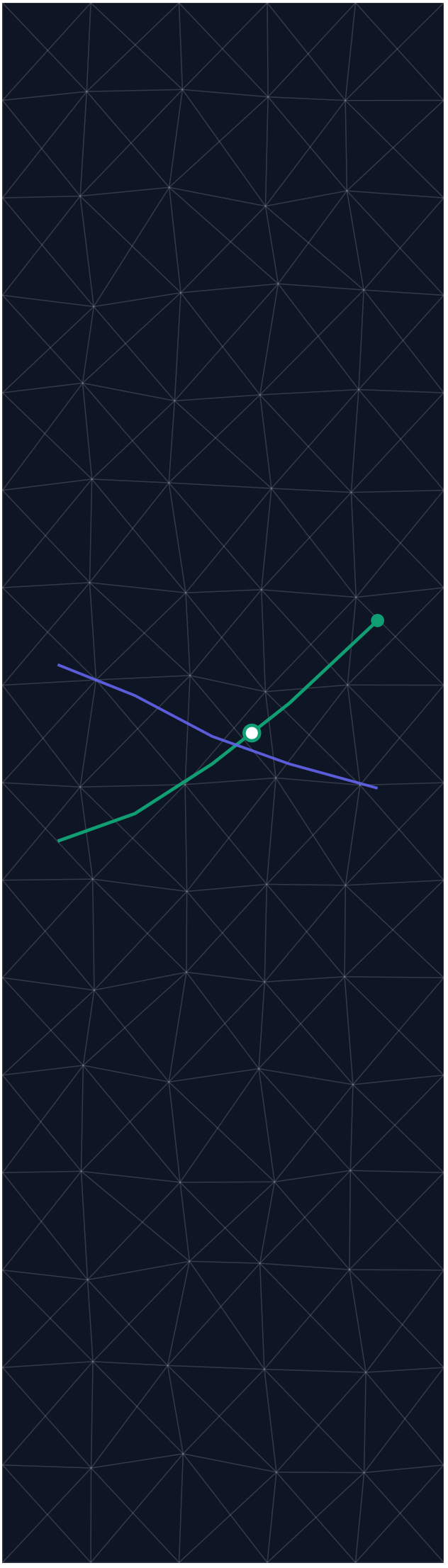
AN ESSAY ON OPERATIONAL AI ARCHITECTURE



Operational AI Becomes a Distributed Systems Problem

As agentic workloads move from demos to production scale, durable value moves from components to composition.

By **Stephen Alexander**

AUTHOR'S NOTE

Why this exists: This is a personal thought experiment: my own views, not my employer's, and not a forecast from anyone's company.

I spend my days around public-sector missions and cloud infrastructure, where the default pattern has long been to pull systems, data, and control planes back toward a Region. I came up through enterprise management, observability, automated response, and cloud security, with much of that work in public-sector environments. Over the last eighteen months, I have also been building AI-enabled applications on my own time to see how the pieces actually fit.

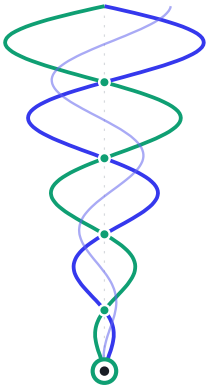
As those applications became more stateful, more tool-driven, and more agentic, I kept finding that the interesting problems, and the opportunities, were not only in the models. They were in the seams between agents, models, data, policy, and the deterministic systems those agents eventually have to touch.

That is what this essay is trying to work through. Operational AI does not look like a bigger chatbot. It looks like a distributed system with cognition inside it.

The framework here is mine. I used an LLM to pressure-test the structure and help produce this as a designed PDF, partly to see how far that workflow has come. Hard-earned judgment, AI-assisted polish.

THE ARGUMENT IN BRIEF

Operational AI is moving from centralized prompt-and-response workloads to distributed, stateful systems. Durable value is shifting from access to models toward the composition of systems: placement, trust, context, coordination, and the control planes that deliver intelligence where latency, policy, and data gravity require it.



Contents

Reading time ~14 min · Digital-first PDF · exhibits rendered as vector diagrams

The first phase of enterprise AI had a simple architecture: send a prompt to a model in a region, get a response back. That simplicity mattered. It made the workload easy to sell, easy to integrate, easy to secure, and easy to reason about. Most applications were single-session, mostly single-turn, and largely stateless between requests. For that workload, the centralized model was not a compromise. It was the right design.

That phase is now giving way to something structurally different. Operational AI, meaning agents that take multi-step actions, hold state, call tools, delegate to other agents, and act on the real world, is not a bigger version of the chat workload. Like every workload that graduates from interesting demo to load-bearing infrastructure, it stops being a model problem and becomes a distributed systems problem. The interesting engineering is no longer inside the model. It is in how the pieces compose.

The interesting engineering is no longer inside the model. It is in how the pieces compose.

01 Beyond the Centralized Prompt

Why the centralized design was right, and why the workload outgrew it.

A single prompt to a frontier model tolerates a round trip to a distant region because a human is on the other end, reading at human speed, and the latency disappears into the cost of the magic. The economics and the ergonomics both work. The centralized pattern dominated the first phase because it was the correct design for that workload.

Operational workflows break the assumption underneath it. An agentic task is a chain: perceive, plan, call a tool, evaluate, re-plan, call again. Each link may be a round trip, and the links are dependent, so step twelve cannot begin until step eleven returns. Latency in a stateful workflow does not average out. It accumulates, and recovery loops accumulate it again. A 400-millisecond hop that was invisible behind a single answer becomes twenty seconds of compounded delay across a fifty-step plan.

Latency is only the first thing that compounds. Because every model call is probabilistic, errors compound too. A modest per-step failure rate, multiplied across dozens of dependent steps, makes naive end-to-end autonomy brittle, which is why robust designs specialize agents, limit each one's action space, and validate between steps rather than trust one long unbroken chain. Both kinds of compounding push the same way: toward decomposition, shorter local loops, and coordination close to the action.

