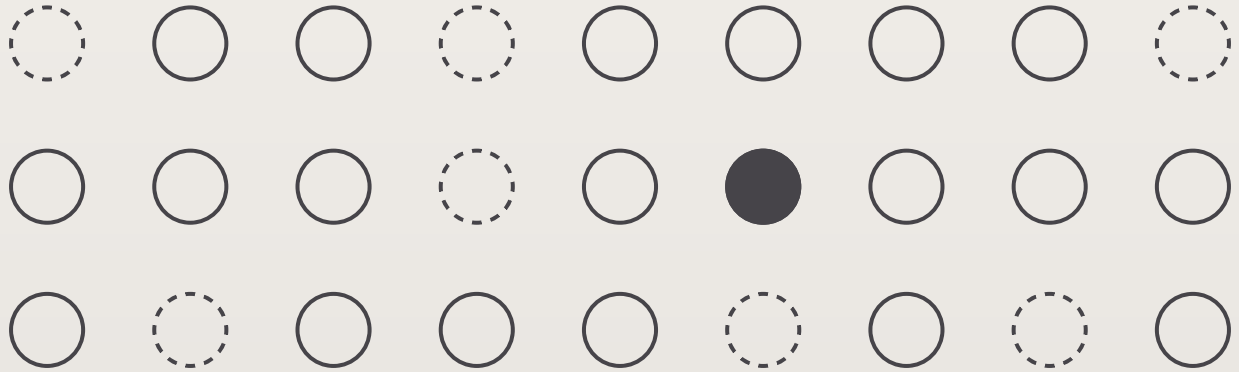




Executive summary

Serving Context to Probabilistic Agents

Correct Answers with 93% Fewer Tokens

27 AI models chose their own steps through two MCP gateways with access to the same systems and the same business context. The one that served that context on demand and ran the computation server-side got them to more correct answers, and reached the same correct answers with 93% fewer tokens and a tenth of the tool calls.

Results from 810 runs across 27 AI models, three business tasks, and two MCP gateways given the same business context.

What we tested

When an agent is given a question and chooses its own steps to answer it, rather than following steps built in advance, it is working probabilistically. Every step it takes costs tokens, and an agent working on business data spends them on two things: the context that tells it what the business means, and the work of fetching and combining the data. Both depend on the gateway the agent works through, and that is what we tested.

The agents worked on an account-health review across three business systems an enterprise already runs: a CRM, a cloud data warehouse, and an IT service-management system. The connections were live and the data was synthetic, built to behave like a real company's records, including the fact that the three systems don't identify accounts the same way.

Each agent worked through one of two gateways, and both gateways were given the same business context. **CData Connect AI Gateway** retrieves the relevant business context when the agent asks for it and runs one request across all three systems. The comparison was **an open-source MCP gateway** in front of each system's own MCP server, with the same business context loaded into every session.

That context came in three layers: system context (what's in each system and how the records connect), semantic definitions (how the business defines account health and declining usage), and company knowledge (how to request data from the systems). The definitions were word for word the same on both gateways; the rest was written for each gateway's tools, so each side got instructions it could actually use.

This made the test probabilistic: the models chose their own steps and wrote their own requests every time. Nothing was built into Connect AI Gateway in advance: no precomputed views, virtual schemas, or custom tools.

The agents were given three tasks.

Task	What the agent had to do	Why it's hard
Rank at-risk accounts	Find the High-priority accounts in poor health, worst first	Combines contract, usage, and support data from three systems with business rules
Find declining at-risk accounts	Find accounts that are both in poor health and declining in usage	Two independent signals have to agree
Queue accounts for review	Add exactly the right 21 accounts to a review queue	A wrong write lands in a live system

We ran 27 AI models from 12 developers, from economy models to frontier models, five times on each task through each gateway: 810 runs in total.

How each gateway served context and handled data

Both gateways had the same business context and access to the same systems. They differed in how that context reached the model, and in where the work on the data happened.

	Open-source MCP gateway	Connect AI Gateway
Business context	All of it, about 8,600 characters, loaded into every session up front	The Context Engine selects only the entries a question needs and serves them when the agent asks
Getting the data	A separate request to each system, with raw records returned a page at a time	One request across all three systems
Processing the data	The model matches accounts, filters, totals, and ranks in its own reasoning	The Virtual Data Layer does the matching, filtering, totaling, and ranking, and returns a short result

We configured the open-source gateway as well as we could. Its business definitions were word for word the same as Connect AI Gateway's, and its system descriptions and know-how for requesting data were written for its own tools, including what those tools can't do. Its IT service-management server offered ready-made lookups, such as open ticket counts for each company, that saved its agents work. And no run could start unless its context had loaded.

