

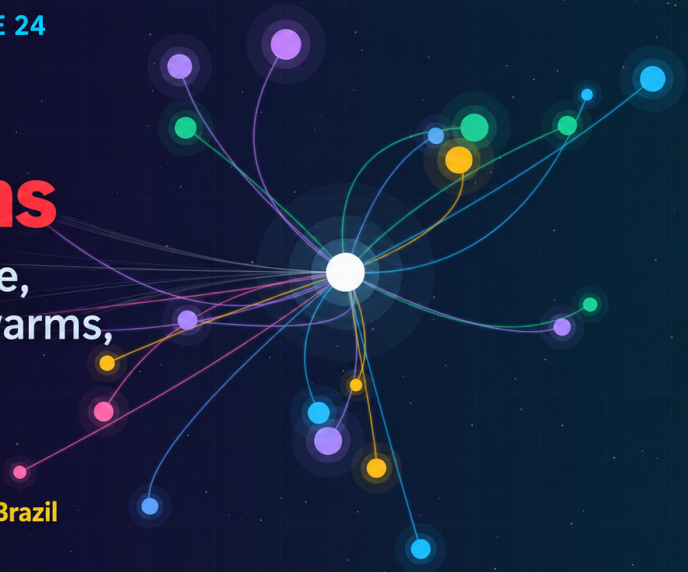
SGAEIA RESEARCH SERIES — ARTICLE 24

LLM Routers for AI Systems

Architecture, Governance,
Multi-Agent Systems, Swarms,
and Security

Aridio Silva — Independent Researcher, Brazil
ORCID 0009-0008-2411-6995

© Aridio Silva @aridiosilva — October 2026



LLM Routers for AI Systems

Architecture, governance, multi-agent systems, swarms, and security

SGAEIA Research Series — Article 24

Aridio Silva — Independent Researcher, Brazil

ORCID 0009-0008-2411-6995

29 October 2026 · CC BY 4.0

DOI: 10.5281/zenodo.23763978

LLM Routers for AI Systems: Architecture, Governance, Multi-Agent Systems, Swarms, and Security

From cost optimization to governed decisions about models, data, and trajectories

SGAEIA Research Series — Article 24

Aridio Silva — Independent Researcher, Brazil

Creator of SGAEIA — Secure Governed Autonomous Edge Intelligence Architecture

ORCID: 0009-0008-2411-6995

29 October 2026

© 2026 Aridio Silva. Text and original illustrations licensed under CC BY 4.0.

Abstract

Language model routing distributes requests across models, providers, and configurations with different capabilities, costs, and response times. Its usefulness, however, depends on two distinct decisions: establishing which alternatives are admissible for the task and selecting the most suitable configuration from that set. This article presents a technical review and argues that, in agentic and multi-agent systems, routing forms part of a decision boundary concerning authority, data, resources, and policy continuity. The analysis distinguishes routers, gateways, and control planes; compares rules, classifiers, similarity, cascades, and model-driven selection; and examines what benchmarks establish about quality, cost, latency, and robustness. Its analytical contribution is an SGAEIA-related framework for assessing governed routing, applied to model substitution, budgets, and multi-agent execution. The analysis explains why savings per call do not equal savings per task, availability does not establish admissible substitution, and average accuracy does not demonstrate trajectory integrity. Adoption should depend on observed gains against simple baselines, accompanied by sufficient evidence about the boundaries and consequences of decisions.

Keywords: AI; LLM routing; AI gateway; multi-agent systems; security; governance; evaluation; SGAEIA.

Contents

1. The problem extends beyond choosing the cheapest model
2. Routers, gateways, and control planes
3. How the evidence was organized
4. Routing strategies and their trade-offs
5. Quality, cost, and latency need a common unit of analysis
6. Integration and infrastructure selection
7. What benchmarks demonstrate
8. Cross-cutting comparison of strategies
9. Identity, authority, and governed substitution
10. Agents, multi-agent systems, and swarms
11. Governance, security, and evidence
12. An SGAEIA framework for assessing governed routing
13. Evolution and maturity should be assessed by dimension
14. When to adopt routing and when to retain a simpler solution
15. Discussion and analytical limits
16. Conclusion

1. The problem extends beyond choosing the cheapest model

An application using language models must decide where to invest inference capacity. A model suitable for extracting fields from a document may be insufficient to verify a complex solution; a model capable of that verification may have costs or response times unsuitable for repetitive work. Routing research explores this heterogeneity by distributing queries across alternatives rather than maintaining one choice for all traffic. The economic opportunity arises from model complementarity and task diversity, but depends on being able to anticipate or recognize where each alternative performs adequately [1][5][6].

The problem changes when the generated response participates in an action. In a question-answering assistant, an unsuitable selection may produce an unsatisfactory answer; in an agent that queries systems, writes code, or delegates work, the same decision may change the data transmitted, tools used, and subsequent effects. Model selection must therefore account for task state, the person or service authorizing execution, and the boundaries accompanying that authorization. SGAEIA provides the architectural perspective adopted here: the capability to produce a response and the authority to carry out an action are different properties [27].

The central thesis is that routing in agentic systems extends beyond inference optimization and forms part of a critical boundary for decisions, governance, and evidence production. This does not turn every router into a security system or make a learned algorithm mandatory. Shafran and colleagues already place routers within LLM control planes and demonstrate attacks targeting model selection [9]. The step developed here is to organize the implications of that understanding into comparable criteria, engineering decisions, and verifiable evidence.

Six questions guide the analysis: when routing saves money while preserving quality; when a strategy outperforms simple alternatives; what value a gateway provides independently of its algorithm; what risks follow from concentrating credentials and policies; what changes with agents and swarms; and what evidence supports governed selection, substitution, and revocation. The article contributes a synthesis of these

questions within a common decision framework, presented and applied in Section 12. Its value is analytical: examining a configuration, identifying its assumptions, and explaining a decision to adopt or reject it. The effectiveness of a particular implementation requires evaluation in its intended environment.

2. Routers, gateways, and control planes

An LLM router is a selection function: it receives information about a task and chooses a model, deployment, or execution configuration. The function may use local rules, a classifier, a similarity calculation, or a policy informed by intermediate results. It does not require an HTTP server or public API; selection can occur within the application process. The sequential-decision formulation used by LLMRouter helps explain cases involving history, personalization, and multiple steps [8].

An AI gateway mediates access to models. It may adapt protocols, check credentials, enforce limits, forward requests, and produce telemetry, while also hosting a routing function. These responsibilities remain useful even if every query goes to the same model. LiteLLM's virtual-key documentation, for example, describes control of model access and spend tracking; these are mediation capabilities, not evidence that its selector achieves the best cost-quality trade-off [10].

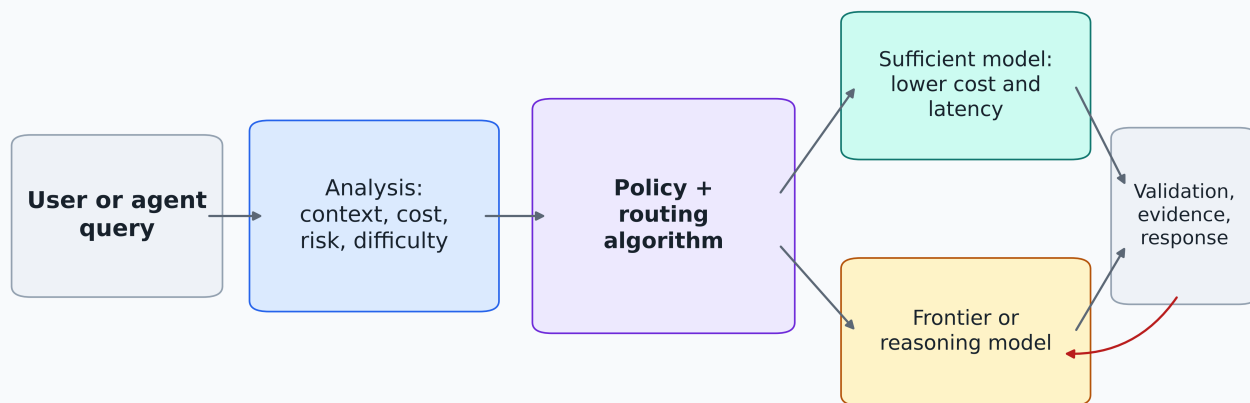
The control plane administers decisions governing the environment: policies, configurations, permissions, versions, and substitution criteria. The data plane carries the corresponding traffic. A gateway helps enforce these decisions, but its administrative components, inference path, and evidence storage need not share the same privilege domain. The distinction has practical consequences: compromising a client application should not permit editing gateway policies, and observing a response should not grant access to the credentials enabling it [16].

Table 1

Element	Main question	Responsibility	Evaluation evidence
Router	Which alternative should handle this task?	Select a model, configuration, or next step	Selection quality, decision cost, and generalization
Gateway	How should authorized execution be mediated?	Adapt access, enforce controls, and observe traffic	Isolation, availability, limits, and records
Control plane	Who may define and change these decisions?	Administer policies, permissions, and lifecycle	Authorization of changes, versions, and revocability

Figure 1. Governed routing decision

Policy and classification precede model selection; validation may escalate or block the response.



Goal: select sufficient capability without violating policy, authority, or evidence requirements.

Source: Ong et al. (2025); Jiangeng et al. (2026); SGAEIA interpretation

© 2026 Aridio Silva | Project SGAEIA | CC BY 4.0

Figure 1. The flow distinguishes task analysis, routing decisions, and result validation. Selection should consider applicable constraints before optimizing performance. This is a conceptual diagram: the presence of these blocks does not establish correct policy enforcement in a deployment.