EXHIBIT 1

The compounding tax of agentic workflows

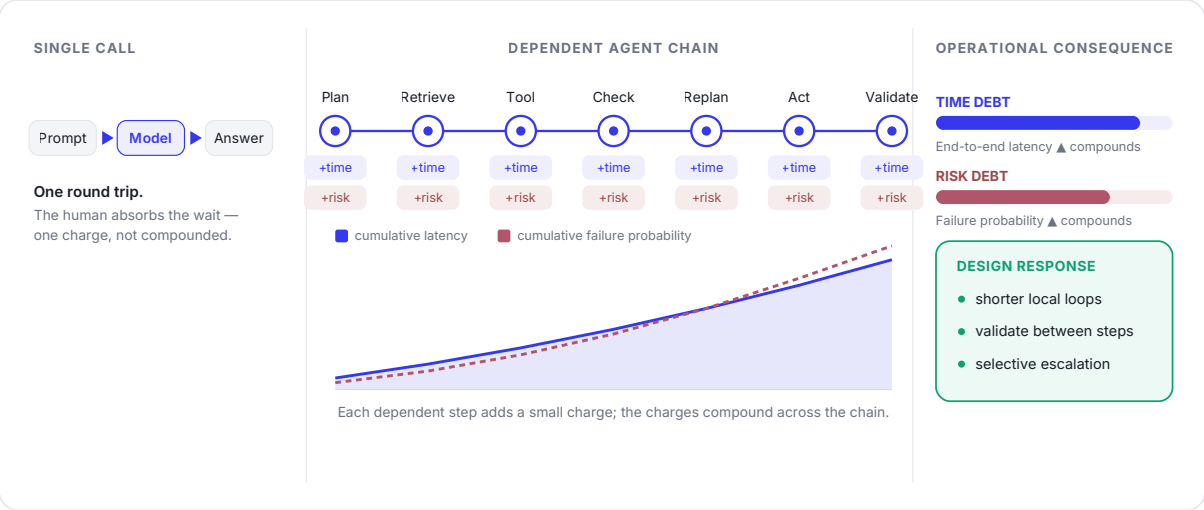


Exhibit 1. Each dependent step adds small charges (time, uncertainty, recovery overhead) that compound into system-level drag.

Data gravity is the second strain, and it pulls against compute. The heavy substrate, meaning the largest models, the training capacity, and the global state, has enormous mass. It wants to stay concentrated in hyperscale regions, and it should, because planning, synthesis, and global consistency are exactly the jobs that benefit from centralization. The flow is not purely one-way. Federated learning, which Google introduced to train a shared model from on-device data without that data ever leaving the device, lets the central tier keep improving from distributed edge experience: model updates travel up while raw data stays in place. Apple has publicly described production deployments of private federated learning across large fleets of consumer devices, so even training is better understood as mostly-central with an upward loop than as a fixed monolith. But the data a workflow needs to act on has its own gravity. It may be bound by residency rules, sit behind an enterprise boundary, or be a live local fact (current inventory, a sensor reading, the user's physical location) that the central model cannot see or infer. A workflow reconciling a centripetal compute substrate with locally bound data is no longer a box with two arrows. It is a system with internal tension, and that tension is the design problem.

The third strain is volume. A single agentic task fans out into many model calls, tool calls, and retrievals, so it generates far more traffic than one chatbot turn, and much of that traffic originates where the user and the data already sit. The shift is visible in the numbers the people building this infrastructure cite: NVIDIA's Jensen Huang argues that reasoning models and agents consume orders of magnitude more tokens than earlier projections assumed. At that volume, hauling everything back to a central region becomes a bandwidth and cost question, not only a latency one, and the economics push the same way the physics do. The analyst forecasts point in the same direction. Gartner expects roughly a third of enterprise applications to embed agentic AI by 2028, up from about one percent in 2024, and IDC projects that most agentic use cases will demand real-time, contextual data. None of this is a precise prediction, but the direction is not in serious dispute. The distribution phase is a planning assumption, not a leap of faith.

EXHIBIT 2

Three strains push operational AI beyond a central-only design

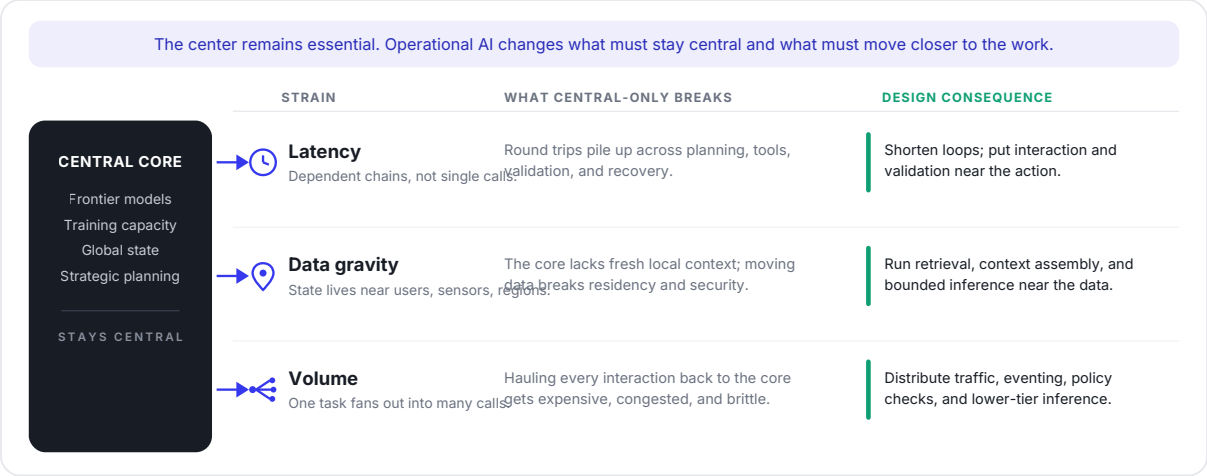


Exhibit 2. The center stays essential for frontier models, training, global state, and planning; operational constraints force the rest outward.

The center still matters. It just no longer describes the whole system. It is one tier in a larger fabric, and the rest of that fabric now has to be designed deliberately rather than assumed away.

The center still matters. It just no longer describes the whole system.

02 The Multi-Tiered Grid

Operational AI decomposes into tiers, fragmenting along three axes at once.

The pressures acting on operational AI appear to favor a particular shape, and it is neither a monolith nor a flat mesh. It is a tiered grid, and it fragments along three axes at once. Designers get into trouble when they plan for one axis and let the other two happen by accident.