With the context the same, the difference comes from how it was served, on demand through the Context Engine, and where the computation ran, server-side in the Virtual Data Layer. The study doesn't separate the two, so it attributes the difference to both together.

From tokenomics to taskonomics

AI costs are still quoted per token. Providers price that way, comparisons lead with it, and estimates for agent workloads usually start from it. What a business buys, though, is completed work: a right answer, or a right action, on its own data. The measure that captures this is cost per correct task, with failed attempts counted in. A cheap model that gets the answer wrong hasn't saved anything, and a correct answer that took an hour of searching and retrying isn't cheap, even if every token in it was.

So the question we tested was this: when the models choose their own steps, how much does the way a gateway serves context and handles data change what a correct task costs?

What it took to get a correct answer

Counting correct answers only, models working through Connect AI Gateway needed about a tenth of the tool calls and 93% fewer tokens (the median when comparing the same model on the same task, across the 44 cases where both gateways got it right).

That efficiency shows up in cost. Through Connect AI Gateway, a correct, governed task cost \$0.26. Through the open-source gateway, the same task cost \$5.67: 21 times as much, with failed attempts counted in.

Two numbers sit behind that result. The agents completed 83% of their tasks correctly through Connect AI Gateway and 36% through the open-source gateway. And for the same 405 tasks on each side (27 models, three tasks, five runs each), total model spend was \$89 through Connect AI Gateway and \$828 through the open-source gateway. The open-source gateway spent more than nine times as much and got less than half as many tasks right.

"Governed" has a specific meaning for the write task. A result only counted if the agent added exactly the right accounts to the review queue and no others. An agent that added all 21 correct accounts plus two extra did not count as a success, because in a live system those two extra reviews are real work for someone and a real error in the record.

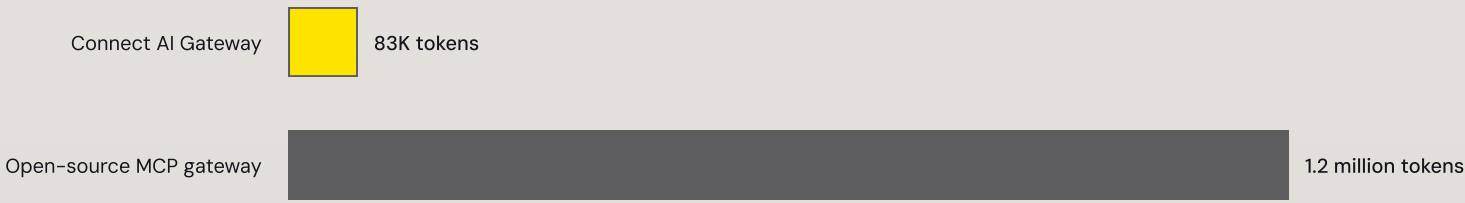


Figure 1. Tokens for the typical (median) correct answer, across the 44 cases where the same model got the same task right through both gateways: 83K through Connect AI Gateway against 1.2 million through the open-source gateway, 93% fewer.

Readers of *The 178x Cost Spread* will notice a smaller number. Both studies counted only right answers; they differ in how the work was set up. The 178x study was deterministic: the steps were built into the gateway ahead of time, and it compared the cheapest and most expensive AI models. This study is probabilistic: the models chose their own steps on every run, and 21x compares two gateways, pooled across all 27 models, with the same business context on both. That scope matters for reading these numbers: they describe agents working out each task for themselves, not workflows built in advance. For individual models the gap reached 156x. The results also build on each other: the savings from cheaper models are only real if those models can do the work, and through the open-source gateway, the cheapest models rarely could. Likewise, the 83% success rate isn't comparable to the 98.5% accuracy in CData's earlier MCP accuracy benchmark: that study scored individual queries, while this one scores multi-step tasks across three systems, averaged over all 27 models, including economy models.

Only frontier models stayed reliable through the open-source gateway

On Connect AI Gateway, 16 of the 27 models completed every task correctly. Through the open-source gateway, 3 did, and all three were frontier models, the most expensive tier we tested. No model did better through the open-source gateway than through Connect AI Gateway.

The drop was steepest at the bottom of the price range. Ten models never produced a single correct result through the open-source gateway, and eight of those ten succeeded on Connect AI Gateway, three of them on every task. Economy models as a group completed 78% of their tasks on Connect AI Gateway at \$0.06 per correct task. Through the open-source gateway, they completed 17%, at \$1.48 per correct task.

