INTELLIGENCE WAREHOUSE · RESEARCH

# Traversal is the unit of intelligence

When an AI system answers a question about your company, the quality of the answer is decided by what made it into the context window. Most enterprise systems fill that window with similarity search, which has a problem no better embedding fixes: inside a large company, most internal text resembles other internal text, and the differences that matter are not in the wording. The Intelligence Warehouse fills the window by walking a typed graph of the company instead. This paper works through what that changes in practice: why answers hold up better, why the system's decisions follow the same path the company's own decisions follow, how a walk crosses departmental lines to weigh a trade-off against company-level goals, and what happens to the compute bill. The numbers come from a seeded simulation shipped with the paper.

**Authors** Akhil Singh · Nidhi Prabhu · Aniruddha Tammewar **Date** September 2026

**Method** Seeded simulation, real graph builds **Reproducible** seed 20260919

| 21% | -3.5% | −50.9% | 0.72 |
|---|---|---|---|
| OF THE RULES AN ANSWER NEEDS SURVIVE FIXED-BUDGET RETRIEVAL AT COMPANY SCALE | CONTEXT GROWTH PER WALK AS THE GRAPH GROWS 60× | COST PER QUERY, MONTH 1 TO MONTH 12 | LLM CALLS PER QUERY BY MONTH 12 |

The first number is an accuracy ceiling: no model can apply a rule that retrieval failed to hand it. The second is the cost side of the same fact: a walk reads roughly the same amount of material whether the graph has 800 nodes or 48,000, while the retrieval baseline's context grows **+964%** over the same range. The last two describe what happens after deployment: on a fixed company, the cost of answering falls as decisions harden into deterministic code, and by the end of the first year the average question no longer needs a full LLM call.

## 01 The context assembly problem

The model is rarely the weak point in an enterprise AI system anymore. Whether an answer comes out right is decided upstream, by which rules, numbers, and facts were put in front of the model, and which were left out.

There are three ways to fill a context window.

The first is to load everything. This works for a small wiki and stops working quickly after that. Cost grows with the size of the corpus, and accuracy degrades too, because the paragraph that matters has to compete for the model's attention with thousands that don't.

The second is retrieval by similarity, which is the standard architecture today. Embed every document, and for each question pull in the chunks that sit closest to it in embedding space. This works well on a varied corpus. It works badly inside a large company, and the reason has nothing to do with embedding quality. Companies are repetitive. A company with 192 departments has, give or take, 192 discount policies, and they read almost identically because most were copied from one another and lightly edited. The policy that applies to a given case differs from the other 191 in ways that barely show up in the text at all: it belongs to a different department, it is owned by a different role, its threshold is a different number. That information was never in the wording. It sits in the relationships between the policy and everything around it, and a vector index has nowhere to put relationships.

The third way is to store the relationships explicitly and use them. Model the company as a graph in which every node and every edge has a type, and answer a question by walking it: start from the things the question names, follow only the kinds of edges that are relevant to this kind of question, read one-line summaries as you go, and load full detail only for the handful of nodes the walk actually lands on. The amount of material read depends on the length of the walk, not on the size of the company. And each fact in the answer got there because a specific edge led to it, which means someone can check the edge.

The rest of the paper is about the third option. Sections 04 to 06 cover what it does to answer quality: accuracy, faithfulness to how the company actually makes decisions, and decisions whose variables live in different departments. Sections 07 and 08 measure what it does to cost.

## 02 The substrate: one typed graph

The Intelligence Warehouse keeps one graph for the whole enterprise. Business concepts are typed nodes, relationships are typed edges, and people never edit the graph directly; it is compiled out of the flows they build and the answers they give the system along the way. The full architecture is documented elsewhere. What follows is the minimum needed for the traversal argument.

| NODE TYPE | WHAT IT IS | TYPED EDGES IT CARRIES |
|---|---|---|
| `org_unit`, `person` | The structure: departments, roles, people | `reports_to`, `owns`, `belongs_to` |
| `kpi` | A metric with a target and an owner | `decomposes_into`, `measured_by` |
| `data_source` | A system of record, carrying its EDA profile | `feeds`, `measured_by` |
| `flow`, `block` | Executable work, deterministic or agentic | `belongs_to`, `acted_on_by` |
| `circuit` | A decision circuit (§05): one recurring decision in executable form | `governs`, `escalates_to` |
| `rule` | One codified constraint, from an SOP or a resolved conflict | `part_of` |
| `entity` | The things acted on: customers, SKUs, distributors, sites | `acted_on_by` |

Two properties of this substrate carry the rest of the paper.

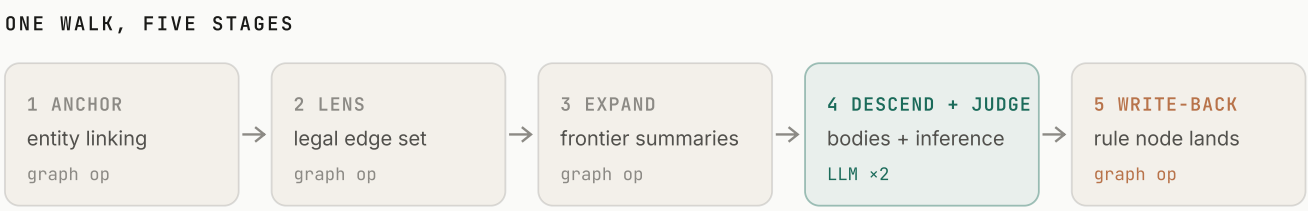
The first is that **typed edges let the question shrink the graph.** A question about who can approve something has no reason to cross a `decomposes_into` edge, and a question about why a metric moved has no reason to cross `reports_to`. Declaring what kind of question is being asked cuts the reachable graph down to a small slice of itself, before anything else happens. The IW's five lenses (Hierarchy, Function, Flow, Entity, Decision) are exactly this mechanism: each lens is a set of edge types that a walk is allowed to follow.

The second is that **every node exists at two resolutions.** There is a one-line summary, about 45 tokens: type, name, one sentence. And there is the full body, about 350 tokens: the contract, the rules, where it came from, the data profile if it has one. A walk reads

summaries while it moves and pays for full bodies only at the nodes it settles on. The same idea as progressive disclosure in a well-organised document, applied to a company.

