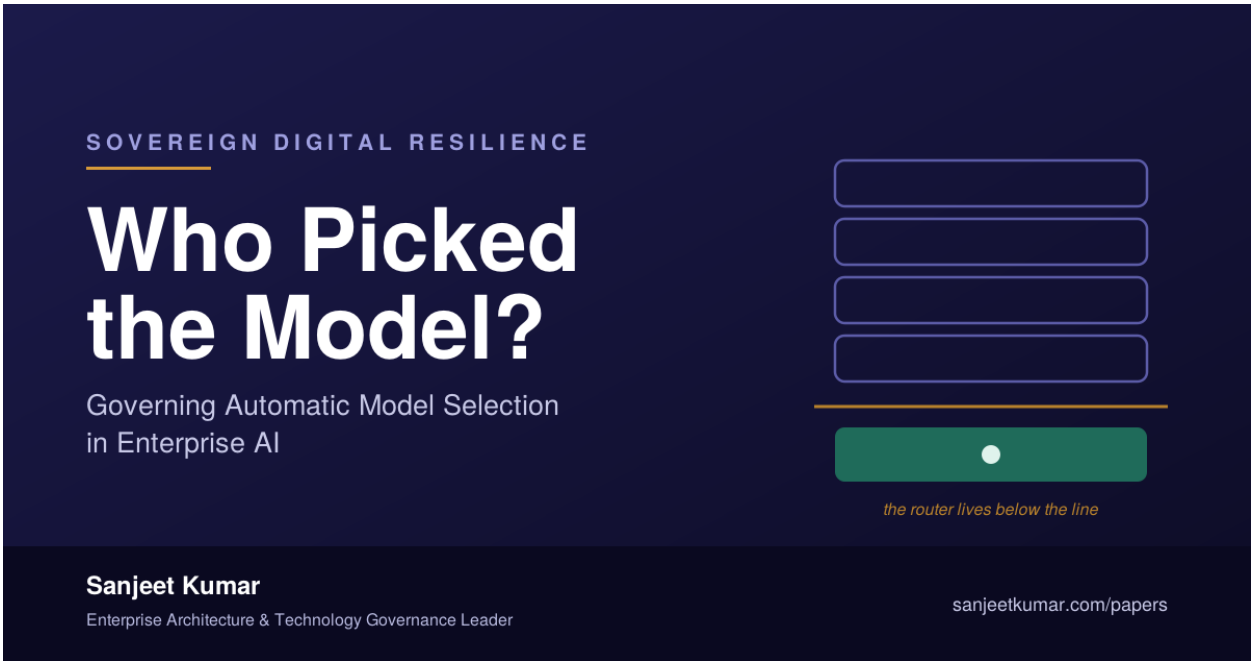


SOVEREIGN DIGITAL RESILIENCE · WORKING PAPER

Who Picked the Model?

Governing Automatic Model Selection in Enterprise AI

Sanjeet Kumar — Enterprise Architecture & Technology Governance Leader

Founder & Principal, Robotiqa Technologies Ltd.

Sovereign Digital Resilience · Decision Custody · AI Governance — Revision 2 · November 2026 · Views are the author's own.

ORCID 0009-0003-1677-684X · DOI: 10.5281/zenodo.23585802

All deposits and versions: sanjeetkumar.com/papers/who-picked-the-model

Keywords: Model Routing · Automatic Model Selection · AI Gateway · AI Control Plane · Selection Provenance · Decision Custody · AI FinOps · Token Economics · AI Governance · Data Sovereignty · Enterprise Architecture

In one page

Choosing a model used to be a procurement decision> made once, by named people, with a contract behind it. Somewhere in 2026 it became a runtime decision, made for every single request, by a router, on whatever cost/latency/availability signal the router was built to chase, with nothing behind it but a policy that nobody in the organization owns. Routing as such is not new; the research is 3 years old and the products are everywhere. What is new is the institutional question. An organization that has to answer for its decisions is now making thousands of small consequential decisions a day that no one reviews and nothing records. In *The AI Gateway Advantage* I argued for owning the seam between your teams and your models. This paper argues that the router now sits on that seam. It separates the rules a router must obey from the signals it is free to optimize, names the record a router must leave behind (I call it selection provenance), and shows why the router, and not the model, is the new concentration point you will one day need an exit from.

