MIXTURE-OF-MODELS: CONDITIONAL COMPUTATION ACROSS MODEL BOUNDARIES

vLLM Semantic Router Project

ABSTRACT

Modern AI deployments combine independently versioned models with different capabilities, costs, and governance constraints. We define a *Mixture-of-Models* (MOM) as a versioned composite model whose lifecycle engine realizes requests through preference-conditioned, resource-bounded execution traces over declared constituents and operators. Selection, cascades, parallel fusion, multi-round collaboration, and bounded workflows become topologies of one model; hard admissibility precedes soft utility optimization. We cast MoM as system-level intelligence: a full-stack co-design over workload, trace policy, constituent pool, and physical realization. Logical identity remains separate from its portable bundle, environment binding, resolution lock, and attributable traces. We ground the architecture in vLLM SEMANTIC ROUTER and released bounded multi-model execution studies, then formulate the decisive test: whether constituent complementarity yields gains at fixed active compute across controlled realizations. MOM moves heterogeneous composition from application glue into model infrastructure.

1 INTRODUCTION

The boundary of a useful model no longer coincides with one checkpoint. A deployment may keep latency-sensitive or private work on a robot or edge NPU, draw on specialists in a sovereign data center, and reach a frontier cloud service only when the task warrants it. Open and closed models, accelerator generations, network conditions, prices, and jurisdictions evolve on different schedules. To the caller, however, the system should still behave as one model: one interface, one set of constraints, and one attributable result.

Heterogeneity is also an opportunity for conditional capacity. When constituent errors or resource profiles are complementary, a bounded controller can outperform a fixed choice under a declared objective. A cost-first variant avoids unnecessary work; an accuracy-first variant spends a declared budget on disagreement detection, verification, and synthesis. Security-first and grounding-first variants change which traces are admissible, not merely how they are ranked. Mixture-of-Experts (MoE) established conditional computation within a checkpoint by sparsely activating jointly trained experts (Shazeer et al., 2017). We move the boundary outward. A *Mixture-of-Models* (MoM) is a versioned composite model whose lifecycle engine realizes requests through preference-conditioned, resource-bounded traces over independently versioned constituents. Routing or selection is one possible topology, not the architecture itself. MoM is therefore a form of system-level intelligence: capability is not localized in any constituent, but emerges from a control policy that allocates heterogeneous model and system resources under workload, preference, and deployment state. This view extends the Workload–Router–Pool thesis that workload, control, and physical realization are coupled design variables rather than independent layers (Chen et al., 2026b).

Routers, cascades, ensembles, and agent pipelines already exploit parts of this space (Chen et al., 2023b; Ong et al., 2024; Wang et al., 2024; Kandogan et al., 2024). In these systems, the composite is typically treated and evaluated as a routing, ensemble, or application pipeline rather than versioned and evaluated as one model artifact. For sovereign AI deployments, this boundary is consequential: concrete cloud-sovereignty requirements can make a capable call inadmissible because of residency, audit, or provider control (European Commission, 2026). We therefore place MoM at the model-invocation infrastructure layer, below agent harnesses. Agents retain application goals and tool intent;

a complete lifecycle engine owns admissibility, finite composition, budgets, and attribution. vLLM SEMANTIC ROUTER supplies an initial execution substrate at this boundary. We use it to ground the architecture while proposing the portable identity, binding, and conformance layer needed to build, train, evaluate, exchange, and serve composite models as durable model objects.

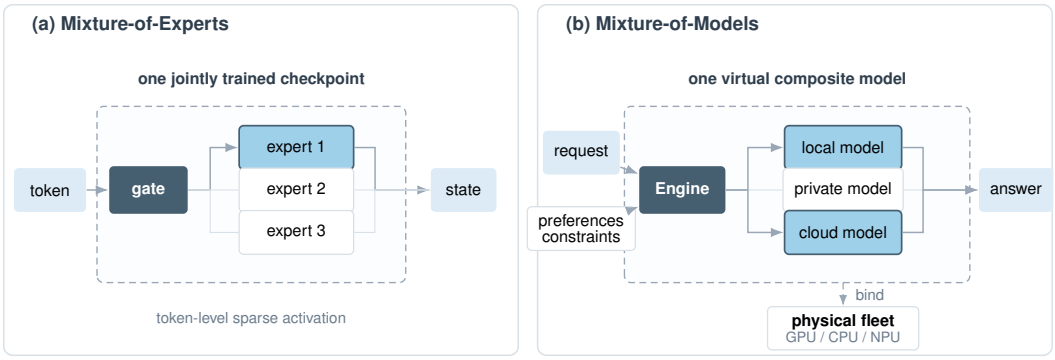

Figure 1: **Conditional computation at two model boundaries.** In MoE, a gate selects experts inside one jointly trained checkpoint. In MoM, an engine selects and composes independently versioned models behind one model interface; a deployment binding maps their logical identities onto a physical fleet. Highlighted nodes show one realized trace.

Crossing the model boundary changes the optimization object. MoM does not require a shared parameter space. Instead, it learns or searches over admissible traces, then promotes changes to control state, pools, topology, contracts, or deployment as explicit revisions. Candidates are evaluated on realized utility and constraint outcomes. Uniform utilization is not an intrinsic goal because constituents differ in capability, price, capacity, and policy eligibility.

This paper makes three contributions:

- We formalize MoM as a preference-conditioned distribution over finite, typed execution traces, separating hard admissibility from soft objectives and defining capacity diagnostics and failure criteria.
- We specify a composite-model format and lifecycle from logical specification to portable bundle, realization profile, environment binding, resolution lock, and attributable execution.
- We ground the execution and systems claims with released multi-model runs and a routing-aware capacity case study, and state the matched-budget tests required to establish conditional scaling.

2 MIXTURE-OF-MODELS AS A MODEL ARCHITECTURE

2.1 ONE MODEL BEYOND ONE CHECKPOINT

In conventional deployments, a model name resolves to one checkpoint or managed endpoint. A Mixture-of-Models (MoM) instead exposes one model interface while computation crosses model, provider, process, and device boundaries. Its components may be open or closed, local or remote, and independently versioned and invoked; they are typically, but need not be, independently trained. Its model identity is defined not by shared weights but by a versioned executable specification that fixes logical references, admissible substitutions, composition, constraints, and the output contract.