## 03 Anatomy of a walk

A traversal runs in five stages, and only one of them touches the model.




Stages 1–3 and 5 are graph operations: microseconds of CPU, no tokens billed. The model is involved only in stage 4.

`1 · anchor` **Anchor resolution.** Link the things the question names to nodes. "The Chennai distributor" becomes one `entity` node; "this discount" becomes the `circuit` that governs trade discounts. Where names are unambiguous this is a lookup; where they are not, one small inference settles it.

`2 · lens` **Lens selection.** Classify what kind of question this is and load the edge types that are legal for it. An authority question walks `reports_to / owns / escalates_to`; a root-cause question walks `decomposes_into / measured_by / feeds`. Section 07 measures how much this one step cuts.

`3 · expand` **Frontier expansion.** Breadth-first from the anchors along the legal edges, reading only summaries, with a token budget instead of exhaustive reachability. No model involved; this is adjacency-list work.

`4 · descend` **Descent and judgment.** The nodes that matter get their full bodies loaded, and the model is called with what the walk assembled: the instruction, the question, the frontier summaries, the grounding bodies. A typical walk makes **two** calls, one to reason at the decision point and one to write the answer with its audit trail.

`5 · write-back` **Write-back.** If the walk ran into a contradiction (two systems disagreeing on a value, a rule that conflicts with an exception), the human resolution comes back as a rule node attached to the circuit. The next walk through the same territory reads the rule instead of rediscovering the problem.

```
# the loop, sketched
anchors  = resolve(question)                # entity linking
legal    = lens(intent(question))           # edge-type set
frontier = expand(anchors, legal,
                  budget=SUMMARY_BUDGET)    # graph ops, no tokens billed
ground   = descend(frontier, question)      # full bodies where it matters
answer   = infer(instruction, question,
                 frontier.summaries, ground)
if answer.conflict: write_back(rule_node)   # next walk reads the rule
```

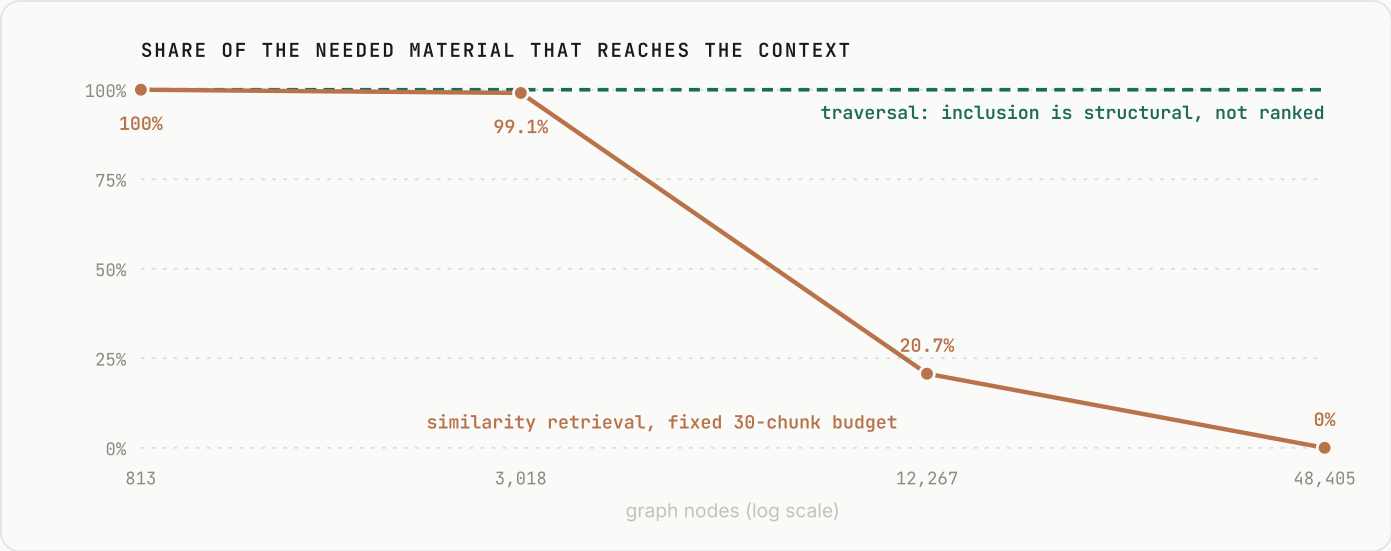
It is worth being explicit about what the model is no longer being asked to do. It does not choose tools, does not re-read a growing transcript, does not sift a pile of retrieved chunks for the relevant one. The control flow runs as code. The model gets a small amount of material that a structural process already vouched for, and reasons about it.

## 04 Why the answers hold up

A model cannot apply a rule it never saw. Whatever else is true of an architecture, the share of the needed material that actually reaches the context window is a ceiling on its accuracy. So that is the first thing to measure.

### Whether the right material arrives at all

In the simulation, each question has a true grounding set: the specific rule nodes, thresholds, and escalation records that the correct answer depends on, between 6 and 14 of them. We gave the retrieval baseline a fixed budget of 30 chunks, about 10,500 tokens, which is roughly double what an entire traversal walk spends on everything. Then we asked, at each company size, what share of the grounding set makes it inside that budget.




Average share of the needed material that lands inside a fixed 30-chunk similarity context, by company size. At 813 nodes essentially all of it arrives. At **12,267** nodes it is **20.7%**, and at **48,405** nodes close to none, because the near-duplicates from other departments outrank the real thing and the budget fills up with look-alikes. Traversal is not plotted, because it does not rank: a rule is in context when the walk reaches it over a `governs` or `part_of` edge, and that test does not change as the company grows.

The mechanism is worth spelling out. Similarity search returns the chunks most similar to the question, and in a big company the chunks most similar to the question are mostly other departments' versions of the same policy. Widening the budget helps, and §07 shows what that costs; at any fixed budget, growth in the company eats the recall.

### What failure looks like

The second accuracy problem is the shape of the failure. When retrieval misses, it does not come back empty. It comes back with the nearest look-alike, the model writes a fluent answer from the wrong department's policy, and nothing in the output shows that anything went wrong. A good share of what gets called hallucination in enterprise deployments is this: the model doing its job properly on the wrong material.