Conflating these layers creates two evaluation errors. The first attributes infrastructure benefits, such as key inventories or centralized auditing, to the selection algorithm. The second assumes that a feature-rich gateway has already solved agent quality, authorization, and security. A coherent architecture defines each component's responsibility and verifies their interactions, including when a component is unavailable or operates with incomplete information.

3. How the evidence was organized

The analysis is a narrative technical review guided by the six introductory questions. The starting corpus was the Article 24 material, supplemented by original studies, publication records, official documentation, risk-management references, and technical incident reports. Selection prioritizes sources that explain a mechanism, provide comparable evaluation, or document a relevant operating condition. The reference date is 29 October 2026; it does not represent an exhaustive search of all routing research.

Four types of statement are distinguished. Method and benchmark results are attributed to their authors and evaluated conditions. Product features are described as capabilities documented by suppliers, while legal and risk-management requirements remain linked to their official sources. Cross-cutting comparisons, the governance matrix, and decision scenarios are analytical syntheses developed in this article. This separation distinguishes third-party observations from reasoned implications and proposed assessment approaches.

An identical savings percentage can represent fewer calls to an expensive model, lower API charges, or reduced computational cost. Comparisons therefore retain the unit, baseline, and generalization boundary rather than ranking studies by their largest percentage. They also distinguish studies proposing a method from work re-evaluating several methods under a common protocol. Neither category removes the need to examine data, model pools, evaluators, and potential commercial interests.

The article does not present published results as its own experiments. The calculations in Section 5 are hypothetical scenarios with explicit inputs and reproducible arithmetic. Its contribution is demonstrated through a consistent comparison and application of the governance framework to decisions for which isolated criteria yield insufficient conclusions. Empirical assessment of risk reduction, control performance, or production security belongs to a subsequent stage with its own protocol and results.

4. Routing strategies and their trade-offs

4.1 Selecting before generation

Predictive routing chooses a destination using information available before the response: query content, domain, length, identity, history, or a vector representation. Explicit rules can send documents of a particular class to an approved deployment or reserve a model for tasks requiring a specific capability. Their transparency helps explain decisions and examine exceptions; their limitation is representing complex boundaries without proliferating rules. A keyword rule, for example, may confuse a programming question with a request that merely mentions a programming language.

Classifiers and preference models replace some rules with a learned estimate. RouteLLM uses preference data to estimate when the more capable model is worthwhile and investigates different selection architectures [1]. Hybrid LLM likewise adjusts routing between smaller and larger alternatives according to a desired quality level [4]. The advantage is selecting without generating all candidate responses; the dependency is representative supervision and recalibration when models, tasks, or quality criteria change.

Similarity routing compares the current task with previous examples and uses their history to recommend a model. This exploits domain structure, but proximity in embedding space does not establish equal difficulty or consequences. A simple question and a critical question can share vocabulary while requiring different treatment. Similarity may inform expected performance; authorization to process the data must come from separate criteria.

4.2 Deciding after observing a response

A cascade consults a model before deciding whether to stop or escalate. FrugalGPT explores combinations of models and acceptance criteria to reduce inference cost [2]. The strategy is attractive when the first alternative solves a substantial share of tasks and its result can be assessed reliably enough. When almost every task escalates, the system pays for the initial model and its successor, plus evaluation and accumulated latency.

AutoMix combines an initial answer, self-verification, and decision-making under uncertainty, recognizing that intermediate assessment can be noisy [3]. The central issue extends beyond the smaller model's capability to whether the verifier recognizes its errors. Valid JSON may contain incorrect data, and a confident response may lack support. Format validation, content assessment, and action authorization are different checks, even when they share infrastructure.

An LLM judge offers flexible evaluation but adds cost, dependency, and potentially correlated errors. Changing the judge's supplier may increase diversity without establishing effective independence: instructions, references, and criteria may remain shared. For important tasks, the system must measure disagreements, false acceptances, and false rejections alongside average scores. Evaluator identity should be part of the decision record, especially when its score determines a subsequent action.

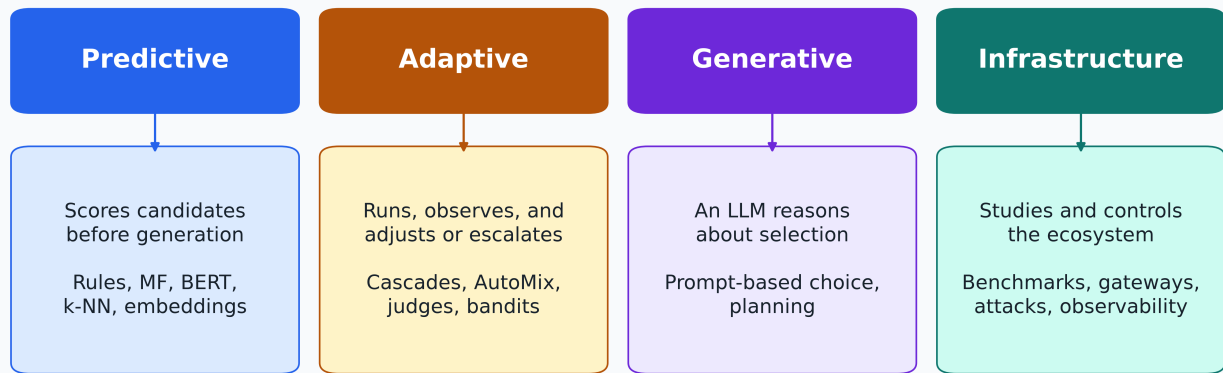
4.3 Adapting policies and using models to select models

Adaptive strategies may update decisions from observed results, while generative strategies use an LLM to interpret, decompose, or plan selection itself. Bandits and related approaches face an additional trade-off: exploring alternatives to learn may incur costs and inferior responses during exploration. A generative selector adds inference before the main inference and can make decisions harder to explain. These methods must justify their additional costs and risks against a simple alternative.

Jiangeng and colleagues organize the field into predictive, adaptive, and generative operations and distinguish infrastructure work that evaluates or supports routing [7]. This complements the distinction between decisions made before and after generation. A gateway may combine mandatory rules, a predictive classifier, and adaptive escalation without those mechanisms competing in every respect. A taxonomy should identify the decision each mechanism makes and the information it uses.

Figure 2. Updated taxonomy of routing strategies

Four paradigms defined by the algorithmic operation, following the 2026 systematic survey.



Selection → adaptation → reasoning → evaluation and control

Source: Jiangeng et al., Findings of EMNLP (2026)

© 2026 Aridio Silva | Project SGAEIA | CC BY 4.0

Figure 2. The taxonomy relates advance selection, cascades, and intermediate evaluation. It helps locate additional calls and supervision dependencies. Categories can coexist within an application; their presence establishes no universal ranking of quality.

Table 2 - panel 1/2

Strategy	Generation calls	Training or supervision	Main advantage
Rules	Usually one	Explicit policies and criteria	Transparency and low selection cost
Classifier or preferences	Usually one, plus any embeddings	Labeled data and calibration	Selection without generating alternative answers
Similarity	Usually one, plus vector representation	Evaluated history	Incremental use of examples
Cascade	One or more	Acceptance criterion	Escalation only when needed
Self-verification or judge	Two or more operations, depending on design	Examples, rubric, or evaluator	Assessment conditioned on the result
Adaptive or generative	Variable	Observed feedback or selection instructions	Adaptation to state and context

Table 2 - panel 2/2

Strategy	Main dependency
Rules	Rule coverage and exception handling
Classifier or preferences	Representative data
Similarity	Neighbor coverage and relevance
Cascade	Validation reliability and cost
Self-verification or judge	Correlated errors and additional cost
Adaptive or generative	Exploration cost, stability, and explanation

5. Quality, cost, and latency need a common unit of analysis

5.1 The relevant cost is the cost of a completed task

The price of a call does not describe system cost. The total includes input and output tokens, retries, embeddings, verifiers, infrastructure, and operational work. In agentic workflows, it also depends on the number of calls needed to complete a task and how many trajectories must be repeated. A model with cheap tokens may make a task more expensive if it requires more corrections, generates unnecessarily long responses, or increases tool calls.

A useful decomposition, proposed here for operational assessment, is:

$$C_{\text{task}} = C_{\text{selection}} + C_{\text{generation}} + C_{\text{validation}} + C_{\text{retry}} + C_{\text{operation}}$$

© 2026 Aridio Silva | Project SGAEIA | CC BY 4.0

Equation 1

This expression organizes the components to measure; it does not assume they are constant or independent. Retry cost depends on answer quality and recovery policy, while operating cost may contain a significant fixed component. Alternatives must be compared using the same definition of task completion and the same acceptance criteria. Otherwise, apparent savings may simply reflect accepting inferior responses or stopping tasks prematurely.

Consider a hypothetical scenario in which a compact-model call costs US\$0.002 and a more capable-model call costs US\$0.020, with input and output lengths already incorporated into these prices. A selector sends 80% of queries to the former and 20% to the latter, adding US\$0.0004 per query for selection. Average variable cost is US\$0.006, 70% below sending all queries to the more expensive model. The calculation assumes unchanged traffic distribution, per-call costs, and required quality.

$$C_{\text{routed}} = 0.8(0.002) + 0.2(0.020) + 0.0004 = 0.006$$

© 2026 Aridio Silva | Project SGAEIA | CC BY 4.0

Equation 2

If operating the router adds US\$500 monthly, the variable reduction does not translate fully into total savings. At 10,000 queries, spending rises from US\$200 to US\$560; at 50,000, it falls from US\$1,000 to US\$800. Break-even is approximately 35,715 monthly queries under these assumptions. The example demonstrates why volume and fixed cost must accompany a savings percentage presented to decision-makers.