Each published model fixes a behavior variant b, its default objective, and an allowed domain $\mathcal{U}_b$ of request preferences. For an input x, conversation or session history h, request preference $u \in \mathcal{U}_b$, and observable environment state e, we define its logical specification as

$$\mathcal{M}_b = (\mathcal{V}, \mathcal{Q}, \mathcal{U}_b, S_\phi, P_\omega, \pi_\psi, \mathcal{A}, \mathcal{G}, \mathcal{K}, \Gamma). \tag{1}$$

Here $\mathcal{V} = \{v_1, \ldots, v_n\}$ contains *logical model references*, not URLs; each declares an exact identity or capability requirement together with interfaces and deployment requirements. Optional $\mathcal{Q}$ contains

declared typed auxiliary operators for retrieval, tool use, and other bounded effects; operators have the same explicit interface, policy, and provenance requirements as model calls, while persistent application state remains outside the model. $\mathcal{U}_b$ contains the typed request preferences and policy refinements admitted by behavior variant b; it cannot override a hard guard or transform one published variant into another. S_ϕ extracts typed signals such as domain, complexity, modality, risk, and confidence; P_ω projects them to calibrated scores and predicates; and π_ψ chooses the next admissible action. $\mathcal{A}$ defines action and transition semantics, $\mathcal{G}$ contains typed preconditions and postconditions, $\mathcal{K}$ defines intermediate and final contracts, and Γ imposes finite bounds on events, path length, fan-out, calls, rounds, tokens, and time. These components may be learned or authored; content-addressed prompts and policy assets parameterize the specification. Catalogs, topologies, prompts, thresholds, contracts, and bounds can be optimized even without gradients through a constituent model.

A deployment binding later resolves a logical requirement to a local checkpoint or provider model. Thus the specification can be inspected and packaged without credentials, then rebound without editing its decision semantics (Section 4). We write $\mathcal{R} = (\mathcal{M}_b, \mathcal{B}, L)$ for a *deployed MoM*, where $\mathcal{B}$ is the environment binding and L is the completed resolution lock, including the engine revision. Logical identity belongs to $\mathcal{M}_b$. The caller observes $\mathcal{R}$ as one virtual model: the engine revision recorded in L executes $\mathcal{M}_b$ against the resolved pool, producing an attributed event DAG for each request. This distinction permits cross-engine portability without reducing the deployed model to either the engine or its endpoints.

2.2 Execution-trace semantics

Before event j, the runtime state is

$$\xi_j = (x, h, u, e, z_j, m_j, b_j), \qquad z_j = P_\omega(S_\phi(x, h, e, m_j), h, u, e, m_j), \tag{2}$$

where m_j is typed working memory and b_j is the remaining resource budget. Recomputing z_j allows later decisions to use validated outputs from earlier events without granting a planner unbounded control. The primitive actions are

$$\begin{aligned} &\textsc{Call}(v, r), \quad \textsc{Op}(q, r), \quad \textsc{Fork}(a_1, \ldots, a_k), \\ &\quad \textsc{Join}(f), \quad \textsc{Check}(k), \quad \textsc{Transform}(f), \quad \textsc{Return}(s, y). \end{aligned} \tag{3}$$

A call invokes v with a checked request; Op invokes a declared, typed external operator q such as retrieval or a tool; fork and join implement bounded parallel composition; check evaluates a guard; and transform is typed and deterministic. Planners and judges are ordinary calls, not sources of unbounded control flow. Every terminal trace ends with a typed status $s \in \{\texttt{ok}, \texttt{abstain}, \texttt{reject}, \texttt{error}\}$; for a non-answer status, $y = \bot$. Let $\bar{y} = (s, y)$ denote the complete terminal outcome.

An execution trace is a finite, labeled event DAG

$$\tau = (N, \prec, \sigma, \{(\xi_j, a_j, o_j)\}_{j \in N}, \bar{y}), \tag{4}$$

where N is the event set, $\prec$ is the control-and-data happens-before relation, and σ is a canonical topological order used only for logging and replay. A sequential trace has a total order; Fork creates incomparable child regions, and Join depends on their terminal events. Stable branch identifiers make σ unique, so different interleavings do not create duplicate traces. The state ξ_j is a deterministic function of the outcomes visible through predecessors of j; wall-clock observations and timeouts remain part of e and o_j. Both $|N|$ and every path length are finite under Γ.

The policy may be stochastic and condition on prior model outputs through m_j. Let L identify the bound constituent revisions, let $C(\tau) \subseteq N$ contain the controller's choice events, and let $E(\tau) \subseteq N$ contain effectful model and operator calls. Graph-prescribed and administrative events have unit probability. For event j, the causal history $H_j = \{o_k : k \prec j\}$ contains only predecessor-visible outcomes; ξ_j is measurable with respect to this history. In particular, a branch cannot observe an incomparable branch before their declared join.

Parallel calls can share provider, load, and runtime randomness, so we do not assume that their outcomes are conditionally independent. Instead, let K_L be the causal execution kernel over all effectful outcomes for the chosen action DAG, and let $r_{E(\tau)}$ collect the checked requests attached to

its effectful actions. The joint probability of a valid trace and terminal outcome is

$$p_{\mathcal{R}}(\bar{y}, \tau \mid x, h, u, e) = \mathbf{1}[\textsc{Return}(\tau) = \bar{y}] \prod_{j \in C(\tau)} \pi_\psi(a_j \mid \xi_j) \cdot K_L\big(o_{E(\tau)} \mid r_{E(\tau)}, a_{E(\tau)}, \prec, x, h, u, e\big)\,. \tag{5}$$

The kernel is causal: an event distribution may depend on H_j and shared runtime state, but not on incomparable or future outcomes. When effectful calls are conditionally independent given their causal histories, K_L reduces to a product of per-call response kernels. The user-facing distribution of the deployed realization marginalizes admissible event DAGs,

$$p_{\mathcal{R}}(\bar{y} \mid x, h, u, e) = \sum_{\tau \in \mathcal{T}_{\text{adm}}} p_{\mathcal{R}}(\bar{y}, \tau \mid x, h, u, e). \tag{6}$$

Closed-model token probabilities need not be accessible: training may use sampled outputs, scores, preferences, or outcomes.

The trace view unifies common structures. **Selection** calls one model; a **cascade** conditionally escalates through an ordered sequence; **fusion** forks a bounded panel and joins its responses; **multi-round collaboration** repeats calls over typed shared state; and a **workflow** instantiates a bounded static graph or validated planner output. Routing is therefore one MoM topology, while a collection of APIs is not a MoM until its admissible traces, contracts, and identity are explicit. Fixed panels and workflows are degenerate policies that assign probability one to one graph; conditional computation does not require every trace to activate a strict subset of the catalog.