A walk fails differently. To name an approver, it has to arrive at a `person` node over an `escalates_to` edge. If the edge is not there, there is no name to produce, and the system reports that it could not find one. A visible "no path" is a much cheaper failure than a confident wrong name. Someone fixes the edge, and that class of failure is gone.

The same holds for numbers. When a walk cites a metric, the figure is read from the `data_source` node that the metric is `measured_by`, with the source's freshness and profile attached, not from whichever quarterly report chunk happened to rank highest that day.

| QUESTION | HOW A WALK ANSWERS IT |
| --- | --- |
| Who approves this, at this amount? | `circuit →rules→ threshold →escalates_to→ person` |
| Who is accountable for this metric? | `kpi →owns⁻¹→ person →belongs_to→ org_unit` |
| If we change this node, what breaks? | downstream reachability over `feeds / governs / depends_on` |
| Can this team see this data? | a path-existence test under boundary edges; the answer is allow or deny, not a paraphrase of a policy |

### Accuracy compounds, because errors are addressable

Every answer carries the path it came from, so when a reviewer catches a wrong answer, they can see which edge or rule produced it, and the fix is a change to the graph. From that point on, every walk that crosses the same spot gets the corrected version. An embedding index offers no equivalent. You can re-chunk, re-rank, or fine-tune, but you cannot repair one fact and know it stays repaired. On a graph, each correction is permanent and shared, which is why accuracy drifts upward with use instead of staying wherever the index left it.

## 05 The walk decides the way the company decides

Consider how a ₹40 lakh discount actually gets approved at a real company. Not the version written in the policy PDF, the real one.

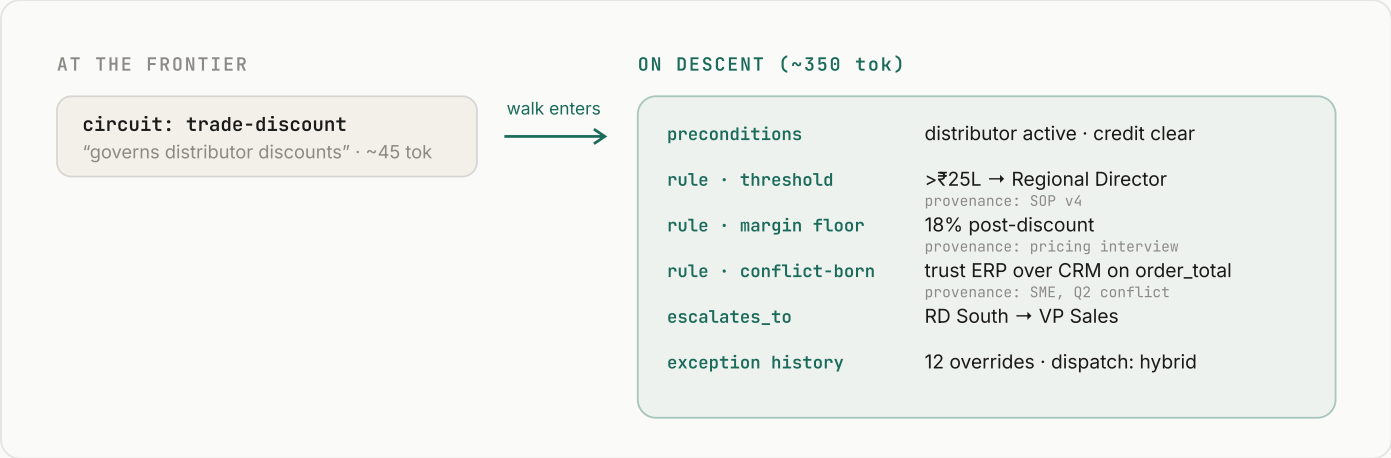
An experienced sales operations person starts by checking that the distributor is even eligible. She pulls the account's history, because a discount for a growing account and a discount for a churning one are different decisions that happen to carry the same number. She checks the amount against the threshold and knows that above ₹25 lakh it stops being her call. She weighs the margin floor against what she knows about this account, and if the case is a genuine exception, she takes it to the Regional Director rather than approving it herself. None of that ordering is decoration. It is the company's decision procedure, and it took years of pricing mistakes to arrive at.

The claim of this section is that a walk reproduces the procedure itself, not a text summary of it. Eligibility is checked first because precondition edges are the cheapest hops. The account's history is present because the entity node carries it. The threshold binds because it is a rule node holding the actual number, not a sentence about the number. The model is

consulted at the point where the company consults a person's judgment, and sign-off lands where the company requires sign-off, because `escalates_to` terminates in a human task rather than a suggestion. The system is not imitating the organisation from its documents. It executes the same dependency structure the organisation executes.

### The circuit: one recurring decision, kept whole

The unit that stores a decision is the decision circuit, a small subgraph that keeps together everything the company knows about one recurring decision.




- **Preconditions**, which say when this decision applies at all.
- **Rule nodes**, each carrying its origin: this one came from the SOP, that one from an interview with the pricing analyst, that one from a conflict a subject-matter expert resolved last quarter.
- **The escalation path**, as edges to the people who own the exceptions, in order.
- **The exception history**, the record of when the rules were overridden and why.
- **A dispatch policy**: deterministic, agentic, or hybrid.

The exception history deserves a sentence of its own. Most of a company's decision quality lives in its exceptions, not in its rules; the rules are what everyone already agrees on. Because the circuit retains its exceptions and the agentic step reads them, the judgment the system exercises on an unusual case is anchored to how this company resolved similar cases before, rather than to what a general-purpose model considers reasonable behaviour.

### Hardening decisions in the same order the company would

Companies already do, by hand, something the IW does mechanically. When a judgment call comes up often enough and keeps getting resolved the same way, someone writes it into an SOP, and from then on it is procedure rather than judgment. Circuit promotion is the same move. A circuit that begins agentic accumulates worked examples with every walk; once its judgment is demonstrably stable (the simulation requires 50 consistent runs), the system proposes compiling it to deterministic code, keeps a 3% sampled audit on the result, and demotes it if its inputs drift. Section 08 prices this. The point here is fidelity: decisions harden most-frequent and most-stable first, which is the order the company itself would choose.