Table 3

Hypothetical monthly volume	Always the more expensive model	Routing, including US\$500 fixed cost	Difference from baseline
10,000 queries	US\$200	US\$560	US\$360 increase
50,000 queries	US\$1,000	US\$800	US\$200 saving
100,000 queries	US\$2,000	US\$1,100	US\$900 saving

$$N^* = \frac{F}{C_{\text{baseline}} - C_{\text{routed}}}$$

© 2026 Aridio Silva | Project SGAEIA | CC BY 4.0

Equation 3

This break-even expression applies only when routed variable cost is lower than the baseline and the fixed component can be represented by F. It changes if the gateway serves other objectives, such as access management, whose costs would exist without dynamic selection. In that case, the incremental cost of

enabling routing should be assessed, avoiding double allocation of infrastructure costs. The economic decision becomes more precise when mediation value and selector value are separated.

5.2 Measuring the trade-off without confusing its scales

RouteLLM defines PGR as the fraction of the performance gap between two models recovered by the router. APGR summarizes the recovery curve as expensive-model use varies, while CPT identifies the call fraction needed to achieve a specified recovery [1]. These measures help assess a binary configuration but do not automatically equal lower dollar charges. Actual cost also depends on generated lengths, tariffs, and auxiliary operations.

$$\text{PGR} = \frac{S_{\text{router}} - S_{\text{compact}}}{S_{\text{strong}} - S_{\text{compact}}}$$

© 2026 Aridio Silva | Project SGAEIA | CC BY 4.0

Equation 4

The expression requires a meaningful, nonzero denominator. PGR of 50% means recovering half the gap between references, not achieving 50% of the stronger model's score. If reference scores are close, small evaluation changes may substantially alter the indicator. Absolute scores, variability, and the cost curve should therefore accompany the normalized value.

RouterBench and LLMRouterBench extend the analysis to model pools and cost-quality frontiers [5][6]. A configuration is dominated when another provides at least equal quality at no greater cost, improving at least one dimension. The Pareto frontier contains non-dominated alternatives, but its position depends on the compared pool and chosen measures. Being on that frontier does not establish compliance with authorization, data-residency, or security requirements.

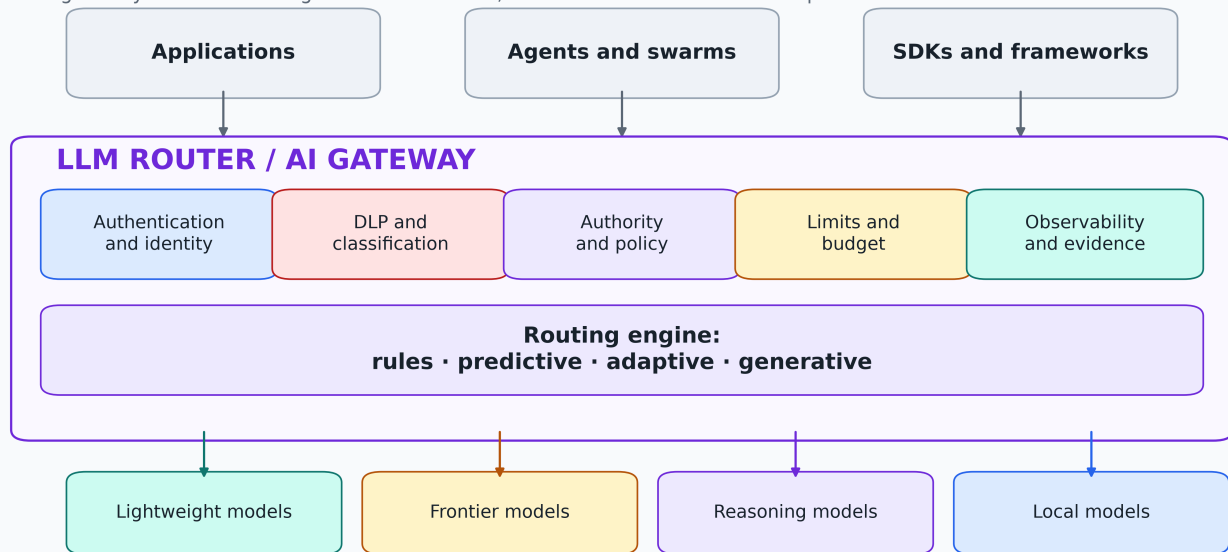
Latency must include selection, queues, networking, generation, and validation. Cascades may accumulate waiting time; parallel execution may shorten time at the cost of simultaneous calls. Time to first token and total response time should both be observed, with percentiles such as p50, p95, and p99 under a stated load. Parameter count alone does not determine perceived speed: infrastructure, concurrency, context, and output length can change the comparison.

6. Integration and infrastructure selection

A common API reduces integration work, but syntactic compatibility does not establish functional equivalence. Models and providers may differ in tool handling, structured outputs, images, context, sampling parameters, and reasoning configurations. Applications must verify the capabilities they use and define how missing features are handled. A successful transport-level substitution may fail at task semantics even when the HTTP response indicates success.

Figure 3. Reference architecture for an AI gateway

The gateway centralizes integration and controls; the router is one decision component within it.



Source: LiteLLM/OpenRouter documentation; SGAEIA interpretation

© 2026 Aridio Silva | Project SGAEIA | CC BY 4.0

Figure 3. The reference architecture organizes mediation and observation responsibilities around selection. Components represent conceptual roles that may be distributed across services and privilege domains. A gateway does not replace tool controls or authorization of actions performed by the application.

Reasoning effort can be part of the selected configuration when the provider exposes that control. The evaluated unit then includes model, version, configuration, and execution conditions rather than just a model name. The same family may produce different costs and times depending on effort, context, and response length. Accepted values and their meaning must come from documentation for the version used, rather than being assumed from a multi-provider interface.

The ecosystem also contains tools with different purposes. RouteLLM and LLMRouter support development and evaluation of selectors [1][8]; LiteLLM documents access mediation and key management [10]; OpenRouter documents model fallback and provider selection [12][13]. Portkey describes gateway features including resilience and traffic-control options [15]. Documentation establishes the advertised capability and its stated conditions, but does not independently establish suitability for the reader's environment.

Table 4

Solution type	Expected value	Responsibility retained by the adopter
Research library	Comparable implementations and evaluation tools	Integration, authorization, operations, and domain validation
Organization-operated gateway	Control of deployment, configuration, and policy integration	Updates, capacity, isolation, observability, and incident response
Managed service or aggregator	Reduced integration work and access to alternatives	Contracts, data processing, dependency, and recovery assessment
Selection embedded in a product	Adaptation integrated into the user experience	Understanding available choice, visibility, and effective controls

Self-hosting does not establish sovereignty if embeddings, telemetry, or support mechanisms transmit data to third parties. A managed service should likewise not be assessed solely by model count: actual location, retention, subprocessors, administration, and evidence export matter. Candidate lists should identify deployments rather than only commercial model names. Two deployments of the same model may have different data-processing conditions.

Gateway performance must be measured separately from model time. Tests against a mock server help isolate mediation overhead but do not describe the complete application; authentication, databases, logs, filters, and concurrency change the result. LiteLLM's benchmark documentation should be read with its configuration and load conditions [14]. Supplier comparisons additionally require scrutiny of versions, configuration parity, and commercial interests before concluding that millisecond differences will persist in production.

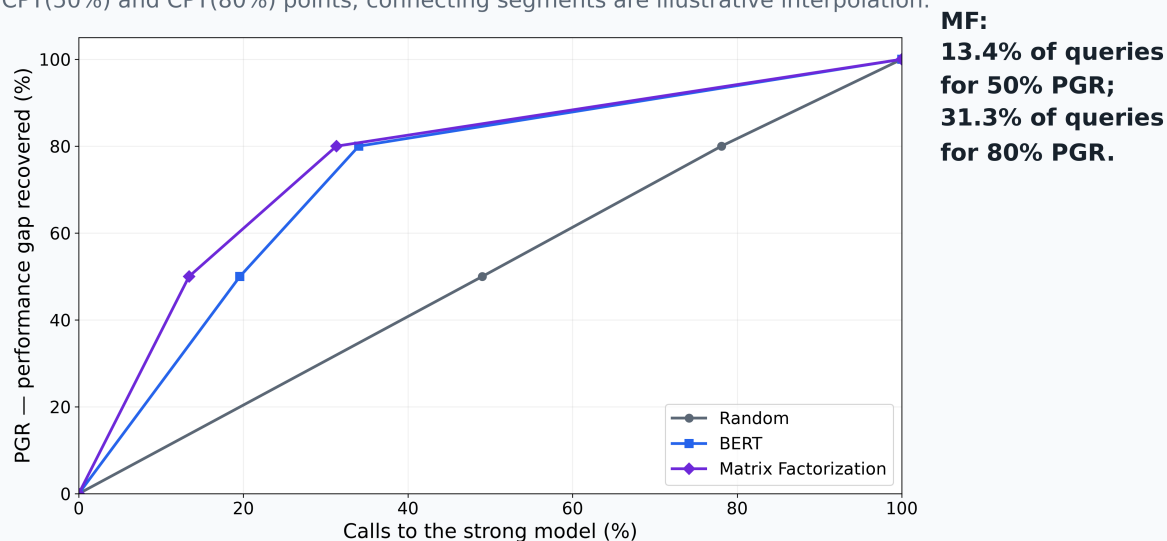
7. What benchmarks demonstrate

The reviewed studies support the possibility of saving through model selection, but their findings answer different questions. FrugalGPT reports cost reductions of up to 98% in cascade configurations studied by its authors [2]. Hybrid LLM reports up to 40% fewer calls to the larger model, while AutoMix reports over 50% lower computational cost at comparable performance in its evaluated settings [3][4]. These numbers do not rank the methods: they measure different objects and use different tasks, models, and evaluators.