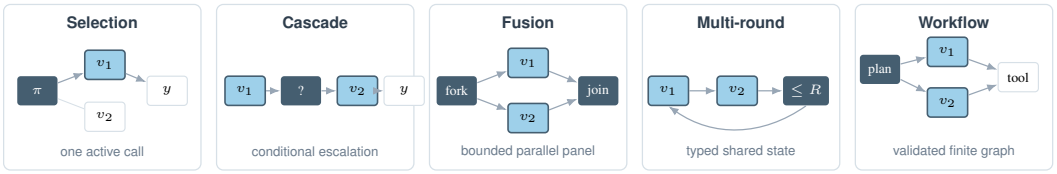

Figure 2: **One trace semantics, multiple execution topologies.** Selection, conditional cascades, bounded parallel fusion, multi-round collaboration, and validated workflows use the same primitive actions and resource bounds. Blue nodes are calls realized in the illustrated trace; white nodes remain admissible but inactive.

Coverage of bounded graphs. The action vocabulary is designed to represent finite, well-typed control-flow graphs whose effectful vertices call members of $\mathcal{V} \cup \mathcal{Q}$, whose branches and joins have declared semantics, and whose cycles have finite bounds. Model and operator vertices map to CALL and OP; branches to CHECK and policy transitions; parallel regions to FORK/JOIN; deterministic vertices to TRANSFORM; and terminals to RETURN. Bounded cycles can be unrolled or represented by a counter in typed memory. For fixed binding, response kernels, and scheduler semantics, the translation preserves typed terminal outcomes and effectful calls up to deterministic administrative steps under these declared assumptions. This is a design property, not a claim that every graph can be learned efficiently. It lets selection, cascades, panels, multi-round protocols, and finite workflows share one runtime action vocabulary.

2.3 PREFERENCES DEFINE OBJECTIVES; POLICIES MUST SATISFY CONSTRAINTS

Let $R(\bar{y}, y^*)$ measure task utility, including the utility of explicit abstention, rejection, or error, and let $c(\tau, e) \in \mathbb{R}^d_+$ contain soft resource outcomes such as monetary cost, latency, generated tokens, energy use, and estimated emissions. A request preference u compiles to weights $\lambda(u)$ and, where appropriate, targets or service-level objectives. For a data distribution $\mathcal{D}$, a basic training objective is

$$\max_{\Theta} \; \mathbb{E}_{(x,h,u,e,y^*)\sim\mathcal{D},\, \tau \sim p_{\mathcal{R}_\Theta}(\cdot \mid x,h,u,e)} \left[R(\bar{y}, y^*) - \lambda(u)^\top c(\tau, e) \right], \tag{7}$$

where $\Theta = (\theta_{\log}, \beta)$ denotes the logical and deployment coordinates introduced in Section 3.

Privacy, residency, model allowlists, and capability rules that are known before a call are typed action preconditions $g^{\text{pre}}_\ell(\xi_j, a) \in \{0, 1, \star\}$, where $\star$ denotes unknown evidence. They define

$$\mathcal{A}_{\text{adm}}(\xi_j) = \{a \in \mathcal{A}(\xi_j) : g^{\text{pre}}_\ell(\xi_j, a) = 1 \; \forall \ell\}, \qquad \pi_\psi(a \mid \xi_j) = 0 \text{ if } a \notin \mathcal{A}_{\text{adm}}(\xi_j). \tag{8}$$

Only a verified value of 1 admits an action. The guards cover graph reachability, artifact policy, environment and binding policy, the selected behavior variant, and request preference. Either may remove actions from the environment-admissible set but cannot restore an action excluded by residency, authorization, or deployment policy. Postconditions capture properties—such as output validity or grounding—that can be evaluated only after an action returns. A trace may return `ok` only if all postconditions on its realized path hold; otherwise it must take a declared repair, fallback, abstention, or rejection edge. Thus $\mathcal{T}_{\mathrm{adm}} \subseteq \mathcal{T}_\Gamma$ contains traces whose actions satisfy Equation 8 and whose failed postconditions follow declared transitions. If no action is admissible, execution fails closed with RETURN(`reject`, $\perp$). Statistical requirements such as tail latency should instead use population constraints such as $\Pr[g_k^{\mathrm{post}} = 0] \le \delta_k$; an unobserved outcome cannot be filtered before execution. These guarantees are relative to the declared evidence and checker semantics; they do not establish confidentiality, safety, or factuality unless those mechanisms are themselves sound.

Published behavior variants such as *cost-first*, *accuracy-first*, *balanced*, *security-first*, and *grounding-first* fix both objective defaults and non-negotiable guards, preventing quality from compensating for a privacy or contract violation. Request preferences may tune declared tradeoffs only within the selected variant's domain.

Two operating points make the distinction precise. With quality floor q, nonnegative resource weights w_c, and expected resource cap $\bar{b}$, the same MoM may be optimized as

$$\begin{aligned} \text{cost-first:} \quad & \min_{\Theta} \ \mathbb{E}[w_c^\top c(\tau, e)] \quad \text{s.t.} \quad \mathbb{E}[R(\bar{y}, y^*)] \ge q, \\ \text{accuracy-first:} \quad & \max_{\Theta} \ \mathbb{E}[R(\bar{y}, y^*)] \quad \text{s.t.} \quad \mathbb{E}[c(\tau, e)] \preceq \bar{b}. \end{aligned} \tag{9}$$

The first uses conditional computation to avoid unnecessary work; the second uses an expected resource envelope to exploit complementary failure modes. Per-trace hard limits remain enforced by Γ. These are first-class behavior variants with separate versioned coordinates, not application-side graphs.

Executor contract. A conforming executor must make the preceding semantics operational. Given a well-typed specification, finite event, fan-out, path, and action-duration bounds, every budget exhaustion, timeout, or stuck state must follow a declared transition, and execution must eventually terminate within Γ with a typed status. The controller's support must remain inside $\mathcal{A}_{\mathrm{adm}}(\xi_j)$, and every successful return must be guarded by its declared postconditions. These obligations yield three checkable runtime invariants: execution terminates within Γ with a typed status; no action with a false declared precondition is issued; and `ok` is reachable only after required checks succeed. They are executor guarantees relative to the declared guards and evidence, not statistical claims about the correctness of those guards.

2.4 ARCHITECTURAL INVARIANTS

Equation 1 imposes five invariants: execution is conditionally selected and resource-bounded; composition is typed; recursion, iteration, and fan-out are bounded; logical semantics are separate from placement; and every response is attributable to a MoM version, binding, and trace. These invariants make a MoM optimizable like a model and operable like a distributed system.

