, `pay.gift_cards` and the rest) are declared by this agent's own program. They are not part of Apollo-1's language. Apollo-1 relies on each variable's role, as its declaration states it: its type, where its value comes from, and which rules and actions read or change it. It never relies on the variable's name. The same runtime runs agents in other domains, with entirely different vocabularies.

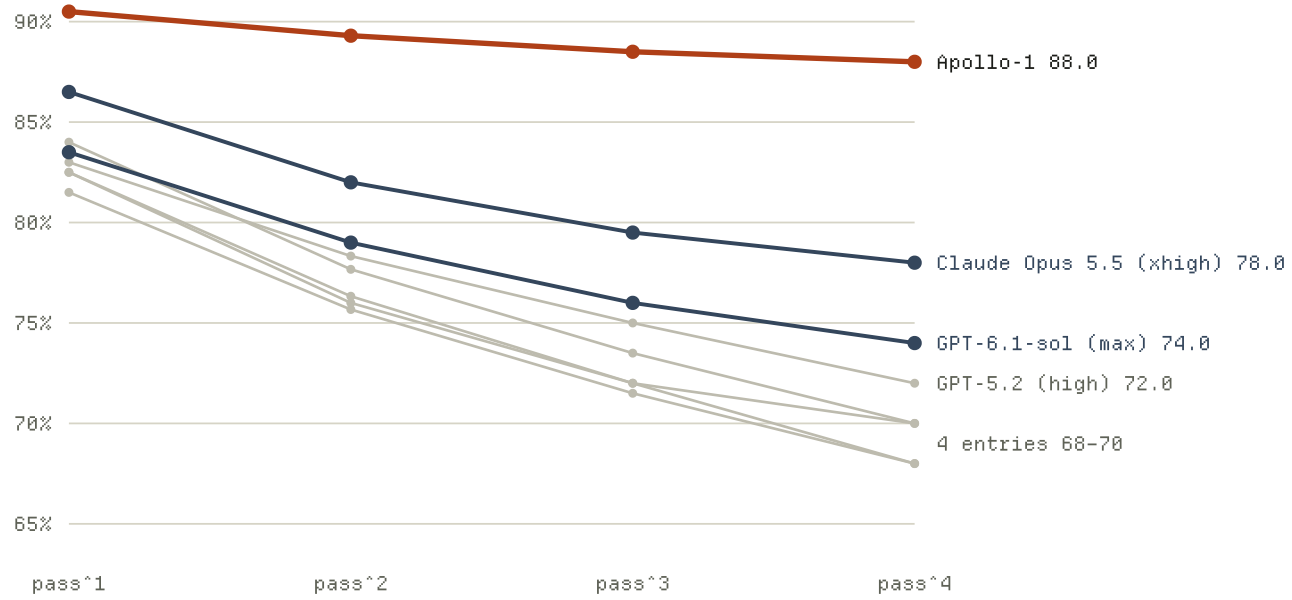
> The method in one line: **codify the policy, translate each customer request into the same symbolic language, compute the answer with logic on the CPU, and use neural models only to translate in and out of natural language.** The model's job is language, and the program's job is the rules.

Consistency

pass^k measures whether an agent succeeds on every one of k independent tries at the same task. Systems that decide by sampling a model lose ground as k grows: a task solved three times in four still counts as a failure at pass^4. When the decision is computed,

the same facts lead to the same decision, and variance is left mostly to the conversation itself: the simulated customer's wording, choices and mistakes.