Model tier	Connect AI Gateway: tasks correct	Connect AI Gateway: cost per correct task	Open-source MCP gateway: tasks correct	Open-source MCP gateway: cost per correct task
Frontier (9 models)	93%	\$0.47	66%	\$6.48
Mid (7 models)	78%	\$0.26	28%	\$7.24
Economy (11 models)	78%	\$0.06	17%	\$1.48

16 models got every task right through Connect AI Gateway; 3 did through the open-source gateway.

● Connect AI Gateway ● Open-source MCP gateway

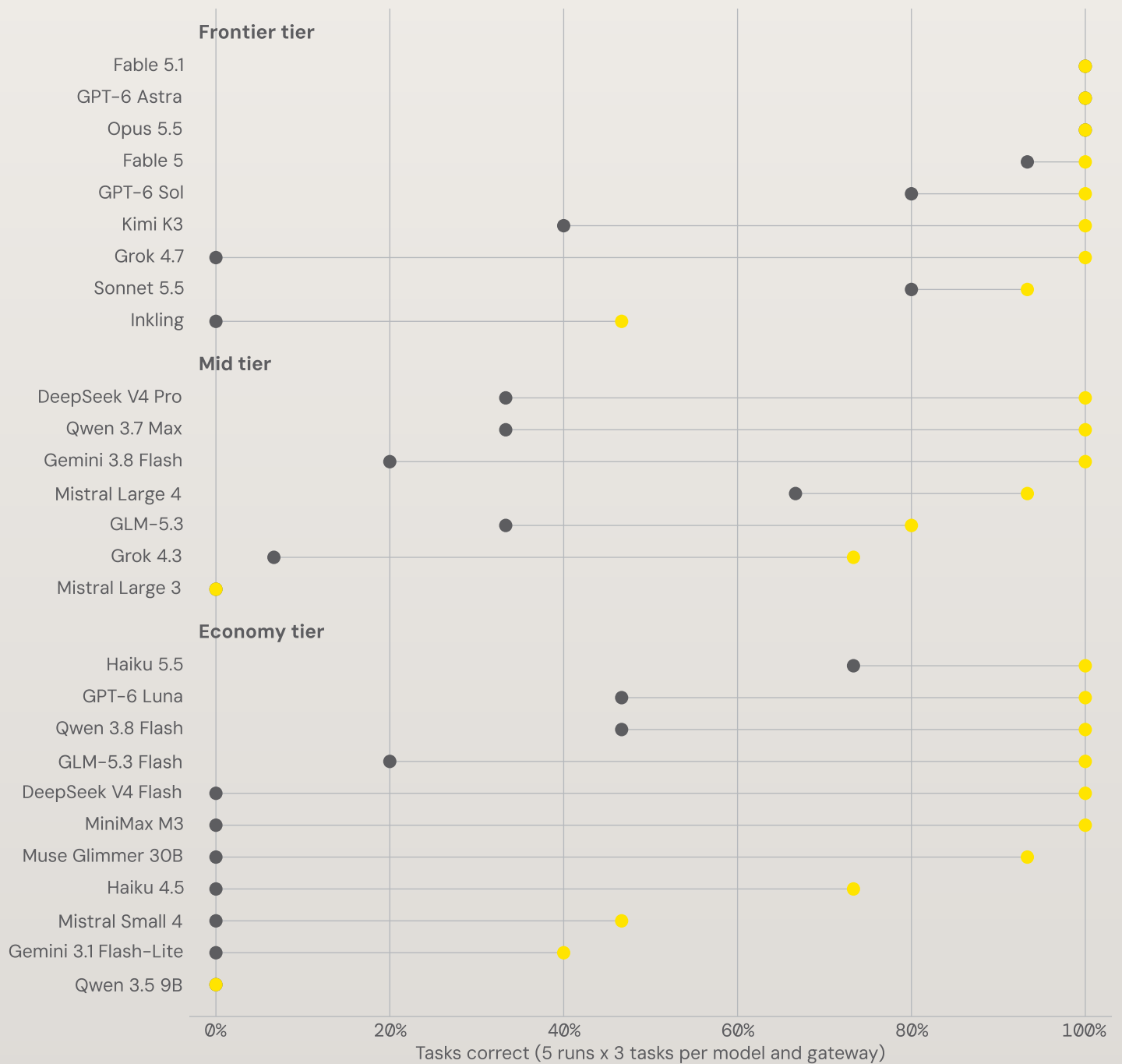


Figure 2. Tasks completed correctly (of 15) by model, on each gateway, grouped by tier.

With a pass-through gateway, your teams pay frontier prices to get a reliable answer at all. With a gateway that serves context on demand and does the math, cheaper models can do the job, which is the point *The 178x Cost Spread* made, now with a reason behind it.