3 OPTIMIZING AND EXECUTING A MIXTURE-OF-MODELS

MoM extends the optimization surface beyond constituent weights to the distribution of computation around independently versioned models. A lifecycle engine can optimize that distribution while preserving one MoM identity across training, evaluation, and serving. Accordingly, MoM optimization is a full-stack co-design problem. Changing the trace policy changes the demand presented to the serving fleet; changing placement, batching, or pool capacity changes the feasible trace distribution. Logical and physical coordinates must therefore be optimized jointly while remaining separately versioned for portability and attribution (Chen et al., 2026b). Figure 3 makes this coupling explicit.

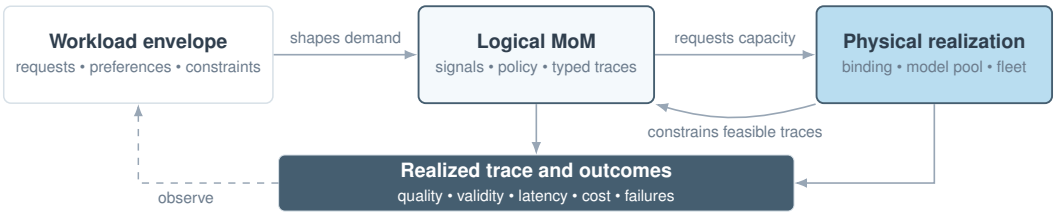

Figure 3: **MoM as system-level intelligence.** Workload, preferences, and constraints shape the useful trace distribution; the logical MoM shapes demand on constituent models; and bindings, placement, and fleet state bound feasible execution. Realized quality, validity, latency, cost, capacity, and failures feed back into both logical optimization and physical planning.

3.1 Training the trace distribution

Let $\mathcal{M}_{\theta_{\log}}$ denote a parameterized logical specification. We write the lifecycle optimization coordinates as

$$\Theta = (\theta_{\log}, \beta), \qquad \theta_{\log} = (\phi, \omega, \psi, \rho, \eta), \qquad \mathcal{B} = \mathcal{B}_\beta, \quad L = \mathrm{Resolve}(\mathcal{M}_{\theta_{\log}}, \mathcal{B}_\beta, e). \tag{10}$$

ϕ parameterizes learned signals, ω their projections, and ψ the action controller. The discrete coordinate ρ selects catalog members, topology, prompts, thresholds, contracts, and resource bounds already represented in $\mathcal{M}$; it does not introduce a second semantic authority. Optional η denotes adapters or constituent weights that the optimizer may update. The deployment coordinate β controls binding, placement, replication, and batching. Promoting $\theta_{\log}$ creates a new logical model version; changing β creates a new binding, after which resolution produces a new lock.

Different coordinates call for different optimizers. Signals and routes can be learned from supervision; rankings and escalation from preferences; thresholds from logged outcomes or bandit feedback; pools and topologies through discrete search; and prompts and budgets through black-box optimization. The resulting policy may be distilled into a small controller. Any adapter or constituent-model fine-tuning must be reported separately.

This definition does not turn every configuration edit into training. A MoM training procedure must identify its experience source, optimization variables, held-out objective, constraints, and produced artifact or binding version. Constituent weights may remain frozen, be tuned independently, or be co-trained when access and interfaces permit. Online adaptation must additionally bound exploration and preserve policy invariants.

For a training set, the engine minimizes task loss and soft resource outcomes subject to the hard admissibility conditions in Equation 8:

$$\begin{aligned} \min_{\Theta} \quad & \frac{1}{N}\sum_{i=1}^{N} \mathbb{E}_{\tau \sim p_{\mathcal{R}_\Theta}(\cdot \mid x_i, h_i, u_i, e_i)} \left[\ell(\bar{y}_i, y_i^*) + \lambda(u_i)^\top c(\tau, e_i) \right] \\ \text{s.t.} \quad & \operatorname{supp} p_{\mathcal{R}_\Theta}(\cdot \mid x_i, h_i, u_i, e_i) \subseteq \mathcal{T}_{\mathrm{adm}} \quad \forall i. \end{aligned} \tag{11}$$

Any allowed statistical violation rate must be stated explicitly and estimated on held-out data; a soft penalty does not enforce an exact privacy or residency rule.

Training produces a candidate version or asset, which evaluation compares with the incumbent under a fixed protocol. Promotion is explicit, and serving never silently rewrites the active specification. Bounded online adaptation remains trace-visible and reversible.

3.2 Capacity is heterogeneous and budgeted

Parameter count is a poor capacity measure: it may be unknown, inactive models consume no generation compute, and similarly sized models can share errors. We distinguish *catalog capacity* (capability coverage), *decision capacity* (complementarity exploitable under a budget), and *serving capacity* (throughput, memory, concurrency, and failure envelope).

For pure selection and a per-example utility $q_i(x)$ for model v_i—with h, u, and e suppressed for notation—two diagnostic ceilings are

$$Q_{\text{single}} = \max_i \mathbb{E}_x[q_i(x)], \qquad Q_{\text{oracle}} = \mathbb{E}_x\left[\max_i q_i(x)\right]. \tag{12}$$

Their gap $\Delta_{\text{comp}} = Q_{\text{oracle}} - Q_{\text{single}}$ measures selection headroom. Let $Q_{\mathcal{M}}$ be the expected utility of a pure-selection MoM on the same population and under the same budget. For a positive gap, controller realization is

$$\eta_{\text{ctrl}} = \frac{Q_{\mathcal{M}} - Q_{\text{single}}}{Q_{\text{oracle}} - Q_{\text{single}}}. \tag{13}$$

Fusion may exceed this oracle because it combines multiple responses; report that excess separately as composition gain. A large catalog provides little decision capacity when constituent errors are strongly correlated.

Capacity scaling requires a predeclared matched active-work envelope over calls, tokens, cost, or latency, with all resource outcomes reported. Increasing catalog size under that envelope tests conditional capacity; increasing catalog and active work only tests a larger system.

Let $\mathcal{F}(\mathcal{V}, \bar{b})$ be the feasible MoM policies over catalog $\mathcal{V}$ under fixed budget $\bar{b}$ and fixed constraints, and let

$$U^*(\mathcal{V}, \bar{b}) = \sup_{\pi' \in \mathcal{F}(\mathcal{V}, \bar{b})} \mathbb{E}[R - \lambda^\top c]. \tag{14}$$

If $\mathcal{V} \subseteq \mathcal{V}'$ and every policy feasible over $\mathcal{V}$ remains feasible over $\mathcal{V}'$, then $U^*(\mathcal{V}', \bar{b}) \geq U^*(\mathcal{V}, \bar{b})$ because a policy over the larger catalog can assign zero probability to every added model. This monotonicity belongs to oracle capacity, not to a learned controller. Realized utility may fall as a catalog grows when controller regret, exploration, or coordination overhead increases faster than the new complementarity. Separating the benefit of added complementarity from the cost of controller regret is the central scaling question.