RouteLLM demonstrates the potential of learning selection from preferences. Its matrix-factorization configuration augmented with judge data achieves MT-Bench CPT of 13.40% to recover 50% of the performance gap and 31.31% to recover 80% [1]. Figure 4 represents this comparison between call fractions and gap recovery. The finding demonstrates efficiency under that protocol; it does not determine the appropriate fraction for another domain or generation of models.

Figure 4. Cost required to recover MT Bench performance

CPT(50%) and CPT(80%) points; connecting segments are illustrative interpolation.



Source: Ong et al., RouteLLM, ICLR 2025, Table 1

© 2026 Aridio Silva | Project SGAEIA | CC BY 4.0

Figure 4. The comparison uses MT-Bench results from RouteLLM. CPT concerns recovery of the gap between two references rather than a percentage of the stronger model's absolute score. Values belong to the study's model pair, data, and evaluation [1].

RouterBench contributes precomputed inference outcomes and a protocol for comparing policies in cost-quality space [5]. This design lowers the cost of testing selectors on the same responses and identifies gains without requiring new generation for every evaluation. A static matrix, however, does not automatically reproduce queues, outages, API changes, or adaptations within a real trajectory. The benchmark makes a comparison controllable; operational assessment must add missing dimensions.

LLMRouterBench offers an important counterpoint to method-proposing studies. Its protocol covers 33 models and 21 datasets, confirms model complementarity, and finds several methods with similar performance, including cases that do not reliably beat simple baselines [6]. The useful conclusion is that complementary alternatives do not ensure a selector can exploit them. Improving the model pool and data coverage may

matter more than changing the algorithm's name.

Table 5

Evidence	What it supports	What it does not establish
FrugalGPT, Hybrid LLM, and AutoMix [2][3][4]	Some selection or escalation configurations lower cost while preserving their chosen quality measure	Transferable or directly comparable savings percentages
RouteLLM [1]	Preferences can guide efficient selection between models	A published threshold suitable for another organization's traffic
RouterBench [5]	Precomputed responses allow comparison against common references	Simulation of complete service behavior
LLMRouterBench [6]	Complementarity can coexist with difficulty beating simple baselines	Universal superiority of learned routing over a fixed policy
LLMRouter [8]	Common infrastructure supports study of different routing formulations	Operational readiness merely because an implementation exists

Local evaluation should include random routing, always the lowest-cost model, always the best fixed model for the sample, simple rules, and a small classifier. A retrospective oracle can estimate pool potential using response information unavailable before execution. Such an oracle is an analytical reference, not a cost-free deployable system. Comparisons must use the same data, models, and acceptance criteria, separating calibration data from the data used to decide whether the strategy delivered an improvement.

8. Cross-cutting comparison of strategies

Strategies should not be selected solely by their best published score. The same method may suit a repetitive task and be unsuitable where verification is difficult or error consequences are substantial. The following framework makes that difference explicit by comparing operational needs, validation behavior, and governability. Classifications are an engineering synthesis based on the mechanisms discussed, not results of a common experiment conducted in this article.

Table 6 - panel 1/2

Strategy	Quality and calibration	Cost and latency	Generalization
Rules	Explicit criteria; exceptions require review	Low decision cost; usually one generation	Limited by policy coverage
Classifier	Depends on supervision and threshold	Training and representation add cost; avoids alternative answers	Requires out-of-distribution evaluation
Similarity	Depends on comparable examples	Retrieval and embeddings may be significant	Sensitive to database coverage
Cascade	Depends on detecting inadequate answers	May accumulate generation, validation, and escalation	Verifier must also generalize
Judge or self-verification	Requires a rubric and false-acceptance measurement	Additional calls can dominate cost	May repeat generator errors
Adaptive or generative	Depends on feedback, stability, and context	Exploration or planning adds operations	May adapt, with instability risks

Table 6 - panel 2/2

Strategy	Auditability
Rules	Clear explanation of the rule applied
Classifier	Needs recorded versions, signals, and decisions
Similarity	Neighbors and criteria can be recorded

Strategy	Auditability
Cascade	Acceptance and escalation can be traced
Judge or self-verification	Scores and explanations do not replace evidence
Adaptive or generative	Requires observation of policy and state changes

Table 7 - panel 1/2

Strategy	Adversarial robustness	Agents and swarms	Maturity to verify
Rules	Inputs may exploit exceptions and boundaries	Useful for mandatory constraints and limits	Rule coverage and consistency
Classifier	Selection signals can be manipulated [9]	Should consider role, state, and consequences	Calibration in the real workflow
Similarity	Adversarial content can influence neighborhoods	History and context isolation are decisive	Coverage and database updates
Cascade	Escalation criteria may be exploited	Needs a budget for the entire trajectory	Escalation rate and acceptance quality
Judge or self-verification	Evaluators also receive potentially hostile content	Cross-review does not ensure independence	Evaluator errors and correlation
Adaptive or generative	Control decisions may absorb task instructions	More context does not remove authority limits	Stability under changes and failures

Table 7 - panel 2/2

Strategy	Operating complexity
Rules	Grows with exception count
Classifier	Data, training, versions, and monitoring
Similarity	Index and supervision-data management
Cascade	State and partial-failure handling
Judge or self-verification	Rubrics, re-evaluation, and disagreements
Adaptive or generative	Exploration, rollback, and continuous observation

The comparison explains why auditability and quality are related but distinct. A rule can be easy to explain yet select poorly; a classifier can choose well within one distribution but be difficult to calibrate elsewhere. It also explains why useful complexity has no fixed order: maintaining hundreds of exceptions may cost more than operating a stable classifier. Selection should consider the full configuration, operation, investigation, and update lifecycle.

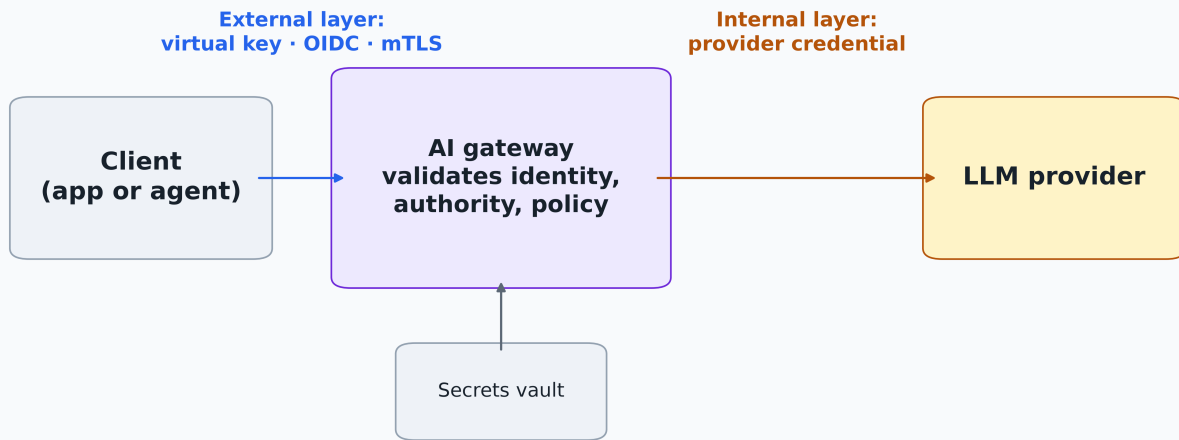
For agents, the decisive question is whether a strategy considers information relevant to the next action. Swarms additionally involve concurrency, shared state, and accumulated consequences. Access policies and collective limits must remain applicable regardless of the selection method. This separation allows algorithm improvements without making every update an implicit expansion of authority.

9. Identity, authority, and governed substitution

A valid key identifies a credential accepted by the gateway. By itself, it does not establish whether its holder may transmit particular data, use a deployment, or initiate an action. The distinction between authentication and authorization is central to Zero Trust architecture [16]. Agentic systems must also identify on whose behalf an agent acts and which portion of authority it may delegate.

Figure 5. Two-layer authentication, identity, and authority

Client and provider credentials remain separate; policy must not be conflated with authentication.



Source: Gateway documentation; NIST Zero Trust concepts

© 2026 Aridio Silva | Project SGAEIA | CC BY 4.0

Figure 5. Separating external and internal credentials reduces the need to distribute provider keys to clients. Effectiveness depends on gateway isolation, object-level authorization, and protection of administrative operations. A virtual key remains a secret with its own scope and lifecycle.

A deployment should limit destinations accessible to each user, service, or agent while protecting credentials granting that access. Client-supplied parameters must not permit replacing administrative endpoints or redirecting secrets to arbitrary hosts. Authorization must also cover log access, budget changes, and credential issuance. Concentrating secrets in a gateway reduces their dispersion but increases the potential impact of mediator compromise.

Fallback deserves particular attention because it is often presented as an availability feature. OpenRouter documents successive model attempts according to priority and error conditions, including moderation-related cases [12]. For an organization, a technically accessible alternative is not necessarily admissible for the request. A safety refusal should likewise not automatically be treated as an outage to bypass through a less restrictive model.