Wrong answers, delivered confidently

The open-source gateway's failures were also harder to catch. On the two read tasks, more than one answer in four (27%) through the open-source gateway was confidently wrong, against one in fourteen (7%) on Connect AI Gateway. A confidently wrong answer looks exactly like a right one: a ranked list of accounts, a stated method, and no sign that anything went wrong. Someone acting on it has no way to know.

The write task showed the same pattern with higher stakes. Through the open-source gateway, almost three times as many runs wrote records they shouldn't have (29% vs. 10%), and those runs came from 17 different models, against 5 on Connect AI Gateway. Each of those runs added accounts to the review queue that didn't qualify, in a live system.

The reason is the arithmetic. Through the open-source gateway, the agent had to work out every account's health and eligibility itself, across hundreds of records from three systems that it could see only a page at a time. Every page it misread or skipped became an error in the answer, and the errors added up to results that were plausible and wrong.

How Connect AI Gateway does it

Both gateways carried the same business context and had access to the same systems. Connect AI Gateway is built in three layers, and the difference in this study came from two of them.

Connectivity reaches each system live, with its schemas, objects, and relationships known from the start. Here, the open-source gateway could reach the same three systems through each system's own MCP server.

The Context Engine unifies business context into one graph that any model can use: system context, semantic definitions, and company knowledge, plus a fourth kind, a data model of curated views, which this study deliberately left out. When the agent asks, the engine walks that graph and returns only what the question needs, so the context window stays small. The open-source gateway carried the same business context but loaded all of it into every session. This study loaded three of the four kinds of context as written entries; it didn't test how the Context Engine gathers or refines context over time.

The Virtual Data Layer handles live data across systems in a single request, doing the joining, filtering, totaling, and ranking where the data lives and returning a short result. Through the open-source gateway, the model fetched raw records from each system and did that work in its own reasoning. With the Virtual Data Layer, the model writes the question and Connect AI Gateway does the math.

That difference holds even when both gateways got the answer right. Among the 44 model-and-task pairs with correct answers on both gateways, every one needed more tool calls and cost more through the open-source gateway, typically about eleven times as much. Across all runs, a typical run took 47 tool calls instead of 8. That extra cost comes from making the model do work the gateway can do for it.

What this means

Three implications follow from these results.

Measure tokens and cost per correct task, not price per token. A per-token price says little about what a correct answer costs. The same models produced correct work at \$0.26 and \$5.67 depending on the gateway they worked through.

Your gateway decides which models you can afford to use. If the gateway serves context on demand and does the computation, cheaper models become viable, and model choice becomes a cost decision again. If it doesn't, your teams pay for the most expensive models and for the extra work those models have to do.

How context is served matters as much as having it. Both gateways had the same business context, and a correct task still cost \$0.26 through one and \$5.67 through the other.

When your teams evaluate a gateway, three questions get to the heart of these results:

- What does a correct task cost through your gateway, measured on our data and our models?
- Where do joins, filters, and calculations across systems happen: in your gateway, or in the model?
- How does your gateway deliver business context: on demand, or in every prompt?

About CData

CData provides connectivity, context, and governance for AI-to-data interactions within organizations. The platform offers live access and data replication across hundreds of enterprise sources, with semantic intelligence designed to improve the accuracy of AI workloads. CData supports AI infrastructure for Anthropic, Databricks, Microsoft, Google, Palantir, and more than 10,000 customers.

Disclaimer. This benchmark was conducted internally by CData Software, Inc. ("CData"). All testing, evaluation, and analysis were performed by CData employees. Results have not been independently verified.

Key limitations:

- Testing was conducted using the specific tasks, data, and conditions described in this paper.
- The dataset was synthetic, built to be structurally representative of enterprise systems rather than drawn from any specific customer.
- Each model ran five times per task on each gateway.
- Both conditions used business context authored by CData, adapted to each gateway's tools.
- One open-source MCP gateway configuration was tested; the open-source gateway read ITSM data from a mirror of the instance's objects, and Connect AI Gateway read the instance or the same mirror.
- Runs affected by source-system outages were rerun under identical settings.
- Models were accessed via public APIs on the dates recorded in the run data.
- Results reflect controlled test conditions and may not represent performance in all production environments.

Performance variability: Actual performance may vary significantly based on deployment configuration, data characteristics, network conditions, query patterns, LLM model versions, API changes, and platform-specific factors. The models tested may have been updated since testing was conducted.

Not professional advice: This report is provided for informational purposes only and does not constitute professional, legal, or technical advice. Organizations should conduct their own independent testing using data and queries representative of their specific use cases before making purchasing or implementation decisions.

Full methodology, aggregated results, and source code are available at github.com/CDataSoftware/connect-ai-gateway-benchmark.