Let the expected activation of logical model i per request be

$$\Lambda_i = \mathbb{E}_\tau\left[\sum_{j \in N} \mathbf{1}[a_j = \textsc{Call}(v_i, \cdot)]\right]. \tag{15}$$

At request rate ν, $\nu\Lambda_i$ determines its mean call rate. Models differ in quality, price, limits, and capacity, so uniform Λ_i is not a desirable default. Balancing should target the highest feasible utility under capacity and tail-latency constraints, not equal traffic.

3.3 Collapse and failure modes

MoM couples statistical and systems failures. **Pool collapse** is harmful when concentrated traffic excludes a better feasible policy; route entropy alone cannot distinguish such collapse from correctly rejecting dominated models. Diagnose it against Q_{single}, oracle headroom, and realized utility. **False complementarity** arises from leakage, judge bias, or small samples and requires held-out, per-item error correlation. **Policy dead zones** leave reachable signal states without valid traces; static coverage and fail-closed tests expose them.

Hidden escalation underreports cost by omitting retries and fallbacks. **Handoff loss** discards context through truncation, poor prompt translation, or lossy summaries. **Contract collapse** masks invalid outputs through unreported repair. Evaluations should therefore trace every attempt and report first-pass validity, repairs, and terminal failures.

Finally, **provider drift and capability aliasing** break assumed substitutability; locks, probes, and continuous evaluation detect and constrain their effects. **Placement skew** overloads a pool or violates locality. Model-generated plans introduce **unbounded control** unless the engine restricts them to declared references and enforces Γ.

3.4 Compile once, schedule under current state

Before serving, the proposed engine type-checks the graph, validates contracts and bounds, and resolves requirements. Per request it evaluates signals, filters inadmissible actions, applies the

preference-conditioned policy, and schedules the topology. Parallel latency follows the critical path. Timeouts, partial joins, fallbacks, caching, batching, and retries are declared semantics.

The engine emits, alongside each response, a trace linked to the MoM artifact and resolution lock for attribution, replay, accounting, optimization, and incident analysis. When policy permits, privacy-preserving redaction should retain the minimum structural metadata needed to reconstruct the decision.

Reference-engine scope. vLLM Semantic Router grounds this execution contract without assuming access to constituent weights or a jointly differentiable trainer. Section 5 audits the implemented subset and the lifecycle boundary proposed here.

4 The Mixture-of-Models Artifact and Lifecycle

A model architecture needs a model artifact, not merely runtime configuration. For MoM, that artifact is the *bundle*: a typed envelope that gives a composite model one interface, identity, policy, and finite execution semantics while leaving constituent checkpoints in their native formats. The same logical object must remain identifiable across authoring, optimization, evaluation, and deployment. The bundle carries the logical half of the co-design; binding and locking instantiate its physical half, and realized traces provide common evidence for optimizing both. We therefore define six lifecycle stages, separating policy and learned assets from environment-specific endpoints and traces:

$$\text{source} \rightarrow \text{typed IR} \rightarrow \text{bundle} \rightarrow \text{binding} \rightarrow \text{lock} \rightarrow \text{realized trace}. \tag{16}$$

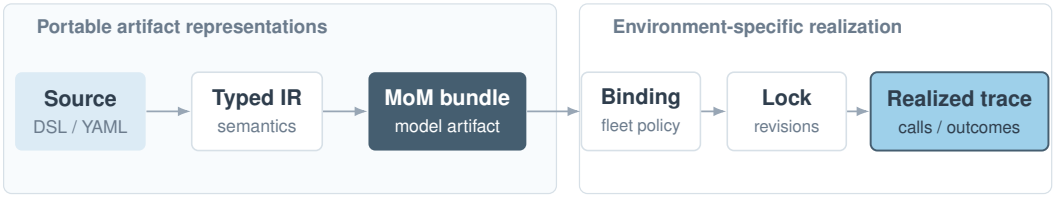

Figure 4: **The MoM lifecycle separates portable representation from environment-specific realization.** The lifecycle engine compiles source into typed IR and an immutable bundle, then binds and resolves logical references into a lock. Every realized trace is attributed to the bundle, binding, and lock; credentials and live health remain outside the portable artifact.

4.1 Six stages, six responsibilities

The following definitions are normative requirements of the proposed format, not a description of the current vLLM Semantic Router artifact format.

Authoring source. Source is a human-readable DSL or YAML with comments, fragments, and defaults. Sources that normalize to the same IR define the same logical model.

Canonical typed IR. The IR is the semantic authority. It declares schemas and entry points, logical model requirements, signals and projections, action graphs, extension interfaces, contracts, behavior variants, request-preference domains, and bounds. It contains neither credentials nor implicit defaults. Canonical maps, numbers, and asset references enable stable hashes and semantic diffs, and unknown fields are rejected. Model requirements have no competing manifest authority; hard constraints are typed fail-closed predicates; and every contract check names explicit success and failure successors.

Portable bundle. The bundle is an immutable distribution unit around the IR:

```
manifest.json            interface, content index, compatibility
mom.ir.json              canonical executable specification
source/ assets/          authoring sources and semantic assets
contracts/ eval/         schemas and evaluation protocols
targets/ MODEL_CARD.md   realization templates and documentation
```

Source is optional, allowing an author to export the executable object without revealing its authoring representation. Leaf artifacts may be safetensors, ONNX, GGUF, or immutable repository objects; closed constituents remain typed provider references. Large weights may be external but content-addressed. Plugin code is not implicitly trusted: the bundle declares an operator set, interface, effects, and digest; deployment policy allows or rejects it.

A *behavior variant* such as accuracy-first or security-first belongs to logical semantics and therefore to model identity; it declares the default and allowed domain of request preferences. A *realization profile* such as CUDA, ROCm, CPU, edge, or hybrid instead declares portable runtime and accelerator requirements. It may be distributed with the bundle, but it contains no endpoint, secret, device identifier, or live capacity and does not change logical identity.

Backend compatibility is not a scalar `backend` choice. A binding keeps six axes distinct: wire transport, API adapter, model runtime, leaf-artifact format, accelerator, and deployment driver. Thus one admissible trace may use a ROCm-hosted vLLM model, a CUDA specialist, a CPU verifier, and a managed cloud API without assigning the composite model itself a single hardware backend.