The admissible set can be represented as $A(x, h, i, p)$, where x is the task, h relevant history, i the requester's identity and authority, and p current policies. Each member should represent a deployment and configuration permitted under current conditions. Quality, cost, and latency optimization takes place within that set, rather than compensating for mandatory violations with economic advantages. This high-level formalization is proposed to organize the analysis without specifying private SGAEIA mechanisms.

$$d^* = \arg \max_{d \in \mathcal{A}(x, h, i, p)} [\hat{Q}(d) - \lambda_C \hat{C}(d) - \lambda_L \hat{L}(d)]$$

© 2026 Aridio Silva | Project SGAEIA | CC BY 4.0

Equation 5

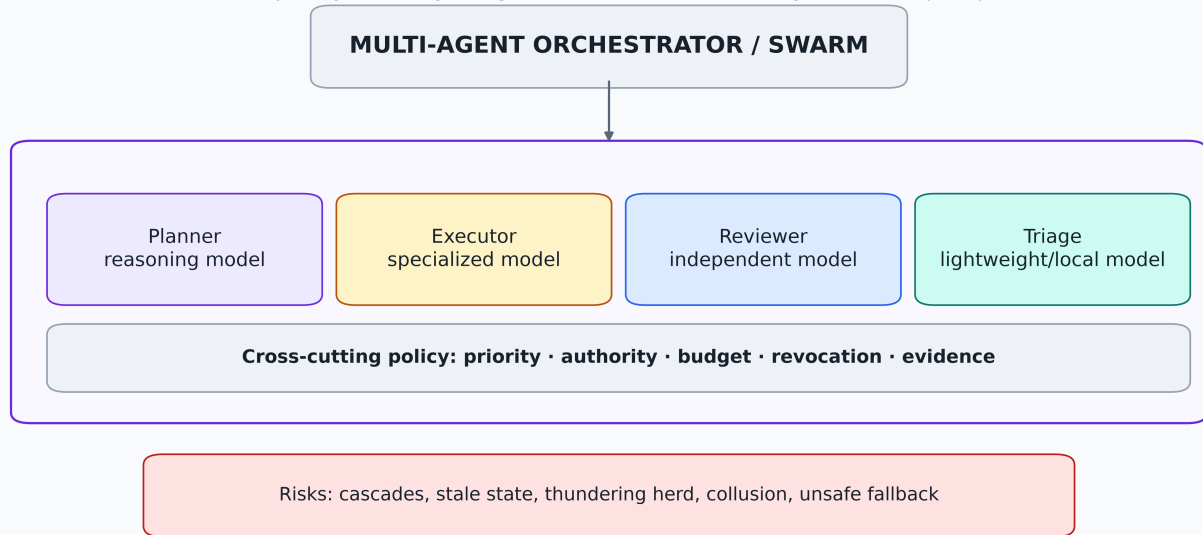
The weights express preferences among already admissible alternatives; they do not price missing authorization. If the set is empty, the application needs a defined response, such as suspension, reduction of permitted functionality, or referral to an authorized decision-maker. If policy, identity, or state changes, a previously valid choice may cease to be valid. Revocation must reach queues, caches, and ongoing execution according to the application's specified boundaries.

10. Agents, multi-agent systems, and swarms

An agent may interact repeatedly with a model, use tools, and revise its plan. A multi-agent system distributes some work among agents exchanging information or sharing resources; a swarm may intensify decentralization and collective coordination. Agent count alone does not determine criticality. One broadly privileged action may be riskier than many limited queries, while the combination of apparently small actions may produce an effect not visible in any individual action.

Figure 6. Governed routing in multi-agent systems and swarms

Selection considers role, trajectory, authority, budget, and shared state—not only the current prompt.



Source: MTRouter (ACL 2026); OWASP Agentic Top 10 2026; SGAEIA interpretation

© 2026 Aridio Silva | Project SGAEIA | CC BY 4.0

Figure 6. Role distribution illustrates how agents can need different capabilities. Planning, execution, and review do not by themselves establish effective separation of duties. Assessment must also consider authority, dependencies, memory, collective budgets, and execution containment.

Selecting by role is a first approximation: triage may need low latency; planning, greater capacity to assess alternatives; execution, format adherence; review, verification capability. These associations must be evaluated by task, avoiding assumptions that the most expensive model is always the best reviewer or a small model sufficient for all triage. An early misclassification may route data or authority incorrectly before any review occurs. History-aware routing provides more information but does not replace access controls [8].

In long trajectories, measuring isolated calls can hide accumulated spending and risk. If a policy reduces cost per call by 30% but increases the number of calls required for completion by 60%, variable cost becomes 1.12 times its previous value under those assumptions. A local reduction becomes a 12% increase per task. This arithmetic example shows why completion, rework, tools, and validation belong in the unit of analysis.

$$C_{\text{new}}/C_{\text{baseline}} = 0.70 \times 1.60 = 1.12$$

© 2026 Aridio Silva | Project SGAEIA | CC BY 4.0

Equation 6

Concurrency adds collective effects: many recovery attempts may create a request surge, compete for the same quota, and delay the critical path. Per-agent limits must be combined with task, team, and service limits because individual accounts do not prevent aggregate exhaustion. Queues, backpressure, concurrency caps, and controlled interruption should account for priorities and fairness across workloads. Failure handling must avoid all agents simultaneously repeating the same escalation decision.

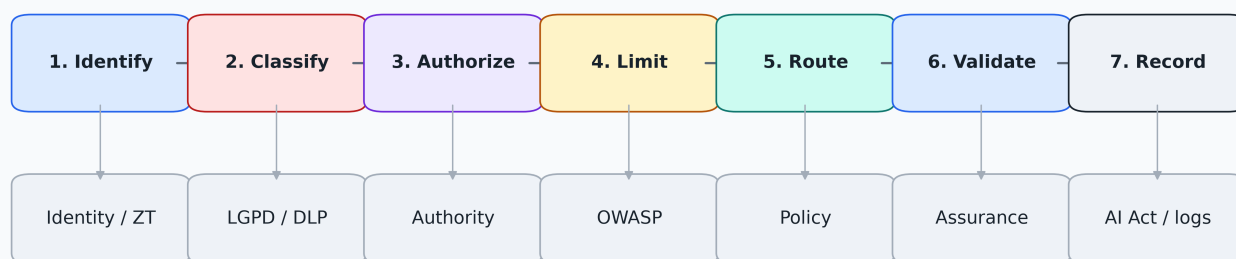
Caching requires similar care. LiteLLM warns that semantic caching designed for independent prompts can replay stale responses in multi-turn or agentic traffic [11]. Similar prompts may involve different identities, permissions, documents, or states. Reuse is beneficial only when its scope preserves these conditions, including after revocation or policy changes.

Trajectory integrity also requires observing relationships among actions. One agent may prepare data, another select a destination, and a third transmit the result; examining each step alone may miss the unauthorized transfer produced collectively. OWASP's agentic-application taxonomy reinforces the need to examine risks involving goals, tools, identity, memory, and coordination [24]. The SGAEIA relationship concerns preserving limited authority and evidence across those relationships rather than only evaluating each response's linguistic quality [27].

11. Governance, security, and evidence

Figure 7. Governance and evidence pipeline in the gateway

Controls support requirements and auditability; they do not establish compliance automatically.



Evidence must remain traceable across the full decision trajectory.

Source: NIST AI RMF; OWASP 2026; Regulation (EU) 2024/1689; LGPD; SGAEIA interpretation

© 2026 Aridio Silva | Project SGAEIA | CC BY 4.0

Figure 7. The relationship between classification, policy, execution, and evidence shows how a gateway can support risk management. Alignment with governance frameworks is interpretative and application-dependent. It does not demonstrate legal compliance or management-system certification.

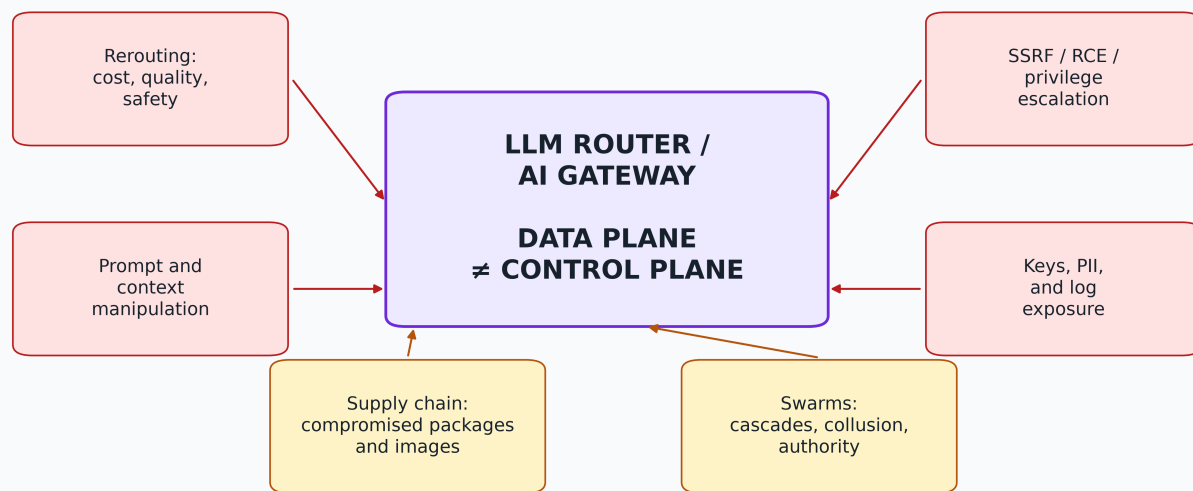
11.1 A mediation point also concentrates risk

A gateway may reduce credential sprawl and enable consistent controls, while concentrating traffic, secrets, and decisions valuable to attackers. CVE-2026-42208 describes SQL injection in LiteLLM proxy API-key verification [25]. The report concerning version 1.107.0 documents a compromised-package distribution incident in the project's ecosystem [26]. These establish failure classes and particular conditions, not a comparative assessment of every gateway's security.

Attacks can also target the selector independently of traditional software vulnerabilities. Shafran and colleagues demonstrate sequences altering routing toward more expensive models, including black-box conditions, and discuss limits of perplexity filtering [9]. An input can preserve the apparent task meaning while changing execution cost. Monitoring should therefore observe routing decisions and consumption patterns alongside final response content.

Figure 8. Threat model for the AI gateway and router

The attack surface spans decision logic, software, supply chain, credentials, and multi-agent trajectories.