The Dial That Disappeared

On 24 June 2026, GitHub made 'Auto' the default and only model-selection experience on the free and student plans of Copilot [1]. The little dial that let a developer pick which model answers was simply gone from those plans. Auto itself was not news; it had been generally available since December 2025 [2], and from May 2026 it had been choosing by task (weighing reasoning depth, code-generation complexity, tool-orchestration needs and so on before picking a model, with a small discount for anyone who let it) [3]. Amazon Bedrock had shipped Intelligent Prompt Routing in April 2025 [4]. Microsoft Foundry's model router is described, in Microsoft's own documentation, as a trained model that selects the model for each prompt in real time, to deliver performance and save on compute [5].

A few weeks before I sat down to write this, someone who builds these systems for a living told me that choosing the model was about to become a thing of the past. The AI would pick the model, per request, to optimize token cost. He was right. He was also a little late. This is not a forecast; it is a product changelog.

I want to make one observation about that changelog and then spend the rest of the paper on what follows from it. On those plans the dial did not disappear. It moved. Something still picks the model, for every request. The questions worth asking are who (or what), on which signal, under whose rules, and whether anyone can find out afterwards. In most organizations I have looked at, the honest answers are: a router, cost, none, and no.

From Procurement to Runtime

Think about how a model used to get chosen. A team evaluated 2 or 3 candidates. Someone with authority signed. Procurement attached a contract with a data-processing clause and a jurisdiction in it. The decision was made once, by people with names, and it left a record because the organization's own processes insisted on one. Slow, yes. Accountable, also yes.

Now the same choice under automatic selection. A policy (very often the vendor's default) looks at each request as it arrives and picks a model out of a pool. The decision is made per request, by software, and it leaves a record only if someone went out of their way to build one. Three things changed at the same time>

	Procurement decision	**Runtime decision**
Who decides	Named people with authority	A routing policy, usually the vendor's
How often	Once, then annually	Every request
What gets recorded	A contract and an approval	Usually nothing the organization holds

In the gateway paper I described two architects (or developers) doing the same task with the same models: one burning ten million tokens on brute force, the other spending half a million by decomposing the problem, curating minimal context, and matching the model tier to the task. The Token Efficiency Framework in that paper exists to find the second architect in the telemetry and spread her habits across the organization. Model-tier selection, in other words, was treated as a human judgment; something to coach.

That judgment has now been lifted out of the person and dropped into the router. The coaching layer does not go away. It changes object. Instead of teaching people which tier suits which task, the organization has to govern the thing that now holds that knowledge> review its choices against outcomes, version its policy, and report cost per outcome by routing policy rather than by person. A policy change becomes something you measure the way you measure a code change, because that is exactly what it is.

What a Router Optimizes, and What It Cannot See

Let me be precise about what is new here and what is not, because the research community reached routing well before the product teams did. FrugalGPT showed in 2023 that cascading cheaper models ahead of expensive ones cuts cost sharply with little loss in quality [6]. RouteLLM, in 2024, trained routers on human preference data to decide per query whether a strong or a weak model was called for [7]. The cloud routers productized the idea; the coding assistants made it a default. Audit logging and provenance have their own literature and I lean on it. None of that is my claim. My claim is about what happens when this routing decision is made inside an institution that has to answer for its decisions, sometimes decades later.

So, what does a router optimize? Cost per token, latency, availability, cache position: the signals it can actually see at the moment the request arrives. Some routers also carry a learned guess at how hard the prompt is and how much quality it deserves. All reasonable things to optimize, and the savings are real.

The mainstream advice now says this in plain words. A July 2026 McKinsey Quarterly piece on managing AI demand tells CIOs to route each task to the lowest-cost model that can deliver the required quality, and closes on the objective of making the economically efficient choice the default choice [16]. Risk is on its list of routing criteria, as 1 input among 5, weighed against cost, quality, latency and availability. Its control plane carries approved-model lists and premium-model approvals, which is an eligibility rule in all but name. What it does not say is which comes first when the approved model is the expensive one and the cheap one is right there, or what gets written down when the router decides. I have no quarrel with the optimization. My quarrel is with the word default, and with risk sitting on the same list as price>

What can a router not see? Everything nobody told it. It cannot see that the prompt carries a citizen's record and may only be processed in-jurisdiction. It cannot see that a statute, a contract or a policy requires a specific model or provider for this class of work. It cannot see that the answer is about to be used to propose a change to a system of public record, where the approver needs to know exactly which

system made the proposal. And it cannot see that the task has an accuracy floor below which a cheaper answer is not a cheaper answer at all, just a wrong one.