Deployment binding. A binding is an environment-owned mapping

$$\mathcal{B} : \mathcal{V} \cup \mathcal{Q} \to 2^{\mathcal{E}} \tag{17}$$

from logical model or operator references to candidate endpoints or local deployments under a selected realization profile. For $r \in \mathcal{V} \cup \mathcal{Q}$, each $d \in \mathcal{B}(r)$ must satisfy modality, context, capability, data-policy, and interface requirements; operator candidates also satisfy their declared effect and sandbox requirements. Bindings declare placement, replicas, limits, prices, and fallbacks, but contain only secret references. Each candidate exposes its capability evidence, capacity envelope, and pricing assumptions, so resolution cannot infer them silently from an endpoint name. Here $\mathcal{E}$ is the environment's set of candidate deployments.

Resolution lock. Resolution lock L records the complete resolved candidate inventory: model, tokenizer, adapter, image, runtime, accelerator, protocol-adapter, asset, and plugin identities plus capability evidence. Binding says what may run; lock says what was resolved; the trace says what actually ran. Revision evidence is typed as *pinned* when an immutable content or repository identifier can be recorded and *observed-opaque* when a provider exposes only an alias. The latter retains an observation time and a content-addressed probe record without claiming bitwise reproducibility. Floating references are a development mode, not a production lock.

Realized trace. The trace records semantic/bundle/binding/lock digests, behavior variant, request preference, realization profile, constraint checks, signals, actions, resolved calls, timing, tokens, usage, accounting version, realized cost, retries, contracts, and terminal status. Content may be encrypted, redacted, hashed, or referenced under data policy. A trace is an observation, not model definition.

4.2 Identity and portability

A human coordinate such as `modelId:version` is not an integrity identity. We instead separate four content identities. The *semantic digest* commits to canonical IR and its path-sorted semantic assets. The *bundle digest* additionally commits to the manifest, source, contracts, evaluation protocols, target templates, and documentation. The *binding digest* identifies canonical environment policy, while the *lock digest* identifies one completed resolution. The manifest stores the semantic digest; the external lock stores bundle and binding digests; the lock digest is computed only after the lock is complete. No object therefore contains its own hash.

Bindings, credentials, and traces are excluded from logical identity. Rebinding realizes the same logical model but produces new binding and lock identities; changing a controller, prompt, contract, requirement, behavior variant, or topology changes the semantic digest. Changing only source formatting, a model card, or a realization template preserves logical identity but changes the distributed bundle. Every result carries all four digests. Together they distinguish semantic, distribution, environment, and realized-resolution equivalence.

Portability preserves logical and control semantics, not predictive equivalence across substitutions. It is tested by fixing semantic and bundle digests, resolving them in two environments, and reporting

quality, validity, cost, and latency under both binding and lock identities. Capability substitutions are explicit and disclosed; two edited recipes are not portable merely because they share an API path.

4.3 Build, import, validation, and export

Build expands source defaults into canonical IR and performs four classes of checks: schema and type validity; reference, reachability, and bound correctness; well-formed contract branches and statically decidable constraint consistency; and path, digest, and license-record integrity. Explicit migration handles older source versions; runtime evidence resolves guards that cannot be decided statically. Import instead verifies an already built artifact, records its content identity, and never recompiles an optional source representation. At deployment, every reachable call needs an admissible binding or a declared fallback or abstention path.

Export writes canonical IR, a deterministic manifest, content-addressed assets, and evaluation protocols—never credentials, endpoint health, online counters, or private traces. It may produce a thin artifact of pinned references, a materialized artifact containing redistributable leaf objects, or a targeted artifact with selected runtime objects. Materialization fails rather than pretending that a closed provider is portable. Source round-tripping preserves semantics, not formatting. IR diff reports behavior changes such as a widened candidate set or relaxed residency guard.

Consumers verify digests, schemas, operator sets, and engine capabilities; deployment policy enforces plugin and license rules and probes model access. Evaluation results, build provenance, SBOMs, and signatures are digest-bound attestations outside the bundle, so new evidence does not redefine the evaluated model.

This boundary follows useful precedents without reducing MoM to any one of them. Hugging Face repositories motivate a save/load model coordinate and immutable revision pinning; MLflow motivates signatures shared by loading, serving, and evaluation; ONNX separates a versioned logical IR from capability-negotiated execution providers; and OCI supplies content-addressed descriptors and distribution primitives (Hugging Face, 2026; MLflow Project, 2026; ONNX Project, 2026; Microsoft, 2026; Open Container Initiative, 2025). MoM applies this separation at the model-call level: an accelerator or runtime realizes the model but does not define it.

4.4 One lifecycle across training, evaluation, and inference

An optimizer consumes data or traces from a deployed realization $(\mathcal{M}_k, \mathcal{B}_k, L_k)$ and emits a complete logical or binding candidate, not an opaque mutation. Under recorded resolution locks, evaluation compares candidate and incumbent on quality, violations, cost, latency, retries, and uncertainty before promotion. Serving emits traces, monitoring detects drift, replay reconstructs decisions, and approved traces feed offline optimization. Immutable local revisions support revision-pinned re-execution; stochastic sampling and runtime scheduling still require recorded seeds and events. Opaque provider aliases support attributed re-evaluation and drift measurement, not bitwise reproduction.

Online adaptation may update trace-linked local state, but the model changes only when that state is promoted into a hashed asset or IR revision. Session protection and fallback remain runtime state, preventing one replica's traffic from silently redefining another's artifact.

5 An Execution Substrate for Mixture-of-Models

vLLM Semantic Router[1] provides an initial execution substrate for MoM. Current releases expose selection and bounded multi-model policies through standard model APIs, driven by typed request analysis and declared model references, with evaluation and replay hooks around the resulting decision. This demonstrates that heterogeneous model identities and multi-call topologies can execute behind one model invocation rather than inside an agent harness. The portable bundle, environment-owned binding and lock, topology-neutral interpreter, and canonical event DAG proposed in Section 4 remain lifecycle extensions rather than shipped features.

[1] Open-source implementation: https://github.com/vllm-project/semantic-router.

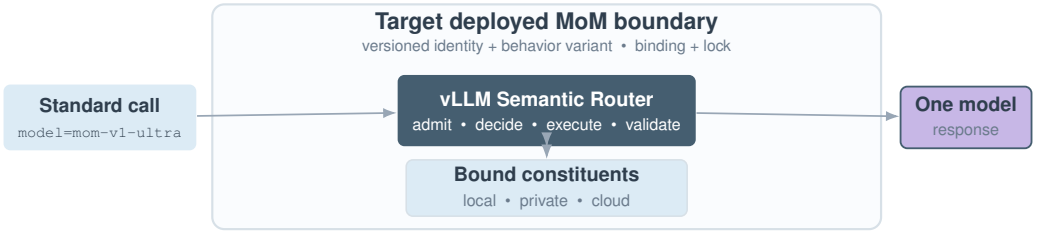

Figure 5: **The model-invocation boundary of a deployed MoM.** A caller issues one standard model request and receives one response, while the engine admits, composes, and observes bounded calls to bound constituents. Current provider mappings resolve configured names to backend targets. Portable identity and environment-specific resolution complete the proposed lifecycle boundary.