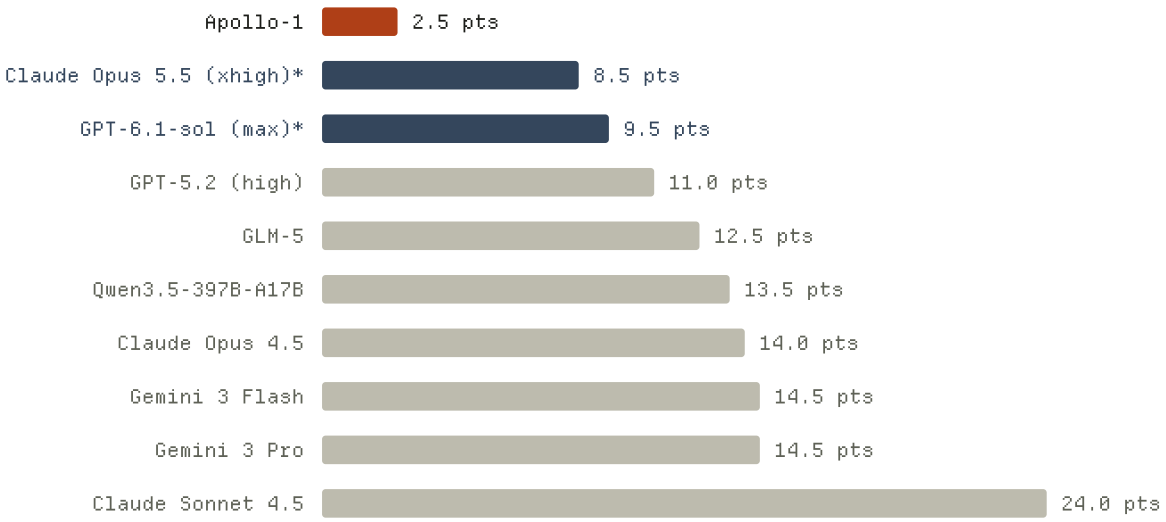
FIGURE 1 · PASS^K, AIRLINE



Apollo-1 run by AUI, same simulator as Apollo-1: Claude Opus 5.5 (xhigh), GPT-6.1-sol (max)
public entries, GPT-5.2 simulator: GPT-5.2 (high), Claude Opus 4.5, GLM-5, Gemini 3 Flash, Qwen3.5-397B-A17B

Apollo-1 is the flattest line. Pink: the two newest frontier models, run by AUI on τ²-bench's official agent with the same gpt-6-luna user simulator as Apollo-1. Grey: the leading airline entries on the public τ²-bench 1.x board, from its published submission files (read 2026-10-20), with GPT-5.2 as the simulator.

FIGURE 2 · POINTS LOST FROM PASS^1 TO PASS^4



* Run by AUI with the same user simulator as Apollo-1 (note).

A 2.5-point fall against 8.5 to 24 points. The fall from pass^1 to pass^4 is a direct measure of run-to-run inconsistency on the same tasks.

How it compares

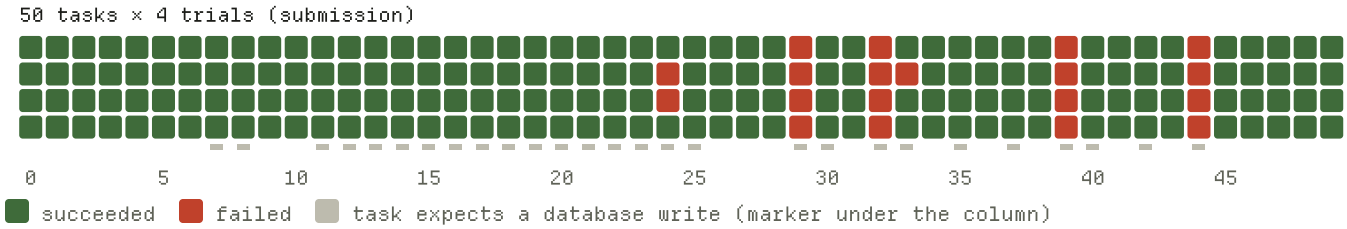
SYSTEM	PASS^1	PASS^2	PASS^3	PASS^4	FALL	COST / CONVERSATION	COST / SUCCESS
Apollo-1 (AUI)	90.5	89.3	88.5	88.0	2.5	$0.0130	$0.0143
Claude Opus 5.5 (xhigh)*	86.5	82.0	79.5	78.0	8.5	$0.390	$0.451
GPT-6.1-sol (max)*	83.5	79.0	76.0	74.0	9.5	$0.0706	$0.0845
GPT-5.2 (high)	83.0	78.3	75.0	72.0	11.0	$0.114	$0.137
Claude Opus 4.5	84.0	77.7	73.5	70.0	14.0	$0.399	$0.475
GLM-5	82.5	76.0	72.0	70.0	12.5	n/p	n/p
Gemini 3 Flash	82.5	76.3	72.0	68.0	14.5	n/p	n/p
Qwen3.5-397B-A17B	81.5	75.7	71.5	68.0	13.5	n/p	n/p
Gemini 3 Pro	80.5	74.7	70.0	66.0	14.5	n/p	n/p
Claude Sonnet 4.5	72.0	62.0	54.5	48.0	24.0	$0.296	$0.412