Technical tiering: a hierarchy of model sizes. Cognition has a property previous distributed workloads did not: it decomposes by capability. You cannot make a cache node that is eighty percent of a cache node, but you can distill a model that is most of the way as capable at a fraction of the cost and latency. Google's open Gemma family is that made concrete, a size ladder built largely through distillation: a 27-billion-parameter model competitive with models roughly twice its size on a single accelerator, down to variants small enough to run multimodally on a phone with two gigabytes of memory. That property pushes the architecture toward four tiers. A strategic tier of frontier models does the slow, broad work of planning, synthesis, and deciding what to do, and lives most comfortably in the hyperscale core where the heaviest compute and global state already sit. An operational tier of mid-sized models coordinates: it delegates, holds workflow state, enforces policy, and decides what to escalate to a human. An interaction tier of small, fast models runs close to the action, perceiving and deciding inside a tight latency budget. Beneath all of it sits a reflex tier of deterministic control loops that fire in milliseconds and should never be handed to a language model. The bottom tier is the distributed system we already know how to build; the new design work is in the three above it. The major providers are already organizing around pieces of this shape: AWS Bedrock AgentCore targets production agent operation, Azure AI Foundry provides a governed platform for agents, and on-device models like Google's Gemini Nano, already doing real-time work such as on-device scam detection where a cloud round trip would break both latency and privacy, anchor the interaction and reflex tiers. The hierarchy is not a forecast. It is visible in the product catalogs.

EXHIBIT 3

The four-tier model

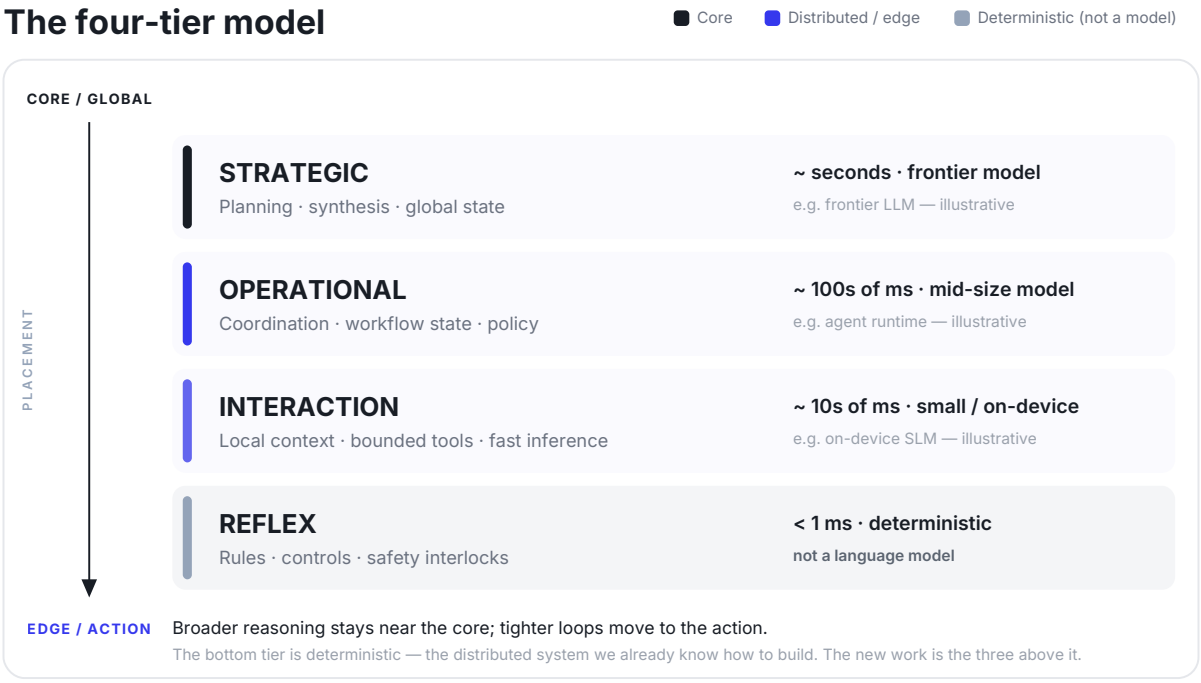


Exhibit 3. Operational AI decomposes into tiers: broad reasoning in the core, governed coordination in the middle, bounded interaction near the action, deterministic reflex at the bottom.

Operational and identity fluidity: from static graphs to fluid dependency graphs. Classical distributed systems are dynamic in operation but usually static in intent. Service A is designed to call service B for a known purpose, even if the instance, route, or replica changes at runtime. Agentic systems make the purpose graph itself fluid. An agent may discover a branch mid-execution, delegate to a new peer, select a new tool, or compose a dependency chain that did not exist when the workflow began. That fluidity creates an operational surface that did not exist when the node was a cached file: every edge in the graph now carries a question of identity and authorization. Who is this agent? On whose behalf is it acting? What is it permitted to do, and is it staying within those bounds? The industry is building the primitives to answer this: first-class agent identities such as Microsoft's Entra Agent ID, neutral agent-identity layers like Okta's that sit over any existing identity provider, interoperability protocols such as A2A (authored by Google, now under the Linux Foundation) and MCP, resting on established foundations like OAuth Token Exchange, SPIFFE workload identity, and NIST 800-207 Zero Trust. The design task is to make identity a property of every edge, not a check at the front door.