The system also inherits the company's unevenness about control, deliberately. Where the organisation is strict (anything writing to a system of record, anything crossing a team boundary), promotion requires human review. Where it is relaxed, promotion goes through. The gates are read off the graph, so the system ends up exactly as careful as the company decided to be, in each place, rather than uniformly cautious or uniformly loose.

> **On the word "skills".** Circuits are close cousins of the skill libraries that agent frameworks are converging on: small self-describing procedures, loaded when needed. We use a different word because the origin is different. A skill is a file someone wrote for an agent. A circuit accumulates out of the company's own operation, and every rule inside it points back to the SOP, the interview, or the resolved conflict it came from. How circuits get built up in the first place, from documents to interviews to decision traces to reinforcement from live usage, is a separate paper.

## 06 Decisions that cross functions

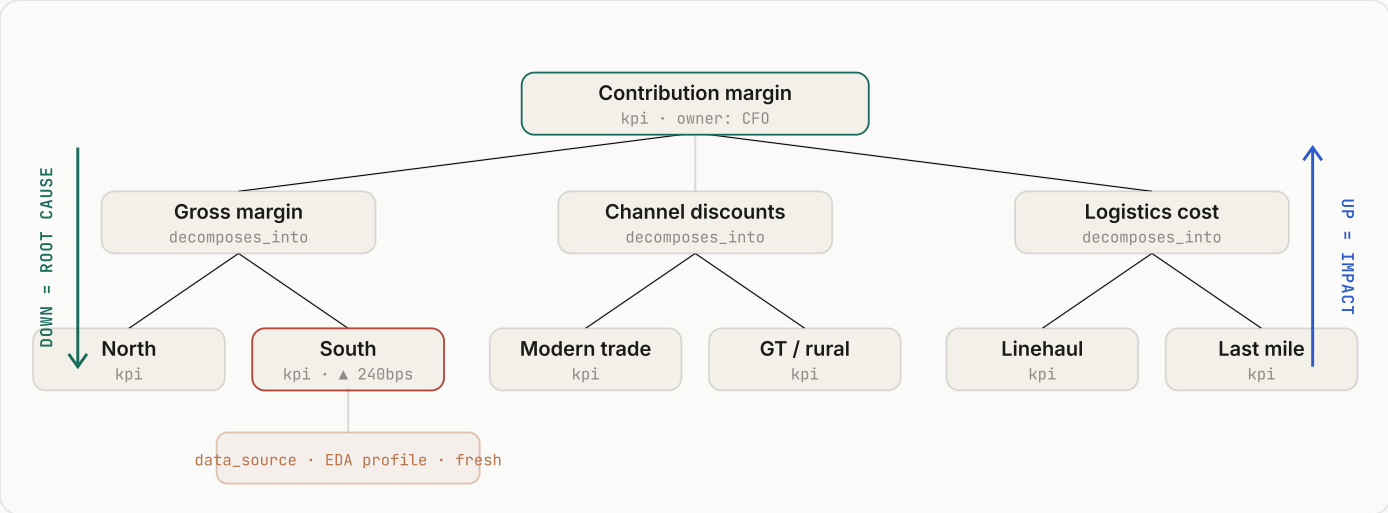
The decisions that matter most in a company are multivariate, and the variables belong to different departments.

Whether that ₹40 lakh discount is a good idea does not depend on the pricing policy alone. It depends on the margin position, which is finance's variable. On the inventory of the SKUs this distributor moves, which is supply chain's. On the account's churn risk, which is sales'. And on whether marketing has a promotion landing in the same quarter. Each function tracks its own variable and optimises its own target, and nobody owns the joint decision. This is why companies settle such questions in meetings, late, with whichever subset of the right people happened to be in the room.

The graph changes the mechanics, because functions that never talk to each other still share nodes. The distributor that sales acts on is connected to the SKUs that supply chain plans. The discount that pricing governs feeds the margin KPI that finance owns. A walk that starts from the discount decision does not stop at a departmental boundary, because there is no boundary in the graph, only edges. Reading the inventory position of the affected SKUs is two hops. It used to be a meeting.

### Shared goals give the trade-off a common currency

Extra context alone would not make the decision better; the walk also needs a way to compare unlike things. That is what the KPI tree provides. Every local metric has a path upward: regional volume rolls into revenue, channel discounts roll into contribution margin, and the two meet at a company-level node with an owner and a target.




So when a walk weighs volume against margin, it can express both sides in the same currency by walking each one up to their shared ancestor. The decision gets scored against the company's goal instead of against whichever local metric the deciding team happens to carry. That is the difference between a sales system that maximises sales and a company system that happens to sit in sales.

The tree also turns the two hardest questions in any operating review into ordinary traversals. Walking down is root-cause work: start at the metric that moved, descend `decomposes_into` edges following the child that explains the parent's variance, and stop at a leaf that has a data profile attached. The arithmetic on the way down is code, not inference; the model is called once at the bottom to say what the decomposition means and what to do about it. Walking up is impact work: start from a supplier or an entity, ascend `feeds` and `decomposes_into` edges, and see which company-level metrics the problem reaches, with the full chain attached as evidence.

### Combinations no single team could see

Some of what a cross-functional walk surfaces is genuinely new, in the sense that no one team had the pieces. Approving a discount to push volume on SKUs that supply chain cannot restock for six weeks means paying to accelerate demand you cannot serve. Both facts are ordinary on their own, one in pricing's world and one in supply chain's. The connection exists only at the SKU nodes both functions touch, which is where the walk goes. The same applies in the other direction: when pricing changes a rule node, every sales flow that depends on it is downstream over real edges, so the affected teams get told about the change rather than discovering it in the quarter's numbers.