SYSTEM	PASS^1	PASS^2	PASS^3	PASS^4	FALL	COST / CONVERSATION	COST / SUCCESS
GPT-5.2 (no reasoning)	52.5	35.7	27.0	22.0	30.5	$0.0537	$0.102

- **Apollo-1's pass^4 is higher than every other system's pass^1.** It succeeded on all four tries of a task (88.0%) more often than any general-purpose model here succeeded on one.
- **Far fewer failures at k=4.** Apollo-1 failed 12% of tasks at pass^4, against 22% for Claude Opus 5.5 (xhigh) and 26% for GPT-6.1-sol (max) on the same simulator, and 28% for GPT-5.2 (high) and 30% for Claude Opus 4.5 on the public board.
- **A fraction of the cost.** At $0.0130 per conversation, Apollo-1 cost 5.4× less than GPT-6.1-sol (max), 8.8× less than GPT-5.2 (high), 30× less than Claude Opus 5.5 (xhigh) and 31× less than Claude Opus 4.5. Per successful conversation it was $0.014, against $0.085, $0.137, $0.451 and $0.475. On OpenRouter's own airline runs, Claude Fable 5 costs $0.92 per task, about 71× more (Figure 4).
- **More reasoning does not buy consistency.** At their highest reasoning settings, Claude Opus 5.5 and GPT-6.1-sol still lost 8.5 and 9.5 points from pass^1 to pass^4, against Apollo-1's 2.5. See The thinking paradox of larger frontier models.
- **A low-cost model inside.** The neural model in this run, gpt-6-luna, is priced at $0.10 / $0.50 per million input / output tokens. The reliability comes from the architecture around it, not from a larger model.

* Run by AUI with τ²-bench v1.0.1's official agent, prompts, tools, tasks and evaluator: 50 tasks × 4 trials, seed 300, no conversation rerun, with `gpt-6-luna` as the user simulator at its API defaults, as in Apollo-1's run. Claude Opus 5.5 ran at effort xhigh and GPT-6.1-sol at reasoning max; only the model client library was updated. Cost is every agent-side call at list prices ($2 / $10 per million input / output tokens for GPT-6.1-sol, $4 / $20 for Claude Opus 5.5). Other rows are the leaderboard's published 1.x airline submissions (read 2026-10-20), which use GPT-5.2 at low reasoning as the simulator; costs as published ("n/p": none published). Cost per success is cost per conversation divided by pass^1.