EXHIBIT 4

The purpose graph forms at runtime

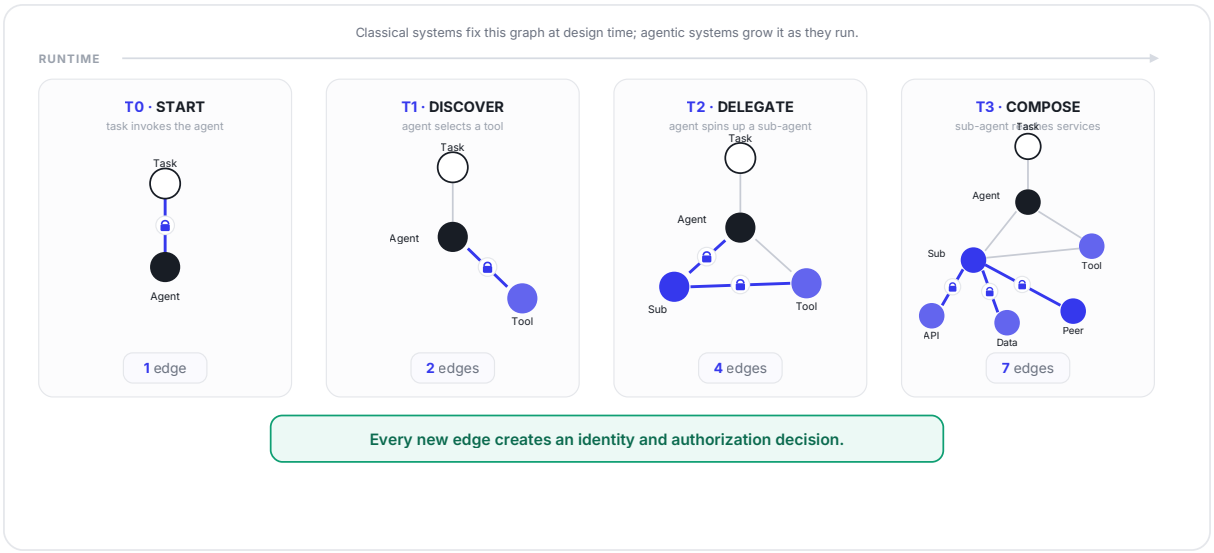


Exhibit 4. Classical systems can be dynamic in routing while static in intent; agentic systems compose the dependency graph at runtime, making identity a property of every newly formed edge.

Physical and compliance constraints: designing for geography. This is the axis pure-software architects most often defer, and it is the least forgiving. Inference has to run somewhere that satisfies both a latency budget and a legal boundary, and those constraints are physical. Data sovereignty requires that some workloads process and remain in-region, which is why localized inference has become its own design pattern rather than a config flag on a central region. Colocation providers like Equinix now market "inferencing pods" in cloud-neutral facilities on exactly this logic, train centrally and infer near the data under consistent governance, and edge networks like Akamai push Blackwell-class inference out to their points of presence for the same reason. Latency budgets dictate how far an interaction tier can sit from the action. And DDIL environments (disconnected, degraded, intermittent, low-bandwidth) remove the option of phoning home at all, so the operational and interaction tiers must plan and act locally, then reconcile when the link returns. Geography stops being a deployment detail and becomes a first-class constraint on where each tier can physically live.

Three axes, decomposing at once, each demanding an explicit decision. This tiered architecture is what results when you plan for all three instead of one.

03 Redefining Scarcity for System Designers

Importance and scarcity are not the same thing.

Here is a principle worth standardizing, because it changes how you value the parts of the system: importance is not the same as scarcity, and scarcity, not importance, is what carries durable architectural value. A component can be essential and still be something five providers supply interchangeably. When that happens it becomes accessible infrastructure, reliable and cheap and undifferentiated, and the design leverage moves elsewhere.

Importance is not the same as scarcity.

For bounded enterprise tasks, portions of the stack are becoming increasingly substitutable, and that is good news for builders. Capable model endpoints increasingly sit behind standard APIs from multiple providers and can be routed or swapped for well-scoped tasks, and the open serving software beneath them is consolidating too: vLLM, the high-throughput engine built on Berkeley's PagedAttention research, has become a de facto standard that Red Hat, NVIDIA, and others build on rather than around. Model size is getting cheaper to move as well, since quantization now routinely compresses a seventy-billion-parameter model from roughly 140 gigabytes to around 40 with little measurable quality loss. Even raw edge capacity is beginning to normalize as distributed GPU footprints become more available, and high-density inference turns out to be economical at a focused set of locations rather than at every point of presence, which makes "where do the GPUs go" a solvable placement question rather than a moat.

Be precise about the limit of this claim. Frontier models are not interchangeable, the dominant accelerator platform is not a commodity, and running true distributed inference at scale is hard, specialized work. What is becoming substitutable is the bounded, well-specified middle. That is the part to treat as infrastructure you build on, not a differentiator you build around.

The scarce resource is further up the stack and harder to name: usable intelligence under operational constraints. Not access to a model, but the delivery of the right model's judgment to the right place, within a latency budget, inside a compliance boundary, governed by the right policy, composed with the steps around it, and reliable when a link fails. Judgment is only half of what has to arrive at the point of decision. The other half is context: fresh, governed, in-the-moment state about the world the agent is acting on, because a model reasoning over a stale snapshot fails no matter how capable it is. Abundant intelligence is becoming an input, and so is abundant data. Turning the two into a dependable operational result is the scarce engineering act, and it is scarce because it does not modularize cleanly. It lives in the connections.

EXHIBIT 5

Durable value migrates from components to seams

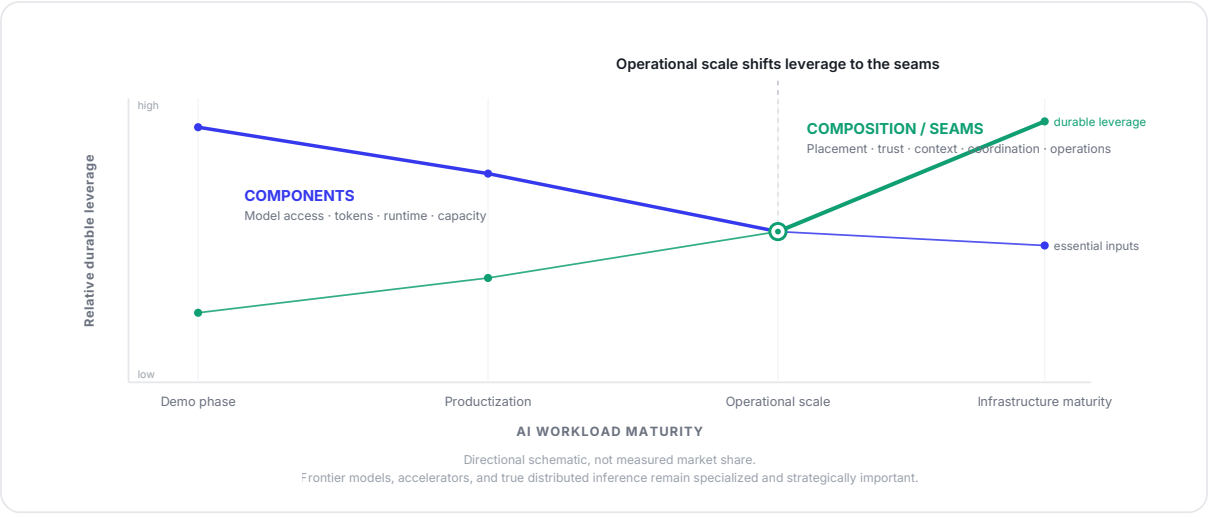


Exhibit 5. As model access, token throughput, and generic runtime become more available they remain essential but less differentiating; durable leverage shifts toward the seams.