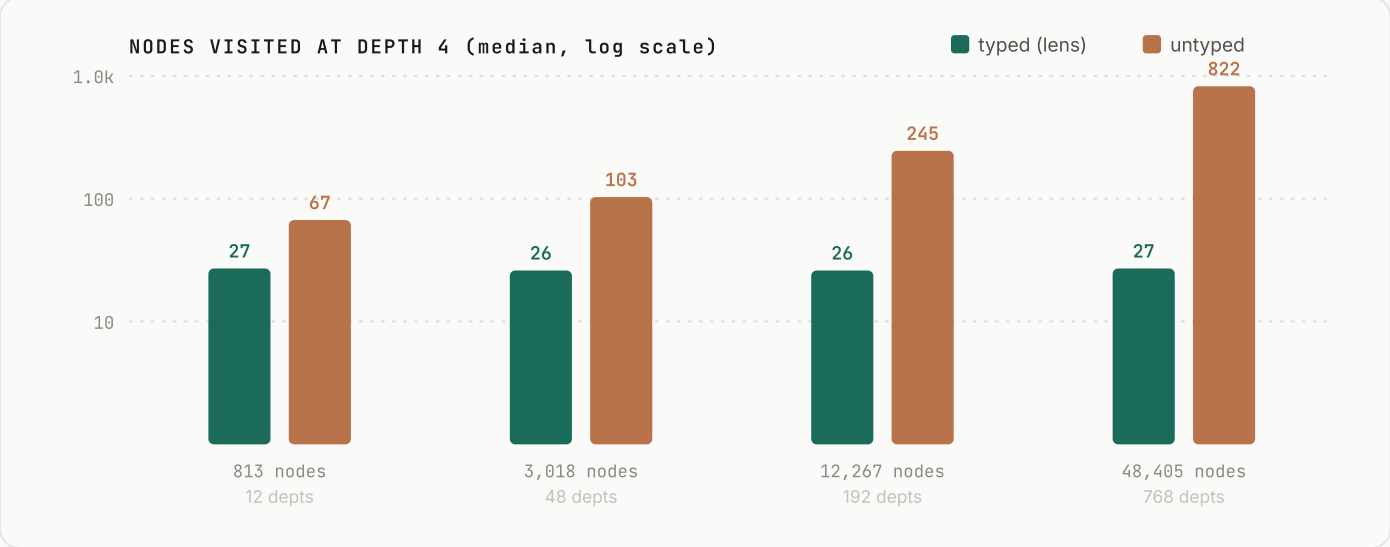
## 07 Compute, part one: the flat line

The cost claim is that what a walk reads depends on the walk, not on the size of the graph. We tested it by building graphs and running the walks.

The simulation constructs synthetic enterprise graphs at four sizes, from 12 departments (**813** nodes) to 768 departments (**48,405** nodes), using the node and edge taxonomy from §02. At each size it runs 300 queries across the three intent classes and measures the frontier and the bill.

### What edge typing cuts, measured

Each query expands breadth-first to depth 4 from its anchor twice: once following only the edge types its lens allows, once following everything. Both runs are plain graph operations on the same adjacency list, so the comparison is direct.

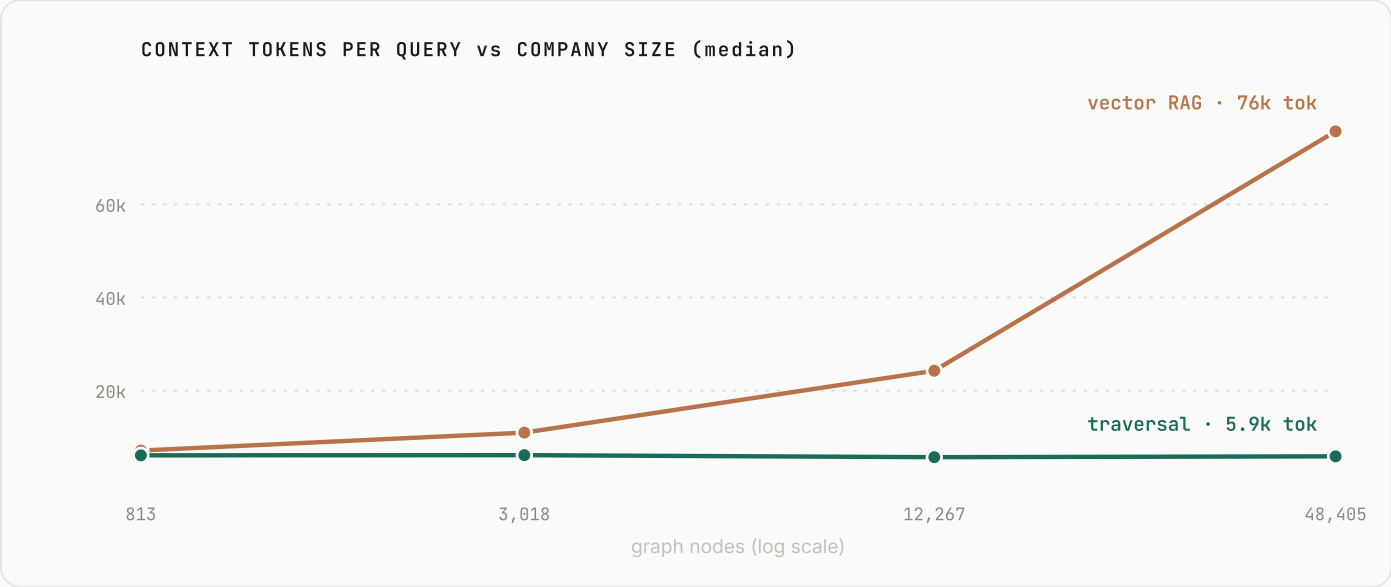


Median nodes visited at depth 4, per graph size. The typed frontier stays at roughly 26 nodes at every scale. The untyped frontier grows with the graph and reaches **822** nodes at the largest size, a **30.4×** difference produced by edge typing alone.

The intuition behind the flat green bars: an untyped neighbourhood grows with the density of the company, but a typed neighbourhood grows only with the structure of the question, and questions do not get more complicated when companies get bigger.

### Tokens per query as the company grows 60×

A walk's context is the frontier summaries plus the grounding bodies plus instruction and answer. For the baseline we were deliberately generous: it gets a perfect embedding, a fully cached prefix, and is only charged for the one thing similarity genuinely cannot separate, near-duplicates. To hold 95% recall of the grounding set (the §04 problem, solved with money), it has to widen its retrieval as the company grows. The confusability probabilities behind that are documented in the script and swept below.