FIGURE 3 · EVERY SCORED CONVERSATION, BY TASK AND TRIAL



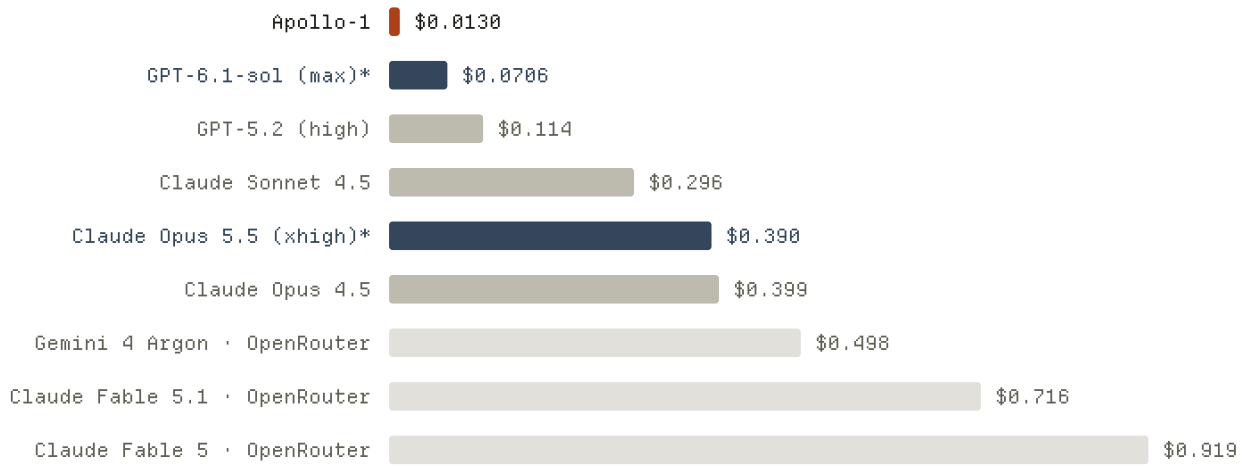
44 tasks pass in all four trials and 4 fail in all four. Only tasks 24 (2 of 4) and 33 (3 of 4) are mixed. All 23 tasks whose correct outcome is a refusal or information without a write pass every trial.

Results in detail

METRIC	VALUE	NOTE
pass^1 / pass^2 / pass^3 / pass^4	90.5 / 89.3 / 88.5 / 88.0	computed by `tau2 submit prepare`
Conversations succeeded	181 / 200	first attempt of every trial is scored
Trials won on tasks whose correct outcome changes the booking	89 / 108	27 tasks: book, cancel or modify

METRIC	VALUE	NOTE
Trials won on tasks whose correct outcome changes nothing	92 / 92	23 tasks: refuse, inform or decline
Database state matches expected	181 / 200	tau2's DB check
Tool calls per conversation	19.6	0 tool errors in 3,916 calls
Writes per conversation	0.85	170 writes, each after an explicit confirmation
Customer turns per conversation	6.0	mean
Agent cost per conversation	$0.0130	median $0.0105; p90 $0.0242
Agent reply time	18.3 s median	mean 24.7 s; p90 52.5 s; 1,005 turns
Conversation duration	114 s median	p90 267 s, with the simulator's time included
Turn errors / run errors / max-step endings	0 / 0 / 0	all 200 conversations completed and were scored; none rerun
Calls that failed or timed out	2 / 9,192	both Type-1 timeouts, both in conversations that succeeded; no lost conversation had a failed call

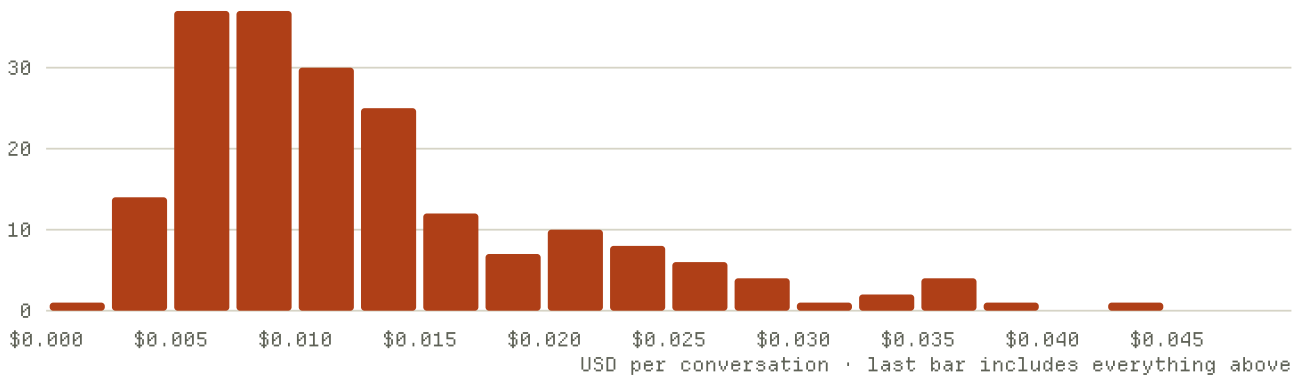
FIGURE 4 · AGENT COST PER AIRLINE CONVERSATION, USD



* Run by AUI with the same user simulator as Apollo-1 (note).