04 Designing the Seams

Four boundaries where the durable value now lives.

When components commoditize, value relocates to the boundaries between them, the seams where two operational constraints meet and pull in different directions. A seam is easy to describe and hard to build, because each side is a clean module while the join across it carries the friction. Four seams matter most.

The connective tissue: events and requests, not events or requests. The tiers have to talk, and how they talk is a design decision the block diagram rarely shows. Two patterns coexist, and mature systems use both on purpose. Request/response is right when an agent needs an answer now: a tool call, a synchronous lookup, a verification check in the path. It is simple, debuggable, and what most deployed systems run on today, which is not a transitional state to be designed away. But request/response alone forces every consumer to poll for change and couples it to every producer, which does not scale to a fluid graph of agents that must act the moment the world shifts. That is where an event-driven fabric earns its place. Agents and the systems they depend on publish and subscribe to events ("inventory dropped," "invoice posted," "sensor tripped"), so coordination happens through loosely coupled messages instead of brittle point-to-point calls. Event-driven design gives the grid real-time context delivered as state changes rather than fetched on a guess, loose coupling so tiers evolve and agents swap in and out, backpressure under load, and a replayable event log that doubles as the audit trail for autonomous actions. The skill is not picking a side. It is knowing which interactions must be synchronous requests and which should be asynchronous events, and composing the two so the system stays both responsive and decoupled.

EXHIBIT 6

Requests answer now. Events tell you when something changed.

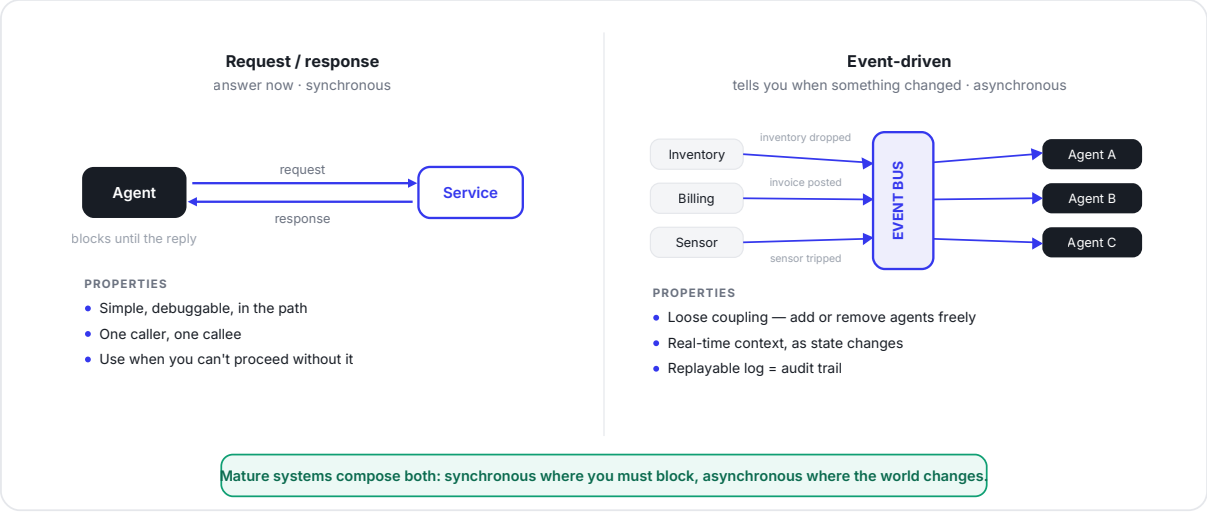


Exhibit 6. Mature systems compose both patterns: synchronous requests where an agent must block, asynchronous events where the world changes.

That is the first seam. The second is harder, because it is usually mistaken for a smaller problem.

The trust plane: continuous, in-path governance. It is tempting to treat agent trust as an identity problem: verify who the agent is and let it through. A signed request proves origin. It does not prove safe behavior. Identity is necessary, but it is one question of several, and treating it as the whole is the most common way this seam gets designed badly. A useful frame, drawn from the Cloud Security Alliance's emerging agentic Zero-Trust work, asks five things of every agent, continuously: who is it (identity), what is it doing (behavior), what is it consuming and producing (data governance), where can it go (segmentation), and what happens if it goes rogue (incident response). Trust is earned through observed behavior and re-checked continuously, not granted once at the door, because a verified agent can still be compromised, over-permissioned, or steered by an injected instruction it picked up mid-task. The identity vendors are arriving at the same place from their own direction: Okta, framing agents as identities that operate at machine speed, argues that grant-and-forget permissioning has to give way to short-lived credentials and a re-check of every action an agent takes.

A signed request proves origin. It does not prove safe behavior.

The architectural move is the same, applied to all five dimensions rather than identity alone. Issuance and authoritative policy belong to a central fabric, the enterprise directories and agent-identity platforms that serve as systems of record. Enforcement belongs in the transit path, at low latency, before an action reaches a core system, because fluid delegation chains generate far too many decisions to route home one at a time. The important point is the split: separate the policy decision point from the policy enforcement point. Open standards are starting to make the identity layer composable. Web Bot Auth, led by Cloudflare and Google, has an agent sign each request with HTTP Message Signatures (RFC 9421) so an intermediary can validate it against a published

key without calling back to the issuer; Visa's Trusted Agent Protocol uses the same signatures, verified at a CDN proxy, to prove an agent is both legitimate and authorized for a specific purchase. But the enforcement plane has to do more than check a signature. It watches behavior for anomalies, inspects inputs and outputs for injection and data leakage, holds each agent inside its segment, and can trip a circuit breaker when something crosses a line. Designing the trust plane means designing that whole continuous loop, in-path, as one distributed enforcement layer over a central system of record.