Goodhart's Law, which I raised in the gateway paper for people, comes back for routers with more force. A router rewarded on cost per token will drift quality, and the drift is invisible without the outcome linkage that the framework's second layer provides. A person who starts reaching for the cheap model too often can be coached. A router doing the same thing is doing precisely what it was configured to do, and no dashboard on the market is going to call that a problem!

Five Consequences

Each of the 5 consequences lands on a paper already in this series. I will keep them short, because the mechanism is the same each time: a decision that used to be made by a person, once, is now made by a policy, continuously, with no record.

Sovereignty. In the gateway paper I wrote that sovereignty is not a paragraph in a policy document; it is an enforced routing policy. Well, a cost router is also a routing policy, and the two can disagree. If the cost rule is evaluated first (or is the only rule, which is the common case) a sensitive prompt goes to the cheapest eligible model wherever that model happens to run, and nobody decided that. The organization did not defect from its sovereignty commitment. The commitment just had no representation in the room at the moment of the request.

Provenance. *Decision Custody* argued that an institution must be able to reproduce, explain, defend and answer for every consequential decision made in its name, for as long as the consequences last. 'Which model answered' is now a per-call fact. Without it the decision record names a system which was, at the moment of the decision, one of several, chosen by a rule nobody wrote down. A decision whose author is a pool is not a decision anyone can reproduce.

Exit. *Exit-Capable by Design* treated vendor concentration as a sovereignty risk and asked the only question that matters: what does it cost to leave? The router is a new concentration point. A tool that hides the dial has made the exit harder; not because the models are locked in (they are interchangeable by design) but because the policy that chooses between them is now the vendor's, and the organization has no record of what it would need to rebuild somewhere else.

Authority. *Governing Agentic AI* placed every agent action on a 4-class ladder by blast radius and reversibility, and ruled that a better model next quarter does not promote an action class. Automatic selection makes that rule more necessary and adds a corollary. Actions in the two gated classes (advisory and propose, where an accountable human approves each instance) need a pinned, named model, because the approver is approving a specific recommendation from a specific system. Supervised actions may route inside a named, pre-approved set. Autonomous actions may route freely. A cheaper model next quarter does not promote an action class; neither does a router.

Measurement. The gateway paper's fourth layer, coaching, was built around the practitioners who chose well. Where the router chooses, the practice turns into a review of the router: its selections against outcomes, by policy version, quarterly. The router gets the review I gave to people and, unlike people, it can be held to a version number.

Eligibility Before Optimization

The design rule of this paper is simple to state and, in my experience, almost never implemented. A router's rules come in 2 kinds, and the order between the kinds is fixed.

Eligibility rules decide which models may answer at all. A router may never relax them, and they are evaluated before any optimization signal gets a look-in. Consider this>

1) Obligation. A specific model, provider or jurisdiction is required by law, contract or policy for this class of work.

2) Sovereignty. This data class may reach only these endpoints, in these jurisdictions.

3) Action class. The downstream action needs a pinned model (gated classes), a named set (supervised), or no constraint (autonomous).

last) Fitness. The task has an accuracy or validation floor, and a model that passes the first 3 rules but fails the floor is still out. Cheaper is not a tier; it is a price.

Optimization signals then choose among the eligible. Cost, latency, availability, cache position, a learned difficulty estimate; any of them, in any mix the organization finds useful. This is where automatic selection belongs, and inside this envelope it is permitted without reservation. >PS: I would go further and say that inside the envelope you should let the router do whatever it likes and never second-guess it by hand.

One more rule gives the structure its teeth. When the eligible set is empty, the request fails closed. A sovereign endpoint being offline is not permission to route elsewhere; it is an outage, and it gets recorded as one. This is the rule a cost router will never implement on its own, because from where it sits an unavailable model is simply a model with infinite latency and the next cheapest one is right there. The organization has to say, in advance and in configuration, that some requests are better left unanswered than answered in the wrong place.

This is also the difference between 2 defaults. "The economically efficient choice as the default" [16] puts the router first and the organization on exception: the cheapest model answers unless someone has intervened. Eligibility before optimization puts the organization first and the router free inside the envelope: the permitted set answers, and the cheapest member of it wins. The savings are the same. The answer to who decided is not.