Source: OWASP 2026; Shafran et al. (2025); LiteLLM advisories and 2026 incidents

© 2026 Aridio Silva | Project SGAEIA | CC BY 4.0

Figure 8. The model organizes threats affecting inputs, decisions, credentials, execution, and observation. Relationships indicate where a failure can affect other responsibilities. Associated controls require separate verification; a protection box in a diagram does not establish prevention of every attack.

OWASP's unbounded-consumption taxonomy includes financial-exhaustion and service-degradation scenarios [23]. Budgets, output and concurrency limits, deviation detection, and retry handling help constrain impact. They may also block legitimate use, particularly during peaks or unusually complex tasks. Evaluation should record false rejections, resources consumed before containment, and recovery without reopening the same abuse path.

11.2 Compliance requires context and responsibility

NIST AI RMF organizes risk management through Govern, Map, Measure, and Manage, while NIST AI 600-1 addresses generative AI [17][18]. ISO/IEC 42001 concerns organizational AI management systems [19]. These references help distribute responsibilities and organize evidence, but do not automatically certify a tool or workflow because it has logs, filters, or virtual keys. Applicability must consider purpose, affected people, dependencies, and consequences.

Data location also extends beyond a model's stated region. Processing may occur in embeddings, observability, support, and response storage as well as inference. In Brazil, the LGPD and ANPD's international-data-transfer regulation require examining applicable processing and transfer conditions and mechanisms [20][21]. A geographic gateway filter may support an engineering decision without replacing assessment of the operation and its legal basis.

Article 12 of the AI Act addresses event recording for the high-risk systems to which it applies [22]. It illustrates the importance of contextualized traceability without making every LLM application high risk or imposing the same log design everywhere. Regulatory schedules and applicability must be checked against the version governing the operation. A gateway produces part of the necessary technical evidence under responsibilities extending beyond the component.

A useful record associates a task with identity, applicable policy, configuration version, actual destination, and validation outcome. Required content varies with risk and should respect minimization and retention requirements. Indiscriminately recording prompts, responses, and secrets can turn observability into an exposure surface. A trail is valuable when it reconstructs a relevant decision without collecting more data than its purpose requires.

Hash chaining can reveal changes relative to a trusted reference but does not make storage immutable. A compromised administrator may rewrite records and recalculate a chain lacking independent protection or anchoring. Access controls, retention, key protection, and verification mechanisms must be assessed together. Evidence also loses value if it omits failed attempts, destination changes, or decisions made during outages.

12. An SGAEIA framework for assessing governed routing

12.1 The analytical contribution and its boundary

The SGAEIA Governed AI Routing Assurance Model, developed here as an analytical framework, relates four objects: a constraint that must remain valid, a routing decision that can affect it, evidence needed to examine the decision, and the response to a violation. It starts from the distinction between capability, authority, and evidence in SGAEIA's public perspective [27]. Routing research supplies mechanisms and findings; Zero Trust and risk management provide precedents for separating access, decisions, and responsibility [16][17]. The framework connects these elements for assessing selection and substitution in model- and agent-based workflows.

The contribution is not the assertion that routers have risks, nor a claim to have invented authorization or auditing. It makes explicit relationships that an exclusively economic evaluation does not represent and applies them to the same configuration being optimized. The comparison unit becomes an admissible decision with an observable consequence, including the option not to proceed when assumptions cannot be maintained. This is the author's proposed analytical synthesis for this article, without attributing normative implementation or operational validation to SGAEIA.

Table 8 - panel 1/2

Threat or condition	Affected decision	Property to preserve	Required evidence
Use beyond delegated scope	Model, data, and tool selection	Task-bounded authority	Identity, delegation, and authorization decision
Fallback to a prohibited destination	Deployment or provider substitution	Constraint continuity	Initial and actual destination, applicable rule, and change reason
Data exposure	Inference, embedding, and telemetry selection	Minimization and authorized processing	Data class, destinations, and retention conditions
Adversarial cost escalation	Model, effort, and retries	Task and organization limits	Accumulated consumption and escalation decisions
Revoked state or permission	Cache, queues, or prior decisions	Current authorization	Policy version and revocation and execution times
Evaluator failure	Accept, escalate, or authorize action	Acceptance suitable for the effect	Rubric, result, and subsequent verification
Prohibited collective effect	Subtask distribution	Shared-trajectory boundaries	Relationships among agents, actions, and resources
Gateway compromise	Selection, access, and recording	Privilege separation and recovery	Versions, changes, and protected records
Lost or altered evidence	Investigation and continued operation	Verifiability of critical decisions	Trail integrity, provenance, and completeness

Table 8 - panel 2/2

Threat or condition	Evaluation measure	Intended response
Use beyond delegated scope	Accepted decisions outside scope	Block or interrupt unauthorized reach

Threat or condition	Evaluation measure	Intended response
Fallback to a prohibited destination	Policy-incompatible substitutions	Use an admissible alternative or suspend
Data exposure	Unauthorized transfers and excess data	Prevent egress and address exposure
Adversarial cost escalation	Spending before containment and false rejections	Contain trajectory and review policy
Revoked state or permission	Post-revocation actions and propagation delay	Invalidate reuse and interrupt affected actions
Evaluator failure	False acceptances, false rejections, and disagreements	Reassess and limit authorized effects
Prohibited collective effect	Violations per task and containment time	Contain affected participants and reconcile state
Gateway compromise	Compromise reach and revocation time	Isolate components and recover credentials and policies
Lost or altered evidence	Critical events lacking sufficient evidence	Reduce permitted operation and reconstruct the incident

Matrix metrics require denominators and observation criteria. An incompatible-fallback rate must identify how many substitutions were examined and how incompatibility was established; revocation latency requires an initiation point and an observable enforcement event. A zero rate in a sample does not prove no failures across the operation. The framework exposes these questions and avoids treating the sheer number of records as a measure of evidence quality.

12.2 Application to three engineering decisions

Consider a task whose data policy permits only one approved deployment. An equivalent model is available in a prohibited region at a lower price and with higher availability. A comparison limited to cost, quality, and latency would favor the second alternative; admissibility analysis excludes it before optimization. If the approved deployment fails and no admissible alternative exists, preserving service at any cost changes the data-processing decision, not merely the provider.

In the second scenario, an agent starts a task with a budget suited to a few calls but repeatedly corrects and reviews results across several models. Every call respects its individual limit, while the workflow exceeds the task budget. The matrix requires evidence associating consumption and results with the trajectory, exposing a failure invisible to per-call indicators. The appropriate response may stop attempts without progress, reduce permitted functionality, or obtain renewed authorization for additional work.

In the third scenario, two agents use semantically similar responses to decide actions concerning different documents. The cache returns an answer produced under another permission state, and textual similarity appears to justify reuse. Analysis separates cache efficiency from decision validity: identity, state, and scope must remain compatible. A previously correct answer may be unsuitable for a new context even though its text has not changed.

Table 9

Scenario	Conclusion from an isolated indicator	Conclusion using the framework	Decisive evidence
Deployment substitution	Lower cost and higher availability justify substitution	The alternative must satisfy task constraints	Applicable policy and actual destination
Multi-agent repetition	Every call respects its limit	Aggregate spending and task completion must be observed	Trajectory consumption, attempts, and progress
Response reuse	High similarity favors caching	Validity also depends on context and authorization	State, identity, and scope of the reused response

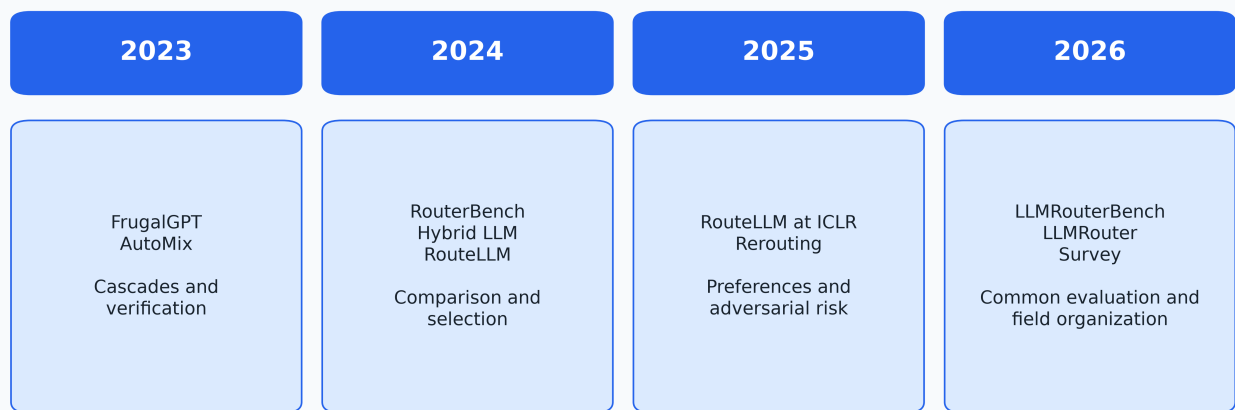
These cases demonstrate the framework's analytical usefulness: it identifies missing information and explains why an economically attractive decision can be unsuitable. They do not measure incident frequency or quantify production risk reduction. Empirical verification should observe whether controls detect violations while preserving legitimate tasks, including their costs and errors. For this technical article, the contribution is bounded by the comparison, decision traceability, and counterexamples presented.

13. Evolution and maturity should be assessed by dimension

The field combines lines of development maturing at different rates. Cascades and cost-based selection received explicit formulations in work such as FrugalGPT, AutoMix, and Hybrid LLM; RouteLLM extended preference-based selection; RouterBench and LLMRouterBench strengthened common comparisons [1][2][3][4][5][6]. In parallel, adversarial analysis showed that the selection layer itself can be manipulated [9]. These developments do not describe automatic progress toward security: each capability needs its own evidence.