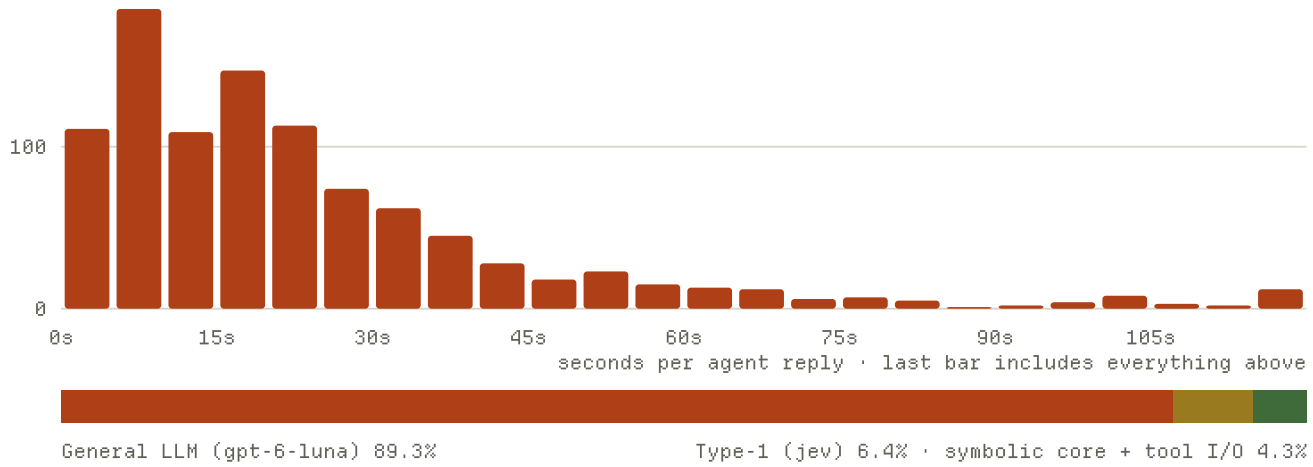
The lowest cost per conversation, at the highest accuracy. Apollo-1's cost covers every model call the agent made at the providers' per-token prices; the two pink entries, run by AUI, use the same basis. The simulator's cost is excluded throughout. Grey entries are as published by their leaderboard submitters; faded entries are OpenRouter's average cost per task on its own airline runs (read 2026-10-20).

FIGURE 5 · DISTRIBUTION OF AGENT COST PER CONVERSATION (THIS RUN)



Four in five conversations cost under two cents. The tail is long multi-change requests: rebooking several reservations, or searches over many flight options.

FIGURE 6 · AGENT REPLY TIME, AND WHERE IT GOES



Reply time is dominated by the general language model. The share of each component is measured as wall time inside each turn's model and Type-1 calls; the rest is the symbolic core and tool I/O.

Every loss

All 19 failed conversations were read. They come from six tasks, all of them tasks whose correct outcome changes the booking. For each task, the table gives what happened and the cause, using the report's labels.

TASK	LOST	WHAT HAPPENED	WHY IT HAPPENED
29	4 / 4	The customer wants a destination change, which the policy requires as cancel plus rebook. In three trials the agent booked the new round trip but did not cancel the old reservation, and in two of those it paid with a gift card where the task expects the original card. In the fourth, the simulated customer ended the call as out of scope.	AGENT BEHAVIOUR The policy requires a destination change to be done as a cancellation plus a new booking, and the program states that sequence (see the note below the table). In three trials the agent carried out only the new booking and left the old reservation in place; in two of those it also paid with a gift card rather than the original card the task expects. The fourth loss came when the simulated customer ended the call as out of scope.
32	4 / 4	The task expects a cabin upgrade, then a flight change. The simulated customer applies its $100 budget to the first step alone ($396) and declines.	SEQUENCING / SIMULATOR The expected path has two steps: upgrade the cabin, then change the flights. At the first step, the simulated customer measured its $100 budget against the upgrade alone, which cost $396, and declined.