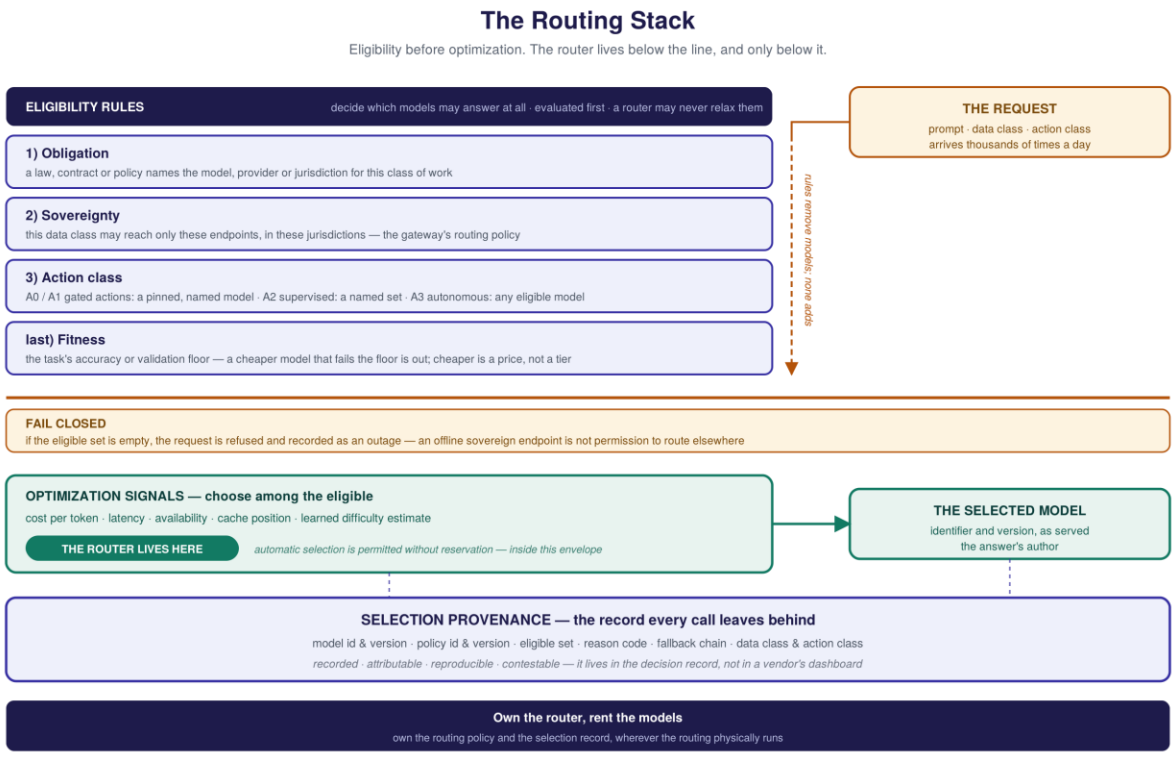


Figure 1 — The Routing Stack. ***Four eligibility rules above the line; optimization signals below it. The router lives in the lower half only. 'Fail closed' sits on the line.***

I expect 2 objections. The first: this adds latency to every call. It does not add inference; it adds a lookup. Four rules checked against a request's data class and action class is the kind of policy check every API gateway already does, and it runs in the same place. The second: most organizations have no idea what the data class or action class of a given request even is. True, and that is not an objection to the rule. That is the finding.

Selection Provenance

Everything above converges on one artefact: the record a router must leave behind. I am going to call it selection provenance, by analogy with the intent provenance of *The Intent Bottleneck* and the chain of custody that *Decision Custody* demands of decisions. It has the same 4 properties> recorded, attributable, reproducible, and last) contestable, meaning a named owner can be asked, years later, why this model answered, and the answer exists.

The minimum record, per call:

Field	Why it is there
Model identifier and version, as served	The answer's author

Field	Why it is there
Routing policy identifier and version	The rule that chose it
Eligible set at the time of the call	What else was permitted
Reason code (which rule or signal decided)	Why this one
Fallback chain, if any was taken	What failed first
Data class and action class of the request	What the rules were protecting

Two honesty clauses, because the sceptic will ask and should.