EXHIBIT 7

The trust plane: decide centrally, enforce in the path

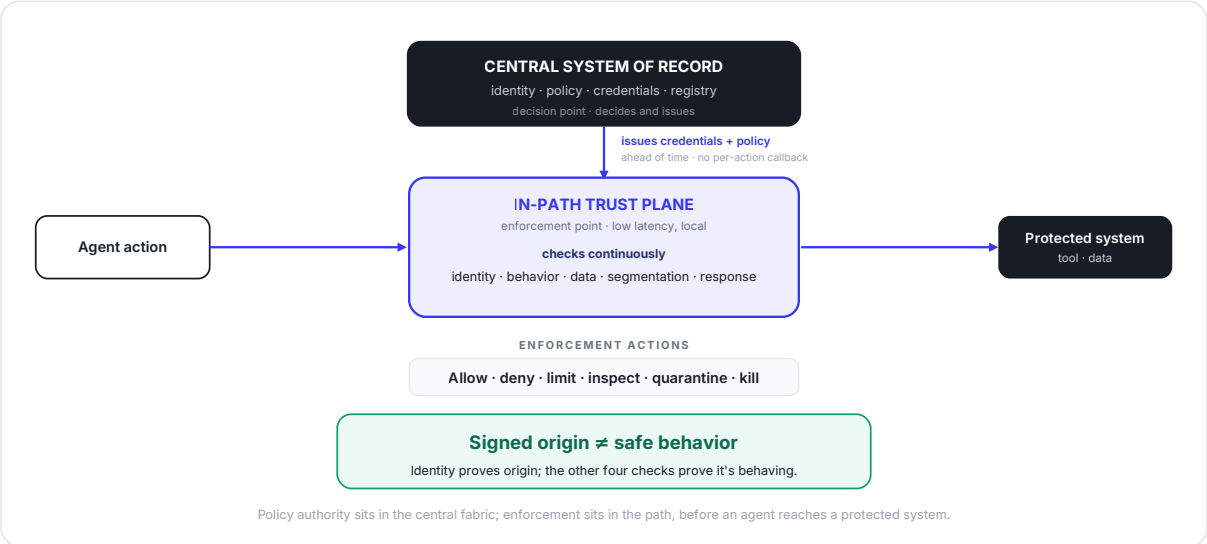


Exhibit 7. Policy authority sits in the central fabric; enforcement sits in the path, before an agent reaches a protected system.

Workload placement, and the Day-2 fleet it creates. This seam is a standing tension, not a one-time decision. Data gravity argues for keeping the workload where the data lives: in-region for residency, behind the boundary for security, near the source for freshness. Model planning argues for running inference where capability and performance are best, often the central substrate. Honor only one side and the design fails predictably. Pin everything to the data and you starve the workflow of the strategic tier's judgment; pin everything to the core and you violate residency or blow the latency budget. The durable pattern routes by constraint: strategic planning in the core, operational coordination wherever the link topology allows, interaction-tier inference local to the action, decided per workload and per jurisdiction rather than globally.

Placement is also not a decision you make once. This sounds like deployment. It is really operations. Even inside a single cluster, serving a model well is already a systems discipline rather than a modeling one: the throughput that makes inference affordable comes from how requests are scheduled and how memory is managed. Continuous batching, introduced in the Orca serving system, and PagedAttention, the memory technique behind vLLM that raised serving

A model placed in fifty locations is a fleet you now have to operate.

throughput two to four times, each improved inference without changing the model at all; and splitting the compute-bound prompt phase from the memory-bound generation phase onto separate hardware, as Microsoft and the University of Washington showed with Splitwise, cut serving cost by roughly a fifth again. None of that is model work. It is distributed-systems work, and it is where the efficiency lives. Now multiply it by geography. A model placed in fifty locations is a fleet you operate: push updates without taking the system down, roll back a bad model version atomically, keep configuration from drifting across sites, and maintain coherent observability over inference happening in hundreds of places at once. This is where distributed AI actually gets hard, and it scales with geography. Operating a few regions is a different problem from operating a fleet that spans all of Europe or all of North America, where update windows, residency rules, connectivity, and failure domains vary site by site. The teams that win this seam treat it as a managed lifecycle, a central control plane over distributed execution, not a one-time deployment.

EXHIBIT 8

A model placed in fifty locations is a fleet

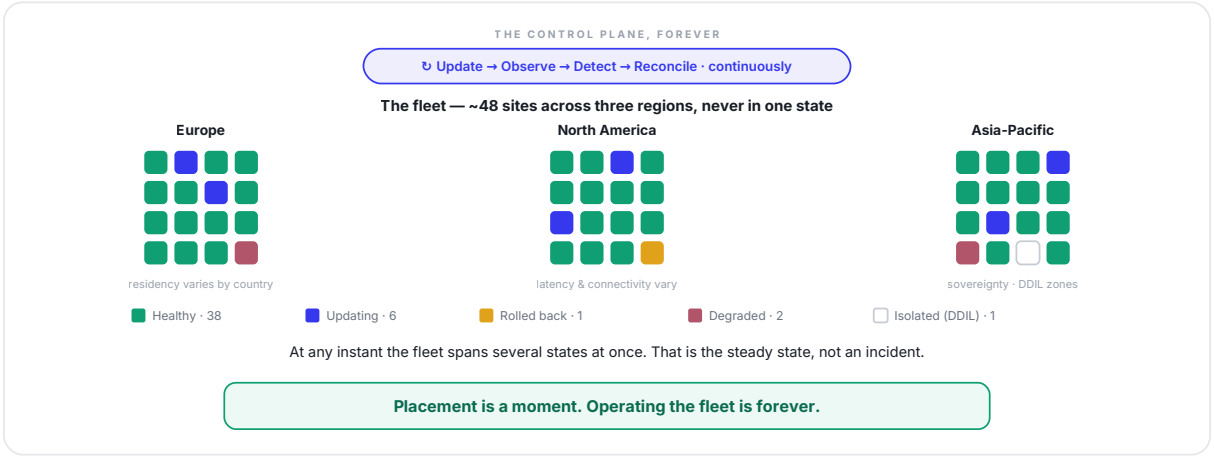


Exhibit 8. Placement is a one-time decision; operating the fleet (never in one state) is the standing job.