TASK	LOST	WHAT HAPPENED	WHY IT HAPPENED
39	4 / 4	The expected outcome cancels an insured economy reservation for a reason that, in our reading of the policy, insurance does not cover; the agent refuses.	POLICY INTERPRETATION Under the policy, insurance makes a reservation cancellable only when the reason is one the insurance covers. As we read §2.14 of the policy, the covered reasons are health and weather, and the program encodes that (see From policy to program). The customer's reason in this task is not one of them in our reading, so the agent refused in every trial, while the answer key expects the cancellation to go through.
44	4 / 4	The task measures a "3 hours" limit per flight; the agent and the simulated customer measure it per trip.	POLICY INTERPRETATION A "3 hours" limit can be measured per flight or across the whole trip. The answer key measures it per flight. Apollo-1 measured it across the trip, and the simulated customer did the same, so the two worked to a different limit from the one the task scores against, in all four trials.
24	2 / 4	Once, the simulated customer ended the call in the same message as its go-ahead, so nothing was booked. Once, the search across both New York airports and the West Coast offered EWR–LAX as the direct round trip and the customer accepted it; the cheapest direct round trip, which the task expects, is JFK–SEA.	SIMULATOR · 1 AGENT ERROR · 1 Two different causes, in a task the agent succeeded on in the other two trials. In one, the simulated customer gave its go-ahead and ended the call in the same message, so the booking was never made. In the other, the agent offered EWR–LAX as the direct round trip and the customer accepted it, but the task expects the cheapest direct round trip, which is JFK–SEA.
33	1 / 4	The insured customer asked for the change fee to be waived. The airline charges no change fee, only the fare difference, but the agent answered that insurance does not waive change fees; after repeated requests it transferred the call instead of making the changes.	AGENT ERROR The airline charges no change fee, only the fare difference, so there was no fee to waive. The agent instead answered that insurance does not waive change fees, and after repeated requests it transferred the call rather than making the changes. The same task succeeded in the other three trials.

Grouped by the labels above, the 19 losses are 6 from agent behaviour or error (tasks 29, 24 and 33), 8 from policy interpretation (tasks 39 and 44), and 5 from sequencing or the simulated customer (tasks 32 and 24).

The four tasks lost in every trial are all among the 27 airline tasks revised in the τ³ task fixes, which are included in τ²-bench v1.0.1. Our program states both policy sequences those fixes enforce: cancel and rebook for a route change, and upgrade before changing a basic-economy flight. The remaining gaps are listed above as they are.

The thinking paradox of larger frontier models

Newer, larger models do not reliably score higher here: OpenRouter's airline results show newer releases scoring below older ones[1]. And on the same simulator as Apollo-1, the two newest frontier models at their highest reasoning settings, Claude Opus 5.5 (xhigh) and GPT-6.1-sol (max), still lost 8.5 and 9.5 points from pass^1 to pass^4.

Our hypothesis: on a strict, policy-bound workflow, extra thinking becomes a liability. Published work documents three effects:

- **More thinking, more variance.** Across agent benchmarks including τ-bench airline, more reasoning effort gave equal or lower accuracy in 21 of 36 cases[2]. Longer thinking also makes outputs more variable[3], and pass^k penalizes variance directly[4].
- **Persistence wears models down.** Under repeated pressure, models give up correct positions[5,6]. On τ-bench airline, two models with similar scores differed about fourfold in how often a persistent customer talked them into a policy violation[7].
- **Actions drift.** Agents take actions that do not match their instructions or what their tools returned[8]. On airline, a single wrong step in a write cuts the odds of success by up to 92%[9].

Apollo-1 is built to avoid all three. Its rules are evaluated by symbolic logic rather than reasoned about, a customer's insistence does not change the facts a rule checks, and a write runs only after a confirmed yes to the exact change. The result is the flat line in Figure 1.