First, this record makes the selection reproducible, not the output. Same input, same policy, same eligible set, same choice; or a logged reason why not. It does not make the model's answer reproducible, because a provider can change a model's behaviour under the same identifier, and routinely does. The record can prove that the organization chose what it said it would choose. It cannot prove that the model is still the model it was. That is a limit, and it is the same limit the gateway paper put on telemetry: the record is evidence of the organization's conduct, not a warranty on the vendor's.

Second, the record is cheap. A few hundred bytes of identifiers per call; not the prompt and not the completion. Digest-sized, which is the very property that lets *The Cold Standard* commit a digest of every consequential record exhaustively while keeping full content only for a sample. The telemetry-volume objection is real for prompts and completions. It does not apply to 6 identifiers.

The record has to survive 4 failure modes, and the design should be tested against each one> 1) the selected model becomes unavailable mid-policy, and the fallback chain is what actually answered; 2) the policy changes between the decision and the audit, so the record must name the version that applied, not the version that is current; 3) the provider quietly retires or alters the model behind the identifier, and the record can show only that the identifier was the one chosen; and last) historical routing cannot be replayed because the eligible set was never written down, which is why the eligible set is in the table, and why the table is the minimum and not a wish-list.

Where does the record live? In the decision record, the same place *Governing Agentic AI* puts the proposal, the approver's identity, the diff and the timestamp; not in a vendor's dashboard.

McKinsey expects the AI control plane to become "the system of record for intelligence consumption" [16]. It should, and the finance function needs exactly that. But a record of consumption says how many tokens were spent, by whom, on what. It does not say which model answered or which rule chose it, and it is not the record an auditor will ask for. The 6 fields above are the difference between a system of record for consumption and a system of record for selection.

Regulators are arriving at the same spot from the other direction. The EU AI Act's record-keeping obligations for high-risk systems and Canada's Directive on Automated Decision-Making both ask, in

substance, for the system that made or informed a decision to be identifiable after the fact [8, 9]. A router that leaves no record turns that into an unanswerable question, and the organization will find this out during the audit, which is the worst possible time.

Own the Router, Rent the Models

Automatic selection can live in 3 places, and only one of them hands the organization the policy and the record by construction.

It can live inside the tool (Copilot's Auto, say, or the equivalent in other coding assistants). You get the discount and, if the plan allows, a hover that tells you which model answered. You do not get the policy, and the record is the vendor's.

It can live inside the provider (Amazon Bedrock's Intelligent Prompt Routing, Microsoft Foundry's model router). You get a managed router with real logs and real controls, inside one provider's catalogue, under that provider's policy. A managed router can certainly be auditable; and the gateway paper's point stands that provider controls stack beneath a gateway rather than replace one.

It can live inside your own gateway (LiteLLM-class routing, or the routing layer of whichever gateway product sits behind your endpoint). Here the policy is yours, in version control, and the record lands in your database because that is where the gateway's telemetry was going anyway.

The gateway paper's principle was: own the endpoint, rent the gateway. This paper adds its successor> own the router, rent the models. More precisely, own the routing policy and the selection record, wherever the routing physically runs. A provider's router behind your endpoint, under your eligibility rules, writing to your telemetry, is owned in every sense that matters. The same router sitting in front of your teams, under its own defaults, writing to a dashboard you cannot export, is rented in every sense that matters.

The exit test follows directly, and it is the gateway paper's test with one word changed. Can you replay last quarter's routing decisions through a different gateway and get the same selections? If yes, the policy is yours and the models are interchangeable. If no, the seam has drifted into the router, and the exit you thought you had engineered is a sprint shorter than you think; in the wrong direction. McKinsey's own sourcing advice ends in 5 verbs: buy, build, host, route, and switch [16]. Route and switch are verbs an organization can only use in the first person if the policy and the record are its own.

Five Questions for a Board

For directors who read this far and want somewhere to put it, 5 questions turn the argument into oversight:

> **1. For our AI-assisted decisions, can we say today which model answered each one?**
>
> **2. Does any routing in our estate optimize cost before it checks jurisdiction or the accuracy floor?**
>
> **3. Who owns our routing policy, by name, and is it version-controlled?**
>
> **4. If our coding or AI tool removed the model dial tomorrow, what would we lose, and would we even notice?**
>
> **last) Which of our consequential actions are tied to a pinned model, and who decided that list?**

The AI Gateway Advantage governed the inference path: what models get called, at what cost, with what telemetry. *Governing Agentic AI* governed the action path: what agents may do, and how the organization proves it. This paper governs the selection path, the decision that sits between the two, made thousands of times a day by a component that until recently was a dial. The dial has moved. The accountability has not. In institutions whose obligations outlast their technologies that gap is the whole subject, and it closes the way every gap in this series closes: by architecture, in advance, with a record. Some of us have been drawing these lines for years; the router has just made this one visible.