Median context tokens per query as the graph grows from **813** to **48,405** nodes (log-scaled x-axis). Traversal moves **-3.5%** across the range. The baseline moves **+964%** and ends at **12.9×** the traversal cost.

| DEPARTMENTS | GRAPH NODES | TYPED FRONTIER | UNTYPED FRONTIER | TRAVERSAL TOK/QUERY | RAG K FOR 95% RECALL | RAG TOK/QUERY | RAG RECALL AT FIXED K=30 |
|---|---|---|---|---|---|---|---|
| 12 | 813 | 27 | 67 | 6,090 | 12 | 7,120 | 100% |
| 48 | 3,018 | 26 | 103 | 6,125 | 23 | 10,970 | 99.1% |
| 192 | 12,267 | 26 | 245 | 5,695 | 61 | 24,270 | 20.7% |
| 768 | 48,405 | 27 | 822 | 5,875 | 208 | 75,720 | 0% |

> **How much the result depends on the confusability model.** We swept the model's probabilities from half to double. At half, a world with unusually distinctive departments, the baseline still ends at **7×** the traversal cost at the largest size. At double it reaches **24.4×**. The traversal line does not move in either case, because none of its numbers depend on similarity.

## 08 Compute, part two: the falling curve

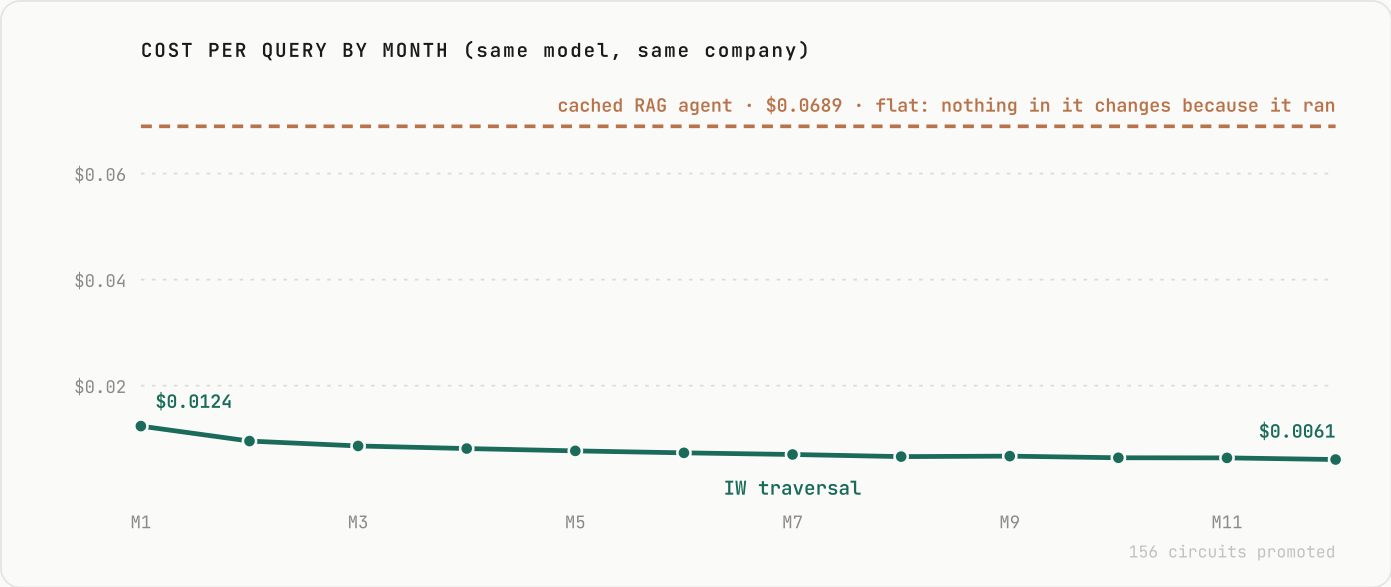
The flat line says growth in the company does not raise the price of a question. The second claim is stronger: on a fixed company, the price of a question falls as the system runs, with no change of model.

Three mechanisms drive it, and all three were introduced in earlier sections.

- **Promotion** (§05). Circuits whose judgment has stabilised get compiled to deterministic code, so the head of the traffic distribution stops paying for inference at all, apart from a 3% sampled audit.
- **Path caching.** Business questions repeat. A repeated walk over unchanged data is a lookup, and the cache entry is invalidated the moment any underlying data node refreshes, so reuse never outlives freshness.
- **Write-back** (§03, stage 5). Every resolved conflict becomes a rule node, and a rule node replaces exploration. The part of a walk that used to weigh alternatives now reads one constraint, so circuits that have been argued over get shorter. We cap the effect at 60% of the original walk.

We ran 12 months of workload against the 192-department company: 8,000 queries a month, Zipf-distributed across 420 circuits, since real question traffic concentrates on a head of recurring decisions. The baseline is the same generous retrieval agent as before,

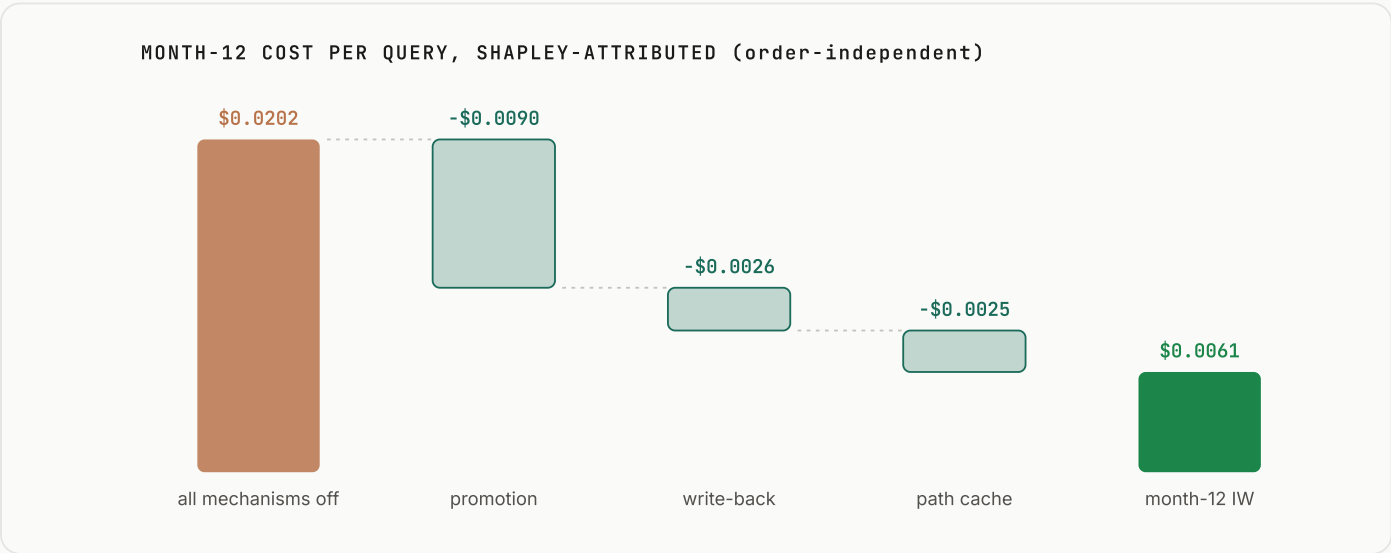
and its line is flat for a structural reason: nothing in that architecture changes because it ran.