5.1 Why the control point is below agents

An agent runtime owns application semantics: goals, task decomposition, interaction state, and the intent to invoke a model. A MoM engine owns a different contract: whether model-level computation is admissible, how it is resolved and scheduled, which finite budget it may consume, and how the result is attributed. The model-invocation boundary is the narrowest common seam at which this contract can cover direct inference, retrieval-augmented generation, agent loops, batch jobs, and embedded systems without depending on one agent framework.

At this seam, a complete engine can enforce authorization and residency, resolve logical references against deployment evidence, reserve budgets, choose legal fallbacks, and account for every attempt. Current vLLM Semantic Router implements only part of that contract; the full admissibility and attribution boundary belongs to the proposed MoM format. When deployed as the exclusive model ingress, this placement prevents direct calls from bypassing policy and avoids reproducing governance logic in every agent harness.

The boundary remains closed-world: one invocation must use only declared models and operators, consume finite resources, and return one terminal response. An agent may invoke a MoM or request a bounded workflow, but persistent goals, task queues, and application state remain outside the logical model. Agents are clients of the model infrastructure, not its portability or accounting boundary.

Figure 6 shows why this placement matters. One logical MoM can retain its interface, graph, preferences, and constraints while different bindings realize it on a robot, at an edge site, in a sovereign data center, or through managed cloud services.

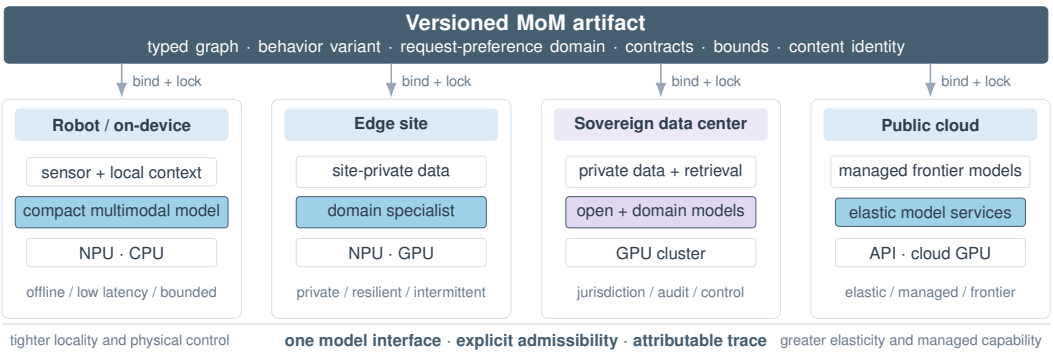

Figure 6: **One logical MoM, multiple target deployment envelopes.** Environment-owned bindings realize the same versioned specification using on-device models, an edge specialist, a jurisdiction-bound private pool, or managed cloud capabilities. Portability preserves logical and control semantics, not identical outputs; every substitution and measured outcome remains explicit behind one model interface.

5.2 From routing engine to composite-model lifecycle

The implementation already separates routing-owned semantics from provider access details. Configured model references participate in decisions and bounded multi-call policies, while provider entries map those names to concrete backends. This is sufficient to execute declared composites, but not for the full MoM admissibility contract. The proposed binder adds an environment-owned, fail-closed check that every realized call satisfies deployment evidence and hard constraints before each call is issued. The change is a lifecycle boundary, not a claim that a larger configuration alone constitutes a model.

The distinction is concrete for accelerators. Current platform selection governs the router's own execution image—CPU by default, or an AMD/ROCm or NVIDIA/CUDA variant—rather than provisioning constituent LLM endpoints. In the proposed lifecycle, accelerator choice is resolved per target: one trace may combine a ROCm-served local model, a CUDA specialist, a CPU verifier, and a managed provider. Wire transport, API adapter, runtime, leaf artifact, accelerator, and deployment driver are independent compatibility dimensions. Existing endpoints are the first binding driver; managed deployment follows behind the same resolver interface.

The user-facing consequence is intentionally simple. Current releases can expose configured routed and multi-call policies through ordinary model slugs such as `vllm-sr/auto`, `vllm-sr/fusion`, `vllm-sr/remom`, and `vllm-sr/flow`. The coordinates in Table 1 illustrate the next step: a registry entry would pin the entire logical MoM and its behavior variant, just as a model version pins a conventional serving target.

Table 1: Illustrative versioned MoM identities exposed through the standard model field.

Model coordinate	Behavior variant	Default objective or guard
`vllm-sr/mom-v1-flash:1.0.0`	Speed-first	Minimize expected latency
`vllm-sr/mom-v1-light:1.0.0`	Cost-first	Minimize expected cost above a quality floor
`vllm-sr/mom-v1-ultra:1.0.0`	Accuracy-first	Maximize utility under a declared budget
`vllm-sr/mom-v1-halu:1.0.0`	Grounding-first	Require evidence checks and fail-closed fallback
`vllm-sr/mom-v1-secu:1.0.0`	Security-first	Require jailbreak and PII checks

To a client, these identities remain values of the standard `model` field and return one response. Internally, their policies may select, escalate, deliberate in parallel, or execute a validated bounded workflow. The project also supplies the lifecycle hooks needed to study this behavior as one object: control policies can be trained without gradients through constituents, complete paths can be evaluated through the serving interface, replay associates requests with decisions and outcomes, and an offline planner converts routed demand into capacity alternatives. These surfaces are currently connected operationally rather than by one content identity. The proposed bundle, lock, and event schema make that identity the unit of promotion, evaluation, replay, and cross-engine conformance.

6 Empirical Grounding and Validation Protocol

We ground two architectural claims with released evidence. Task studies show that bounded panel-and-synthesis policies can execute behind one model interface. A separate planning case study estimates how routing structure changes fleet capacity. These studies ground the execution and systems claims; they do not establish the matched-budget scaling hypothesis posed in Section 6.3.