The simulated customer itself makes mistakes in a large share of airline conversations[10], which is why consistency over four trials is the measure we lead with.

How the agent was built

- **Program:** the airline program was written from the benchmark's airline policy and tool schemas. It contains no task identifiers, no expected actions and no rules for individual tasks. Every rule cites the policy sentence it encodes.
- **Development:** the public airline tasks were used during development to find and diagnose failures. A program change was accepted only if it traced to a policy sentence. The agent runs on the same general Apollo-1 model and runtime as every other Apollo-1 agent; only the program is specific to the airline.
- **Calibration:** some decision thresholds of the Type-1 readers were calibrated on hand-labelled answers. Those answers came from development conversations in several domains, including conversations on the public airline tasks.
- **Training:** no model was trained or fine-tuned by AUI on τ-bench tasks, rewards or evaluation data for this submission.
- **Runtime information:** the agent sees only what a standard agent sees: the conversation, the tools' results and the airline policy, as encoded in the program.

Configuration

BENCHMARK	τ²-bench v1.0.1, airline domain, base split (50 tasks)
TRIALS	4 per task, seed 300; every first attempt scored, no reruns or retries
AGENT	Apollo-1 runtime with an airline agent program, through τ²-bench's agent interface
NEURAL MODELS	`openai/gpt-6-luna` (reasoning effort high); `typesafe/jev` (Type-1 reader)
USER SIMULATOR	`openai/gpt-6-luna`, temperature 1.0, reasoning effort API default (not set; measured ≈ medium), τ²-bench's default user-simulator prompt
UNCHANGED	tasks, policy, tools, database, user-simulator prompt, orchestration, evaluator

COST METHOD	every agent-side model call from the run's own ledger, at $0.10 / $0.01 / $0.50 per million input / cached / output tokens for gpt-6-luna and $0.042 per million tokens for jev. CPU cost is negligible and not included: the whole run averaged about 7.1 CPU-seconds per conversation, benchmark harness and simulator client included.
TRAJECTORIES	all 200, in tau2's native format, in the submission package

References

1. OpenRouter (2026). τ²-Bench Airline benchmark results, read 2026-10-20. openrouter.ai/benchmarks/tau2-bench-airline
2. Kapoor, S. et al. (2025). Holistic Agent Leaderboard: The Missing Infrastructure for AI Agent Evaluation. arXiv:2510.11977. arxiv.org/abs/2510.11977
3. Ghosal, S. S. et al. (2025). Does Thinking More always Help? Mirage of Test-Time Scaling in Reasoning Models. NeurIPS 2025. arXiv:2506.04210. arxiv.org/abs/2506.04210
4. Yao, S. et al. (2024). τ-bench: A Benchmark for Tool-Agent-User Interaction in Real-World Domains. arXiv:2406.12045. arxiv.org/abs/2406.12045
5. Hong, J. et al. (2025). Measuring Sycophancy of Language Models in Multi-turn Dialogues. Findings of EMNLP 2025. arXiv:2505.23840. arxiv.org/abs/2505.23840
6. Tang, L. et al. (2026). SPINE: Measuring LLM Sycophancy under Sustained Multi-Turn Pressure. arXiv:2609.09090. arxiv.org/abs/2609.09090
7. Kirgis, P. et al. (2026). Log analysis is necessary for credible evaluation of AI agents. arXiv:2605.08545. arxiv.org/abs/2605.08545
8. Zhang, W. et al. (2025). MIRAGE-Bench: LLM Agent is Hallucinating and Where to Find Them. arXiv:2507.21017. arxiv.org/abs/2507.21017
9. Cuadron, A. et al. (2025). SABER: Small Actions, Big Errors — Safeguarding Mutating Steps in LLM Agents. arXiv:2512.07850. arxiv.org/abs/2512.07850
10. Barres, V. et al. (2025). τ²-Bench: Evaluating Conversational Agents in a Dual-Control Environment. arXiv:2506.07982. arxiv.org/abs/2506.07982