Cost per query by month. The walk opens at **$0.0124**, already **82%** under the baseline on the strength of §07 alone, and ends at **$0.0061**, a **50.9%** decline, **91.2%** under the baseline. By month 12, **156** of the 420 circuits run as deterministic code and the mean query uses **0.72** LLM calls.

## Where the decline comes from

The three mechanisms are independent switches in one cost function, so their contributions can be separated exactly. We ran all eight switch combinations on the same seed and attributed the decline with Shapley values, which makes the split independent of the order you imagine applying them in.




Month-12 cost per query, from every mechanism off (**$0.0202**) to all three on (**$0.0061**). Promotion accounts for **63.8%** of the decline; write-back and caching split the remainder. A workload with a longer tail would shift weight from promotion toward caching without changing the endpoint much.

### The assumption doing the most work

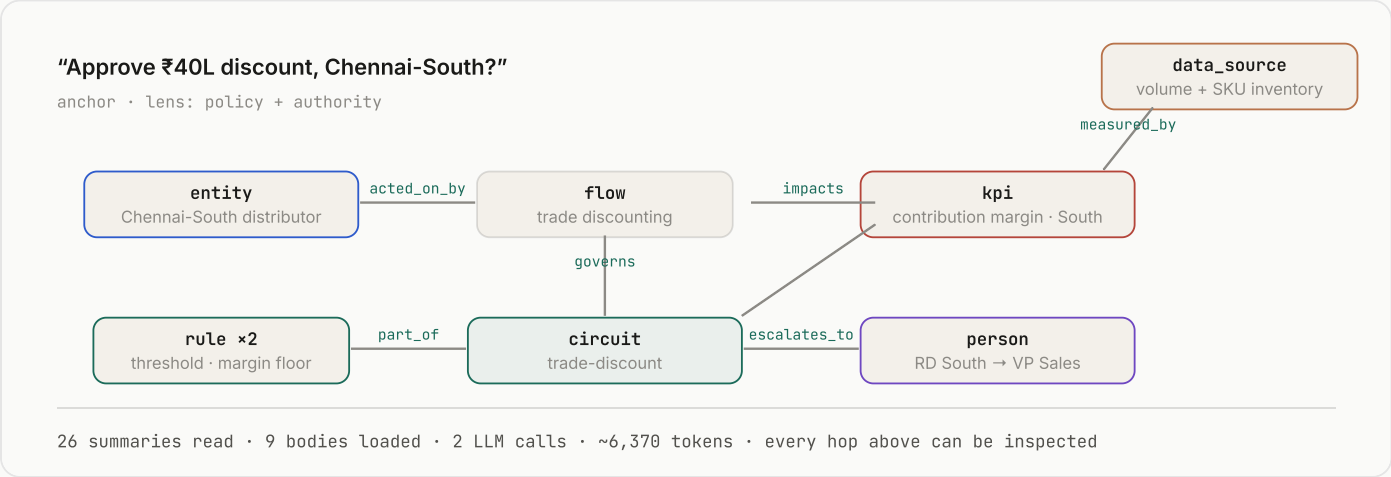
Everything above leans on one number: the share of a company's decision traffic that sits on circuits whose logic ever stabilises. We used 70%. The table below reruns the full 12 months at other values.

| SHARE OF CIRCUITS THAT EVER STABILISE | 12-MONTH DECLINE | MONTH-12 VS BASELINE |
|---|---|---|
| 30% | -30.2% | -87.1% |
| 50% | -43.4% | -90.3% |
| 70% | -53.1% | -91.9% |
| 90% | -74.3% | -96.1% |

At 30%, a pessimistic reading in which most decisions stay judgment calls forever, the curve still falls by about a third and ends 87% under the baseline, carried by the flat line and the caching. We think 70% is closer to the truth for operational traffic, since an SOP is precisely a company's own record that a decision recurs and stabilises. But the result does not hinge on being right about that.

## 09 A worked walk: the ₹40 lakh discount

Here is the question from §05 run end to end: *"Should we approve a ₹40L trade discount for the Chennai-South distributor?"* It exercises everything this paper has argued: the authority chain, the KPI impact, the decision circuit, and a variable from another function.



| # | STAGE | WHAT HAPPENS | COMPUTE | TOKENS |
|---|---|---|---|---|
| 1 | Anchor | "Chennai-South distributor" resolves to its `entity` node; "trade discount" resolves to `circuit:trade-discount` | graph op | 0 |
| 2 | Lens | The question classifies as policy plus authority; legal edges: `governs, part_of, escalates_to, belongs_to, decomposes_into` | graph op | 0 |
| 3 | Expand | Depth-4 typed frontier: 26 node summaries. Ignoring edge types, the same expansion would visit 245 nodes | graph op | 0 |
| 4 | Descend | 9 full bodies load: the two rule nodes (threshold: above ₹25L needs the Regional Director; margin floor: 18%), the circuit with its exception history, the escalation chain (RD South, then VP Sales), the contribution-margin KPI and its South-region child, the distributor entity, and its trailing-volume and SKU-inventory `data_source` nodes with their EDA profiles. The inventory node is the cross-functional hop: it belongs to supply chain's side of the graph and enters this walk through the distributor's SKUs | graph op | 0 |
| 5 | Judgment 1 | The exception assessment. The amount exceeds the threshold; the distributor's volume trend and churn signal argue for it; the inventory position on its top SKUs says the extra volume can actually be served. All of that is weighed against the margin floor and the circuit's exception history | **LLM call** | 4,780 in / 180 out |
| 6 | Judgment 2 | The recommendation, with its audit trail: approve at ₹34L within the RD's authority, or escalate the full ₹40L to VP Sales. Each claim in it cites the path it came from | **LLM call** | 1,150 in / 260 out |
| 7 | Write-back | No conflict surfaced this time; the walk's trace is appended to the circuit's exception history | graph op | 0 |
| | **Total** | **2 LLM calls, 7 stages, every hop inspectable** | | **~6,370** |