Revision note

Revision 2 (November 2026) names the position this paper argues with. In July 2026 McKinsey Quarterly set out the mainstream case for managing AI demand at scale [16]: an AI control plane between users, applications, agents and models; cost attributed to business outcomes rather than tokens; each task routed to the lowest-cost model that meets the required quality; and the economically efficient choice made the default. The first two are the gateway paper's argument, reached from the finance side, and I am glad of the company. The last two are what this paper is about. The order of a router's rules is not in that advice, the record each call should leave is not in it, and jurisdiction does not appear. Sections 3, 5 and 6 now say where the two positions part, and why the difference is a gate and a record, not a dashboard.

References

1. GitHub, "Changes to model selection for Free and Student plans," GitHub Changelog, 24 June 2026. github.blog/changelog
2. GitHub, "Auto model selection is generally available in GitHub Copilot in Visual Studio Code," GitHub Changelog, 10 December 2025. github.blog/changelog
3. GitHub, "Auto model selection now routes based on your task in VS Code," GitHub Changelog, 20 May 2026. github.blog/changelog

4. Amazon Web Services, "Amazon Bedrock Intelligent Prompt Routing is now generally available," AWS What's New, 22 April 2025. aws.amazon.com/about-aws/whats-new
5. Microsoft, "Use model router for Microsoft Foundry," Microsoft Learn, 2026. learn.microsoft.com/azure/ai-foundry/openai/how-to/model-router
6. Chen, L., Zaharia, M., Zou, J., "FrugalGPT: How to Use Large Language Models While Reducing Cost and Improving Performance," arXiv:2305.05176, 2023.
7. Ong, I., Almahairi, A., Wu, V., Chiang, W.-L., Wu, T., Gonzalez, J. E., Kadous, M. W., Stoica, I., "RouteLLM: Learning to Route LLMs with Preference Data," arXiv:2406.18665, 2024.
8. European Union, Regulation (EU) 2024/1689 (Artificial Intelligence Act), Official Journal of the European Union, 2024 — record-keeping obligations for high-risk AI systems.
9. Treasury Board of Canada Secretariat, Directive on Automated Decision-Making, Government of Canada.
10. Kumar, S., *The AI Gateway Advantage: Value over Spend. Smart Work over Hard Work.*, sanjeetkumar.com/papers/ai-gateway-advantage, 2026. doi:10.5281/zenodo.21970942
11. Kumar, S., *Governing Agentic AI: In Systems of Public Record, Governance Is the Accelerant*, sanjeetkumar.com/papers/governing-agentic-ai, 2026. doi:10.5281/zenodo.21971009
12. Kumar, S., *The Intent Bottleneck: When AI Compresses Delivery, Strategy Becomes the Constraint*, sanjeetkumar.com/papers/the-intent-bottleneck, 2026. doi:10.5281/zenodo.21943397
13. Kumar, S., *Exit-Capable by Design: Vendor Concentration Is a Sovereignty Risk*, sanjeetkumar.com/papers/exit-capable-by-design, 2026. doi:10.5281/zenodo.21970869
14. Kumar, S., *Decision Custody*, sanjeetkumar.com/papers/decision-custody, 2026. doi:10.5281/zenodo.21959271
15. Kumar, S., *The Cold Standard*, sanjeetkumar.com/papers/cold-standard, 2026. doi:10.5281/zenodo.22073326
16. Sachdeva, P., Lala, W., Javaji, A., Takkar, K., Arora, P., "The cost of intelligence: How CIOs can manage AI demand at scale," McKinsey Quarterly (QuantumBlack, AI by McKinsey), 20 July 2026. mckinsey.com/capabilities/quantumblack/our-insights/the-cost-of-intelligence-how-cios-can-manage-ai-demand-at-scale

AI-assistance disclosure> This paper was drafted and refined with AI assistance. The arguments, positions, judgements, and final text are the author's own. © 2026 Sanjeet Kumar. Views are the author's own and do not represent any employer.