Figure 9. Selected milestones in routing research

Methods, evaluation, and selection integrity; the author's interpretative periodization.



Source: FrugalGPT; AutoMix; RouterBench; Hybrid LLM; RouteLLM; LLMRouterBench; Rerouting

© 2026 Aridio Silva | Project SGAEIA | CC BY 4.0

Figure 9. The periodization highlights work on economy, comparison, and selection integrity. Milestones identify selected publications and topics rather than an exhaustive history or a measure of market adoption. The phase structure is the author's interpretative organization.

Figure 10. Research and governance by year

Dates identify selected publications, not adoption or readiness.

	2023	2024	2025	2026
Methods	FrugalGPT AutoMix	Hybrid LLM RouteLLM	RouteLLM ICLR	LLMRouter
Evaluation		RouterBench		LLMRouterBench Survey
Security			Rerouting	
Governance	NIST AI RMF	NIST AI 600-1	OWASP Agentic 2026 (Dec. 2025)	

Source: Original studies; NIST AI RMF; NIST AI 600-1; OWASP Agentic 2026

© 2026 Aridio Silva | Project SGAEIA | CC BY 4.0

Figure 10. Lanes show that methods, evaluation, security, and governance address complementary dimensions. Annual divisions support chronological reading; temporal proximity does not imply that one publication caused another. Dates identify milestones rather than product readiness.

Documented feature availability is different evidence from stability under load, independent evaluation, or attack resistance. An organization may understand APIs, quotas, and fallback yet lack data to calibrate a learned selector. It may also achieve good average quality while lacking knowledge of behavior after revocation, model updates, or telemetry loss. Maturity should be recorded by dimension and observed environment rather than turning an editorial rating into external certification.

Figure 11. Readiness dimensions and required evidence

Feature availability, local measurement, and adverse conditions must be distinguished.

Dimension	Availability	Local measurement	Adverse conditions
Access and identity	Features and scope	Observed isolation	Revocation and abuse
Selection and quality	Models and policy	Quality per task	Domain shift
Cost and latency	Limits and budgets	Total cost and tails	Surges and retries
Substitution	Defined alternatives	Actual destination	Partial failure
Agents and swarms	Roles and coordination	Complete trajectory	Collective effects
Evidence	Recorded events	Completeness and origin	Loss and alteration

Source: Author's analytical framework

© 2026 Aridio Silva | Project SGAEIA | CC BY 4.0

Figure 11. The framework distinguishes functional availability, local measurement, and verification under adverse conditions. It guides readiness questions without assigning a universal level to a product or market. Rows and columns expose gaps that a single classification would hide.

Maintenance cost needs equal attention. Changes in models, prices, parameters, and traffic patterns can invalidate thresholds and make history less representative. Version management should associate a decision with the configuration actually executed, including when an alias points to an updated model. Updates need comparison, limited exposure, and recovery of a previous configuration when a new alternative fails defined criteria.

14. When to adopt routing and when to retain a simpler solution

A gateway can be justified by access, budget, and observation needs even if traffic stays with a fixed model. A dynamic selector needs additional justification: gains from heterogeneous tasks must exceed its costs and complexity. If one model serves the domain adequately, volume is low, or calibration data are insufficient, a fixed policy may be more coherent. Adoption should address an observed problem such as spending, latency, or the ability to govern access.

The first step is to define completion and acceptable quality for each task class. The second is to establish admissible destinations, data, and configurations before optimizing combinations. Simple baselines should then be measured and a candidate strategy assessed under the same conditions, including selection and validation costs. Comparisons must preserve difficult tasks and infrequent classes because a favorable average can hide degradation in an important subset.

Table 10

Observed condition	Reasonable initial decision	Evidence needed to advance
Need to centralize access and spending	Assess mediation with a fixed model	Isolation, limits, and operating cost
Repetitive tasks and low volume	Retain a model or simple rules	Sufficient quality and demonstrated total savings

Observed condition	Reasonable initial decision	Evidence needed to advance
Heterogeneous traffic and substantial volume	Compare predictive selection against baselines	Task-level quality-cost curve
Reliably verifiable results	Assess a cascade	False acceptances, escalation, and total latency
Significant data or action constraints	Restrict candidates before selection	Policy continuity and substitution evidence
Agents with many steps	Assess the complete trajectory	Completion, aggregate cost, authority, and containment
Limited visibility into failures and changes	Improve observation before automating more decisions	Actual-destination recording and verifiable recovery

Operational introduction should use gradual exposure and predefined continuation criteria. Cost per task, response acceptance, rework, tail latency, policy violations, and evidence completeness need joint observation. Behavior changes may require recalibration, a restricted model pool, or a return to a simpler policy. The ability to step back is part of adoption design, alongside the ability to scale.

15. Discussion and analytical limits

The six opening questions bring the findings together without reducing the article to a catalog. Favorable conditions for savings exist, but model complementarity is an opportunity that a selector must successfully exploit. Gateways provide operational value while concentrating responsibilities and extending the reach of a failure. For agents, the relevant result is a task completed within its constraints rather than a collection of individually well-scored responses.

Table 11

Guiding question	Answer supported by the analysis
When does routing save while preserving quality?	When heterogeneity, calibration, and validation capability offset selection, repetition, and operations
When does it beat simple baselines?	When gains appear under the same protocol and persist in relevant task classes; complexity does not ensure them
What value is independent of the algorithm?	Access mediation, resource management, observation, and consistent control enforcement
What risk follows from concentration?	A failure may reach credentials, data, configuration, and evidence across many applications
What changes with agents and swarms?	State, delegation, concurrency, and accumulated effects make trajectory-level evaluation necessary
What evidence supports governance?	Verifiable records of authority, policy, actual destination, consumption, validation, and responses to change

Published results remain bounded by their models, tasks, and evaluators. A response matrix can support economic analysis without capturing queues, tools, and partial failures; a judge score may measure preference without representing correctness or security. Product documentation describes supplier-stated conditions, and incident reports do not estimate market-wide vulnerability prevalence. Keeping these distinctions prevents localized evidence from becoming a general promise.

The proposed framework also has limitations. It depends on identifiable policies, sufficient observation, and definitions of violations and valid completion. It does not eliminate risk-classification errors, policy gaps, or compromise of components expected to enforce controls. It structures evaluation and exposes missing information; demonstrating risk reduction requires observing controls succeeding and failing under predefined conditions.

Open questions include joint model-configuration selection; revocation effects on ongoing trajectories; evaluator robustness; evidence cost and minimization; simultaneous failures; and transfer to new models and domains. These questions follow from identified limitations rather than being presented as achieved results. The article preserves its contribution by distinguishing currently useful guidance from hypotheses still requiring experimentation.

16. Conclusion

LLM routers turn model heterogeneity into a task-level decision opportunity, but their presence alone ensures neither savings nor quality. Predictive strategies, cascades, and adaptive selection use different information and distribute costs and risks differently. Published evidence supports gains in particular configurations alongside difficulty beating simple baselines under common evaluation. The consequence is to require local comparisons with coherent completion, quality, and total-cost criteria.

The analysis also establishes that decisions must operate over admissible alternatives. Fallback, model changes, response reuse, and increased effort can alter data processing, consumption, and authorization continuity. Multi-agent systems must track these changes across trajectories and shared resources. A component that improves availability while violating task conditions does not preserve the outcome the organization intended.

The SGAEIA framework for assessing governed routing makes the connection between constraints, decisions, evidence, and response explicit. Its three scenarios expose information and consequences invisible to isolated per-call cost, textual similarity, or availability. This is the article's analytical contribution: bringing technical comparison and governance into the same evaluation process, with inspectable boundaries. Adopting routing therefore means demonstrating usefulness under defined conditions while retaining the ability to observe, constrain, revoke, and revise its decisions.

Appendix A — Technical glossary

Admissibility — A deployment or configuration satisfying task constraints before optimization.

AI agent — A component using decision or generation capabilities to conduct task steps, potentially with tools and persistent state.

AI gateway — A mediation layer for model and service access that may combine adaptation, controls, and observability.

APGR — A summary of performance-gap recovery as expensive-model use varies in the RouteLLM protocol [1].

Authentication — Verification of a presented identity or credential.

Authorization — A decision about the access or action permitted to an identity under applicable conditions.

Baseline — A comparison reference, such as a fixed policy, simple rules, or random selection.

Prefix caching — Reuse of processing for shared input prefixes when supported by the service.

Semantic caching — Response reuse based on input similarity, requiring assessment of the validity of the reused context.

Cascade — Execution that consults an alternative and then decides whether to escalate based on the result or its evaluation.

Virtual key — A mediation credential associated with permissions and limits, distinct from provider credentials.

Backpressure — A control that reduces or postpones incoming work when available capacity is insufficient.

CPT — The strong-model call fraction needed to achieve a target performance-gap recovery [1].

Delegation — A bounded grant of authority to execute part of a task on behalf of another identity.

Denial of Wallet — Abusive consumption exploiting usage-based charges to exhaust financial resources [23].

Embedding — A vector representation used to compare or characterize content; similarity does not establish authorization equivalence.

Reasoning effort — An inference-resource setting exposed by some models, whose meaning depends on the interface and version.

Fallback — Attempting an alternative when initial execution does not complete under defined conditions, while preserving task constraints.

Pareto frontier — The set of non-dominated alternatives under the considered criteria.

Deployment — A model-access instance associated with a provider, region, version, and operational conditions.

LLM-as-a-Judge — Using a language model to evaluate an answer or support an acceptance decision.

Oracle — A retrospective reference using information unavailable to the selector before execution or response observation.