Architectural neutrality versus deep integration. The last seam is a genuine trade-off with no universally correct answer. A deeply integrated single-provider ecosystem offers compounding advantages: co-designed silicon and runtime, low operational friction, unified identity and observability, and a coherent end-to-end governance story. The major clouds have invested heavily here because integration delivers reliability and performance loose coupling struggles to match, and for many workloads that is the right call. A modular, cross-provider abstraction layer offers a different set: portability, freedom to place each tier with the best-suited provider, resilience against single-provider outages, and a neutral enforcement plane that can sit in front of any core, a pattern edge networks such as Cloudflare and Akamai make viable across clouds. Most real systems will run a hybrid: deep integration where co-design pays off and the workload is stable, modular abstraction at the seams where portability, compliance, and resilience matter more than peak optimization. Designing this seam well means knowing, tier by tier, which trade you are making and why.

EXHIBIT 9

Neutrality vs deep integration: a spectrum, not a binary

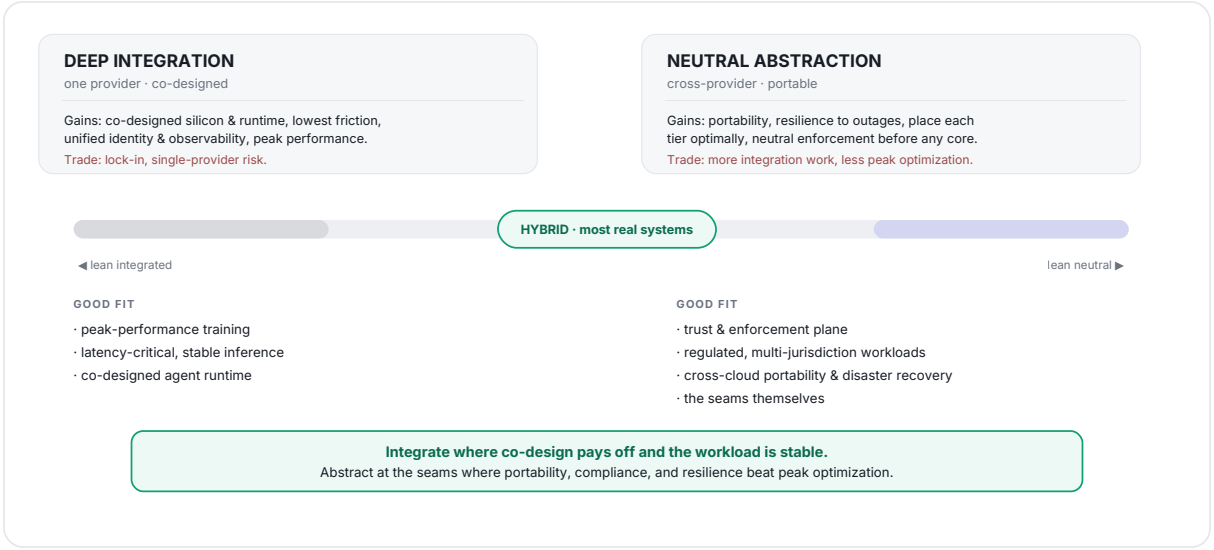


Exhibit 9. Integrate where co-design pays off and the workload is stable; abstract at the seams where portability, compliance, and resilience beat peak optimization.

05 How the Layers Interlock

Three planes compose into one operating system.

The picture that emerges is not a contest won by a single layer. It is a fabric that only works when the layers compose, and each major movement in the industry maps to a tier the others depend on. The hyperscale core (AWS, Microsoft, Google, and their peers) provides the permanent foundation: the heaviest compute, the frontier models, the global state, and the strategic planning tier the whole grid reasons from. That foundation is not being displaced by distribution. It is being extended by it, the way origin infrastructure was extended, not replaced, by content delivery. The enterprise identity fabric (Entra, Okta, the Zero Trust and security-for-AI work from vendors like Zscaler) supplies the authoritative answer to who may act and on whose behalf, which the fluid dependency graph relies on at every edge. The distributed enforcement and inference planes take more than one physical form: neutral edge networks and points of presence, on-premises converged appliances on the branch, store, or factory floor, and the agent-trust layers that run across them. Together they carry verification, policy, and local cognition out to where data gravity, traffic volume, and latency budgets require them, in front of any core and across any boundary.

The future state is a handshake between these three: strategic cognition concentrated in the core, identity issued and governed by the fabric, and verification and interaction-tier inference enforced at the distributed plane, composed into a single coherent distributed system. No tier is sufficient alone. The core cannot be everywhere the action is. The identity fabric cannot enforce in every transit path at low latency. The distributed plane cannot host the frontier substrate or hold global state. They are complementary by construction, and the systems that work will be the ones designed to let them interlock.