The retrieval baseline at this company's size spends a median **24,270** tokens on the same question (§07), and there is a part of stage 6 it cannot produce at any budget: the ₹34 lakh figure. That number is not written anywhere. It exists only as the meeting point of the threshold rule and the RD's authority bound, two nodes connected by edges. A system that retrieves text can quote the policy. Applying it takes the graph.

And because distributor discounts sit near the head of the traffic distribution, this walk does not stay at 6,370 tokens. The circuit accumulates examples, qualifies for promotion, and some months in, the whole table above becomes a deterministic code path with a sampled audit, at which point the marginal cost of this question is close to zero and the audit trail is better than it was.

## 10 What this is not

Four adjacent ideas get mixed up with typed traversal. The differences are architectural.

| | VECTOR RAG | GRAPHRAG | CONTEXT GRAPH | ONTOLOGY / KG | IW TRAVERSAL |
|---|---|---|---|---|---|
| **Unit of retrieval** | chunk | community summary | decision trace | triple | typed walk |
| **Grounding** | statistical | statistical over clusters | historical | structural | **structural** |
| **Cost vs company size** | grows | grows (index + summaries) | grows with history | flat-ish | **flat (§07)** |
| **Crosses functions** | only if the documents do | by topic, not by operation | records that a crossing happened | in schema only | **yes, via shared nodes (§06)** |
| **Learns from use** | no | no | records, doesn't learn | no | **yes (§08)** |
| **Executable** | no | no | no | no | **yes, circuits dispatch** |

**GraphRAG** builds a graph over documents in order to summarise a corpus, and it is good at that. But its nodes are extracted topics rather than operational objects; nothing in it holds a threshold, escalates to a person, or carries a dispatch policy. **Context graphs** record decision traces after the fact. Useful forensics, and honest about being forensics, but a record of last quarter's decision is not machinery for making the next one. **Ontologies** got the typing right decades ago and stopped at schema: no walk economics, no two-resolution nodes, no promotion path, and no good answer to who keeps them current. The IW's answer to that last question is that nobody maintains the graph as a job. The graph is a side effect of running the company on it; every flow built, conflict resolved, and circuit promoted is maintenance that already happened.

## 11 The honest ledger

What traversal costs, and where this analysis could be wrong.

WHAT THE IW SPENDS THAT THE BASELINES DON'T

- **Building the graph.** Hydrating circuits out of SOPs and interviews is real, front-loaded work. Cold-start research seeds the base graph from public material, and after that the graph is maintained by operation rather than by a curation team, but a company unwilling to make the initial investment should not expect the curves in this paper.
- **Freshness machinery.** Caching and promotion are only safe with honest invalidation: a promoted circuit has to demote when its input distributions drift, and a cached walk has to expire when an underlying data node refreshes. The retrieval baseline never reuses anything, so it never needs any of this engineering.
- **Promotion risk.** Compiling judgment into code is the most aggressive move in the architecture, and logic frozen wrongly is worse than inference paid for. The guards are the sampled audit, drift-triggered demotion, and mandatory review for promotions that touch a system of record.

WHERE THIS COULD BE WRONG

- **The confusability model is a model.** We did not run a real embedding over a real company's corpus. We modelled the near-duplicate problem with documented tier probabilities and swept them across a fourfold range. The direction of the effect is well established; the magnitude at any particular company depends on how repetitive its documents actually are, and the script accepts measured numbers in place of ours.
- **The promotable share is a claim about companies, not software.** If your decision traffic is genuinely novel every time, promotion never fires and you keep the flat line and the caching. The sweep in §08 shows the floor.
- **Everything conditions on the graph being roughly right.** A wrong edge produces a confidently wrong walk. That is why edges carry owners and provenance, why write-back routes through people, and why §04's argument that errors are addressable matters as much as the raw numbers: the graph will contain mistakes, and the architecture's real claim is that its mistakes can be found and stay fixed.

## 12 Reproduce it

Every number on this page is read from the output of one seeded script. The graphs are built, the expansions are run, and none of the figures are typed by hand.

RUN IT YOURSELF

```
$ python3 iw-traversal-sim.py
# → iw-traversal-data.json    (every figure on this page)
# → iw-traversal-monthly.csv  (the 12-month table, per row)
```

sim `iw-traversal-sim.py`: standard-library Python, no dependencies. Builds the four graphs, runs typed and untyped expansions, computes the fixed-budget recall experiment, prices both architectures, runs the 12-month workload with the Shapley decomposition, and both sensitivity sweeps.

data `iw-traversal-data.json`: the aggregates this page renders from, embedded below.

seed `20260919`: changing it moves individual draws and leaves the shape of every curve where it is.

To test it against your own environment, put in your provider's pricing, your department count, your query mix, and, if you have retrieval logs, measured confusability in place of our tiers. The two central results, context flat in company size and cost falling with use, come from the architecture rather than from the draws.

---

INTELLIGENCE WAREHOUSE · RESEARCH · SEPTEMBER 2026

**Authors.** Akhil Singh, Nidhi Prabhu, and Aniruddha Tammewar.

**The claim, in one line.** Walk a typed graph of the company instead of searching for text that resembles the question. The material an answer depends on reaches the model because a walk arrived at it (at company scale, fixed-budget retrieval delivers **21%** of it), decisions follow the company's own procedure including its escalations and exceptions, trade-offs that span departments get scored against shared goals, and the cost of a question is flat in company size and falls **50.9%** across the first year of use.

Figures rendered live from `iw-traversal-data.json`. Simulation seed 20260919 · real graph builds at 4 sizes · Claude Sonnet-class pricing.