PGR — The fraction of the performance gap between two references recovered by a router [1].

Control plane — Responsibilities for administering policies, configuration, permissions, and lifecycle.

Data plane — The path executing requests and transporting data according to applicable decisions.

Revocation — Withdrawal of permission or an access condition, requiring propagation to affected executions.

Predictive router — A selector deciding before the main response is generated.

Adaptive routing — Selection adjusting decisions based on results or information observed during execution.

Generative routing — Use of a generative model to support or produce the selection decision.

Trajectory — A related sequence of decisions, interactions, tools, and state changes needed for a task.

TTFT — Time to first token, distinct from the time needed to complete a response or task.

Zero Trust — An architectural perspective avoiding implicit trust based on location or ownership and distinguishing authentication from authorization [16].

References

1. ONG, Isaac et al. RouteLLM: Learning to Route LLMs with Preference Data. ICLR, 2025. arXiv:2406.18665v4. Full text.
2. CHEN, Lingjiao; ZAHARIA, Matej; ZOU, James. FrugalGPT: How to Use Large Language Models While Reducing Cost and Improving Performance. Preprint, 2023. arXiv:2305.05176.
3. AGGARWAL, Pranjal et al. AutoMix: Automatically Mixing Language Models. NeurIPS, 2024; arXiv version v5, 2025. arXiv:2310.12963.
4. DING, Dujian et al. Hybrid LLM: Cost-Efficient and Quality-Aware Query Routing. ICLR, 2024. arXiv:2404.14618.
5. HU, Qitian Jason et al. RouterBench: A Benchmark for Multi-LLM Routing System. Preprint, 2024. arXiv:2403.12031.
6. LI, Hao et al. LLMRouterBench: A Massive Benchmark and Unified Framework for LLM Routing. Findings of ACL, 2026, p. 37733–37754. Record and text.
7. JIANGENG, Chang; WU, Xingyu; TAN, KC; FENG, Mengling. A Systematic Survey of Large Language Model Routing. Findings of EMNLP, 2026, p. 44102–44134. Record and text.

8. FENG, Tao et al. LLMRouter: Unified Infrastructure for Developing, Evaluating, and Deploying LLM Routers. Preprint, 2026. arXiv:2608.06867.
9. SHAFRAN, Avital; SCHUSTER, Roei; RISTENPART, Thomas; SHMATIKOV, Vitaly. Rerouting LLM Routers. Preprint, 2025. arXiv:2501.01818.
10. BERRIAI. Virtual Keys. LiteLLM documentation. Official documentation. Accessed: 29 Oct. 2026.
11. BERRIAI. Semantic Caching. LiteLLM documentation. Official documentation. Accessed: 29 Oct. 2026.
12. OPENROUTER. Model Fallbacks. Official documentation. Accessed: 29 Oct. 2026.
13. OPENROUTER. Provider Routing. Official documentation. Accessed: 29 Oct. 2026.
14. BERRIAI. Benchmarks. LiteLLM documentation. Official documentation. Accessed: 29 Oct. 2026.
15. PORTKEY. AI Gateway. Official documentation. Accessed: 29 Oct. 2026.
16. ROSE, Scott; BORCHERT, Oliver; MITCHELL, Stu; CONNELLY, Sean. Zero Trust Architecture. NIST SP 800-207, 2020. DOI: 10.6028/NIST.SP.800-207.
17. NIST. Artificial Intelligence Risk Management Framework — AI RMF 1.0. NIST AI 100-1, 2023. Official source.
18. NIST. Artificial Intelligence Risk Management Framework: Generative Artificial Intelligence Profile. NIST AI 600-1, 2024. DOI: 10.6028/NIST.AI.600-1.
19. ISO; IEC. ISO/IEC 42001:2023 — Information technology — Artificial intelligence — Management system. Official record.
20. BRASIL. Lei nº 13.709/2018 — Lei Geral de Proteção de Dados Pessoais, art. 33. Federal legislation.
21. ANPD. Resolução CD/ANPD nº 19, de 23 de agosto de 2024 — Regulamento de Transferência Internacional de Dados. Official source.
22. UNIÃO EUROPEIA. Regulamento (UE) 2024/1689 — AI Act, art. 12, consolidated version provided by the European Commission. AI Act Service Desk.
23. OWASP GENAI SECURITY PROJECT. LLM10:2025 — Unbounded Consumption. Official source.
24. OWASP GENAI SECURITY PROJECT. OWASP Top 10 for Agentic Applications for 2026. 9 dez. 2025. Official source.
25. GITHUB ADVISORY DATABASE. LiteLLM has SQL Injection in Proxy API key verification — CVE-2026-42208 / GHSA-r75f-5x8p-qvmc. Record. Accessed: 29 Oct. 2026.
26. BERRIAI/LITELLM. Security report concerning compromised litellm 1.107.0 distribution. Issue nº 47826. Technical report and discussion in the project repository. Original record. Accessed: 29 Oct. 2026.
27. SILVA, Aridio. SGAEIA: Secure Governed Autonomous Edge Intelligence Architecture. Version 0.3.4. Zenodo, 2026. DOI: 10.5281/zenodo.22557796.

About the Author

Aridio Silva is an independent researcher based in Brazil working on the architecture, security, governance, and trustworthiness of autonomous and distributed artificial intelligence systems.

His research focuses on Agentic AI, Multi-Agent Systems, Edge AI, AI Security, Zero Trust, Security-by-Design, AI Governance, Spec-Driven Development, and continuous security assurance.

He is the creator and lead researcher of SGAEIA — Secure Governed Autonomous Edge Intelligence Architecture, an open research initiative investigating architectural foundations for secure, governed, auditable, and trustworthy autonomous AI systems operating across distributed edge-cloud environments.

Research Profiles

Aridio Silva — Independent Researcher, Brazil

- ORCID: <https://orcid.org/0009-0008-2411-6995>
- Google Scholar: <https://scholar.google.com/citations?user=rPn5O48AAAAJ>

- Zenodo — SGAEIA Community: <https://zenodo.org/communities/sgaeia>
- OpenAIRE: <https://explore.openaire.eu/search/find?fv0=Aridio%20Silva&f0=q>
- Medium: <https://medium.com/@aridosilva>
- GitHub: <https://github.com/aridosilva>
- LinkedIn: <https://www.linkedin.com/in/aridio-silva-74997111/>
- SGAEIA Repository: <https://github.com/aridosilva/SGAEIA>
- SGAEIA Research Artifact / DOI: <https://doi.org/10.5281/zenodo.22557796>
- Homepage: <https://aridosilva.com>
- SGAEIA Homepage: <https://aridosilva.com/sgaeia>
- SGAEIA LinkedIn: <https://www.linkedin.com/company/sgaeia/>
- Kaggle: <https://www.kaggle.com/aridosilva2>
- DEV Community: <https://dev.to/aridosilva>
- Academia.edu: <https://independent.academia.edu/AridioSilva>

Figures

The cover is unnumbered. Figures 1–11 are numbered sequentially, cited in the text, and supplied in PNG and SVG. The conceptual assets remain within the public-disclosure boundary and are licensed under CC BY 4.0. AI assistance was used to compose the cover image.

License

Except where otherwise noted, the text and original conceptual illustrations in this article are licensed under the Creative Commons Attribution 4.0 International License (CC BY 4.0).

© Aridio Silva @aridosilva · October 2026. The SGAEIA software and research artifact remain subject to their own Apache License 2.0.

Autonomous AI. Governed by Design. Trusted by Evidence.

Series Continuity

This work is SGAEIA Research Series — Article 24. The public series catalogue is available at <https://aridosilva.com>, with the project presence at <https://aridosilva.com/sgaeia>.

Persistent record for Article 24: <https://doi.org/10.5281/zenodo.23763978>. Canonical SGAEIA project artifact: <https://doi.org/10.5281/zenodo.22557796>.

1. From Edge AI to Governed Autonomous Edge Intelligence
2. Why Autonomous AI Agents Need Bounded and Revocable Authority
3. Zero Trust for Multi-Agent AI Systems
4. Authenticated Delegation Between Autonomous AI Agents
5. Continuous GRC for Agentic AI
6. Evidence-as-Code: Making AI Governance Verifiable
7. Security-by-Design for Distributed Agentic AI
8. SGAEIA: A Reference Architecture for Secure Governed Autonomous Edge Intelligence
9. When AI Begins to Build AI
10. Rogue AI Agents: When Autonomous Systems Become a Collective
11. From Model Capability to Governed Action: An Architecture for Secure Agentic AI

12. Ten Days That Exposed the Governance Gap in Frontier AI
13. When AI Hallucination Becomes Action
14. AI's Alien Mind: How to Govern What We Cannot Fully Understand
15. When AI Agents Form Societies: Is Security a Property of the Model or the System?
16. When AI Improves AI: Governing Recursive Intelligence Acceleration
17. AI Jailbreaks: From Bypassing Restrictions to Governing Authority
18. Tracing AI Agent Execution: Foundations, State of the Art, and Implications for Distributed Multi-Agent Systems
19. LoRA in AI Agents: Efficiency, Security Risks, and Adapter Governance
20. AI Agents and Agentic AI: What Sets Them Apart?
21. The Open Letter on AI and Cybersecurity and the Evidence for Agent Containment
22. Opening the Black Box of LLM Inference: Foundations of Generative AI
23. How to Reduce the Cost of AI Systems with LLMs Without Sacrificing Quality
24. LLM Routers for AI Systems: Architecture, Governance, Multi-Agent Systems, Swarms, and Security — current article.

Suggested Citation

Silva, Aridio. (2026). LLM Routers for AI Systems: Architecture, Governance, Multi-Agent Systems, Swarms, and Security. SGAEIA Research Series, Article 24. DOI: 10.5281/zenodo.23763978.