EXHIBIT 10

The interlock

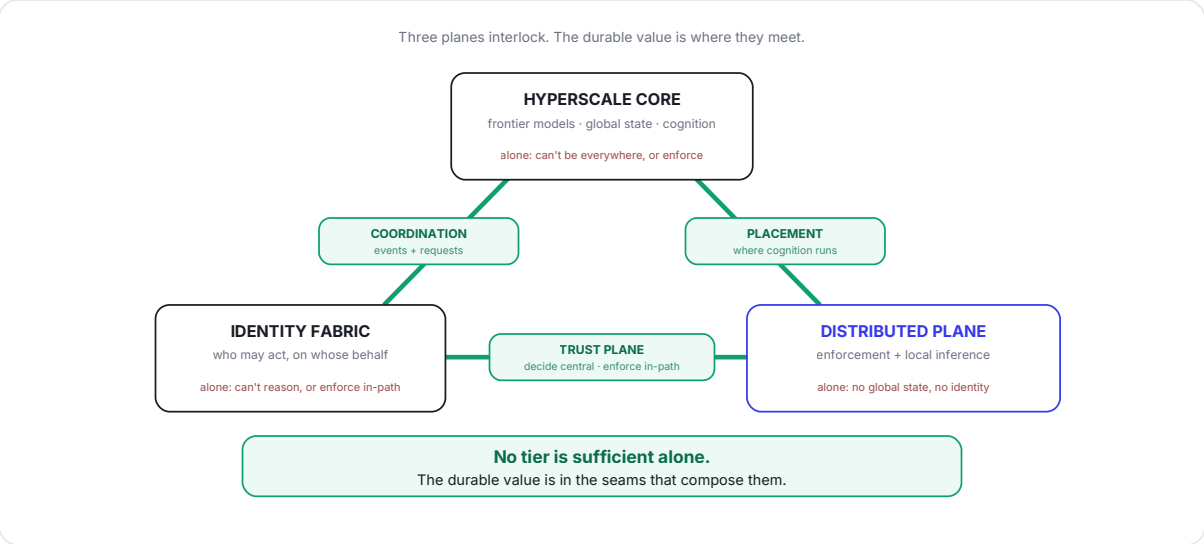


Exhibit 10. Core cognition, governed identity, and distributed enforcement interlock into one system, and the durable value is in the seams that compose them.

This complementarity should not be mistaken for equal economics. Every tier is necessary, but not every tier captures the same value. As model access, token throughput, basic inference capacity, and generic edge runtime become more substitutable, they become inputs. The durable leverage sits with the systems that decide placement, preserve context, enforce trust, and keep the graph coherent across providers and locations. The winning platforms will not merely supply a component. They will make the heterogeneous system operable.

That is the shift worth internalizing. The first phase rewarded access to the smartest component. The operational phase rewards the design of the composition: the placement, the trust plane, the handshake across boundaries. Components will keep getting better and cheaper, which is exactly why they stop being where the leverage is. The scarce, durable, rewarding work is in the seams, and the architects who treat the seams as the product are the ones building the systems the next decade will actually run on.

SOURCES & INFLUENCES

A partial list of what shaped this thinking: standards, primary research, vendor documentation, and analyst and practitioner work I read while writing it. The synthesis is mine; so are any misreadings.

STANDARDS & PRIMITIVES

NIST SP 800-207, Zero Trust Architecture
NIST, 2020. The zero-trust baseline the agent trust plane extends to every edge.

HTTP Message Signatures (RFC 9421) & Web Bot Auth
IETF; Cloudflare and Google, 2025. Signed-request verification at the edge; the shared mechanism behind Web Bot Auth and Visa's protocol.

OAuth 2.0 Token Exchange (RFC 8693)
IETF, 2020. Delegation primitive (act / may_act) for an agent acting on a user's behalf.

Agent2Agent (A2A) & Model Context Protocol
Linux Foundation; Anthropic, 2025. Interoperability for how agents discover and call each other and their tools.

SERVING & INFERENCE SYSTEMS

Efficient Memory Management for LLM Serving with PagedAttention (vLLM)
Kwon et al., UC Berkeley, SOSP 2023. The paging technique behind vLLM; two to four times serving throughput. The open serving standard the essay names.

Orca: A Distributed Serving System for Generative Models
Yu et al., OSDI 2022. Origin of continuous / in-flight batching, the scheduling primitive every serving stack now uses.

Splitwise: Efficient Generative LLM Inference Using Phase Splitting
Patel et al., UW and Microsoft, ISCA 2024. Routing the prompt and generation phases to different hardware; about a fifth lower serving cost.

Mastering LLM Techniques: Inference Optimization
NVIDIA, 2023. Why inference is memory-bound; the KV-cache math under the serving-layer argument.

Get Started with AI Inference
Red Hat, 2026. Quantization compressing a 70B model from roughly 140GB to around 40GB; the substitutable middle.

MODEL TIERING & ON-DEVICE

Gemma and Gemma 4 open models
Google DeepMind, 2024-2026. Distillation plus a size ladder from microcontroller to single accelerator; the four-tier model made concrete.

Gemini Nano on AICore
Google, 2023-2026. On-device foundation model doing real-time work, such as scam detection, a cloud round trip would break.

Switch Transformers: Scaling to Trillion Parameter Models
Fedus, Zoph, Shazeer, Google, JMLR 2022. Sparsity decoupling parameter count from per-token compute.

AGENT IDENTITY & TRUST

The Agentic Trust Framework
Cloud Security Alliance, 2026. The five continuous questions (identity, behavior, data, segmentation, incident response) the trust plane asks.

The AI Agent Era & Okta for AI Agents
Okta, 2026. Agents at machine speed; grant-and-forget gives way to short-lived credentials and re-checking every action.

Trusted Agent Protocol
Visa, 2026. Cryptographic agent identity for commerce, verified at a CDN proxy via RFC 9421.

Bedrock AgentCore; Entra Agent ID & Azure AI Foundry
AWS; Microsoft, 2025-2026. Production agent runtimes and first-class agent identities; the catalog the tiering is visible in.

EDGE, PLACEMENT & SOVEREIGNTY

Architecting the Agentic Web; Akamai Inference Cloud

Akamai, 2025. Inference as the dominant workload; a three-layer edge architecture mirroring the placement and trust seams.

The Best Place on Region: Earth for Inference

Cloudflare, 2023. The network as the 'goldilocks' of inference; the device, edge, and cloud placement spectrum.

Optimized AI Inferencing at the Edge; Sovereign Control with Global Reach

Equinix, 2025-2026. 'Inferencing pods' in cloud-neutral colocation; centralized policy with distributed execution.

Moving AI to the Edge: Benefits, Challenges and Solutions

Red Hat, 2025. Why edge inference simplifies latency, residency, and sovereignty.

FEDERATED LEARNING & PRIVACY

Communication-Efficient Learning of Deep Networks from Decentralized Data

McMahan et al., Google, AISTATS 2017. The paper that coined federated learning: updates travel up, raw data stays on the device.

Private Federated Learning in Real World Application

Apple, 2025. Production private FL: the central model improves from edge experience; improved models push back down.

ANALYST FORECASTS & MULTI-AGENT PRACTICE

Multi-Agent Systems Need Real-Time Context and Event-Driven Architecture

Solace, 2026 (carries Gartner and IDC). Analyst forecasts (Gartner roughly 1% to 33% by 2028; IDC real-time data) and the events-versus-requests seam.

How We Built Our Multi-Agent Research System

Anthropic Engineering, 2025. A practitioner account of decomposing agents and validating between steps.