6.1 End-to-end multi-model execution

The current public evaluation manifest[2] records summaries of the GPQA-Diamond and LiveCodeBench runs in Table 2. Each invokes `vllm-sr/auto` through the standard API `model`

[2]Versioned evaluation manifest: github.com/vllm-project/semantic-router.

field, but uses a benchmark-specific constituent pool, signal policy, topology, and response contract. The alias is therefore a serving surface, not an immutable MoM identity. Router-side normalization is applied where configured, and benchmark-native scoring treats failed or missing items as incorrect. The tasks are GPQA-Diamond (Rein et al., 2023) and the January–April 2025 slice of LiveCodeBench (Jain et al., 2024).

Accuracy-first composition. The two configurations spend a finite panel and synthesis budget rather than minimizing active calls. They demonstrate an accuracy-first operating mode in which the served object includes coordination and format handling, not merely the final constituent call. Because the public benchmarks predate the evaluated 2026 constituents, these results characterize end-to-end system execution and should not be read as contamination-free generalization.

Table 2: Project-reported end-to-end runs in the versioned evaluation manifest (vLLM Semantic Router Team, 2026). Scores map each task's native metric to a 0–100 scale; workloads and active-call budgets differ across rows.

Evaluation	Executed topology	Metric	Score (95% Wilson CI)	n
GPQA-Diamond, formal-50	Three parallel reasoning calls followed by synthesis	Accuracy	96.0 [86.5, 98.9]	50
LiveCodeBench v6, Jan.–Apr. 2025	Risk-conditioned panel totaling four or six calls, including synthesis	Pass@1	92.6 [87.7, 95.6]	175

GPQA is the formal-50 run. LiveCodeBench retains a 175-item denominator and counts five missing generations as failures. Wilson intervals reflect per-item sampling uncertainty, not protocol, benchmark-exposure, or model stochasticity.

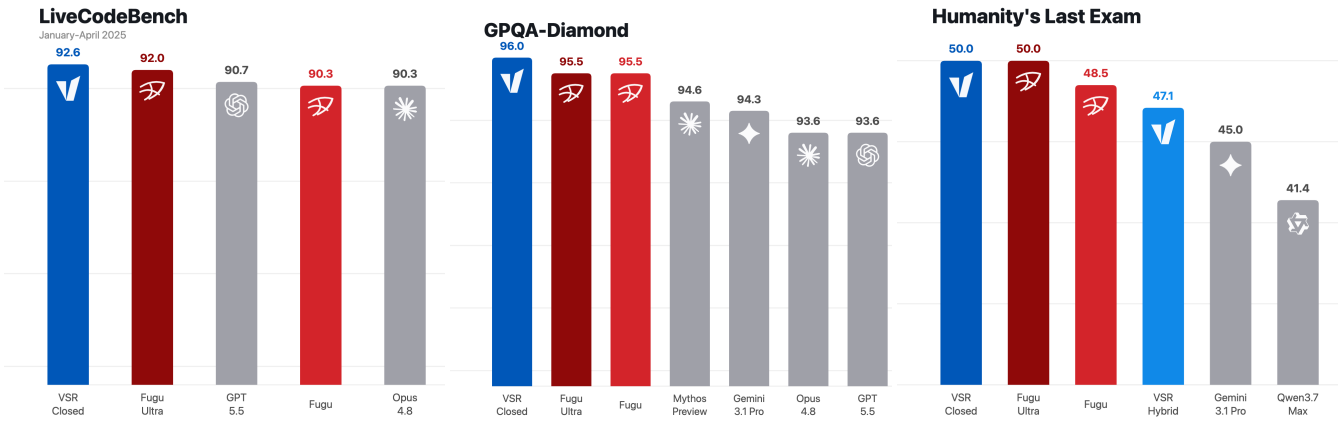

Figure 7: **Leading entries from the released accuracy-first MoM scorecards.** *VSR Closed* denotes a realization using closed constituents; *VSR Hybrid* combines open and closed constituents. GPQA uses the formal-50 slice, LiveCodeBench the 175-item January–April 2025 slice, and HLE a 2,158-item text-only filter. These are protocol-specific release snapshots, not a matched-budget comparison (vLLM Semantic Router Team, 2026; Tang et al., 2026b).

These runs exercise bounded deliberation and synthesis rather than the entire action algebra. Looper metadata records models used and call counts. Router Replay and usage records provide aggregate decision, token, and cost data, while runtime metrics provide latency observability. The canonical per-call event trace required for strict matched-budget comparison is part of the proposed evaluation contract.

Figure 7 reproduces the broader scorecard context in which the results were released. It also includes prior project-reported HLE closed and hybrid aggregates over a 2,158-item text-only filter (Center for AI Safety et al., 2026). Those HLE rows are retained as a released system snapshot; the current versioned evaluation manifest covers GPQA and LiveCodeBench. Public reference bars

use source-specific protocols and provide context, not a matched-budget baseline. In particular, the LiveCodeBench and text-only HLE comparators are pinned to the 175-item and 2,158-item protocols reported with Fugu v1 rather than the later full-benchmark revision (Tang et al., 2026b). The figure therefore illustrates that MoM admits an accuracy-first operating point; it is not the controlled scaling comparison.

6.2 Routing-aware fleet planning

Logical routing changes the workload presented to the serving fleet. To illustrate the connection, an analytical two-pool planner replays the request-length distribution from LMSYS-Chat-1M, a real-world conversation dataset collected partly through Chatbot Arena (Zheng et al., 2023a; Chiang et al., 2024), at 100 requests per second. The homogeneous estimate assigns all traffic to a 65K-context 70B profile and yields 14 A100 GPU-equivalents of continuous capacity. A length-aware split between 4K- and 65K-context pools estimates 8 A100 GPU-equivalents under the same 500 ms time-to-first-token target. At the planner's assumed $2.21 per GPU-hour, annualized cost changes from approximately $271K to $155K, a 42.9% simulated reduction.

This case study instantiates Workload–Router–Pool coupling: the trace policy shapes the demand seen by each pool, so capacity planning cannot be an afterthought to model selection (Chen et al., 2026b). Once routing probabilities and context requirements are explicit, placement and capacity become optimization variables rather than hidden endpoint properties. The numbers are analytical capacity estimates, not a production-cluster measurement; deployment studies must calibrate the planner and repeat the comparison across hardware, providers, and failure conditions.

6.3 The controlled scaling test

The controlled scaling experiment fixes a benchmark and constructs nested pools of increasing size and capability diversity. At each size, it compares the strongest constituent, learned selection, a fixed panel, an adaptive MoM, and an oracle selector under matched call, token, cost, or latency budgets. It reports oracle headroom, controller realization, per-item error correlation, topology activation, contract failures, and repeated-run uncertainty. This design separates conditional capacity from the effect of spending more inference compute. Section 8 extends the test to portability, adaptation, sovereignty, and governance. Figure 8 summarizes the comparison.

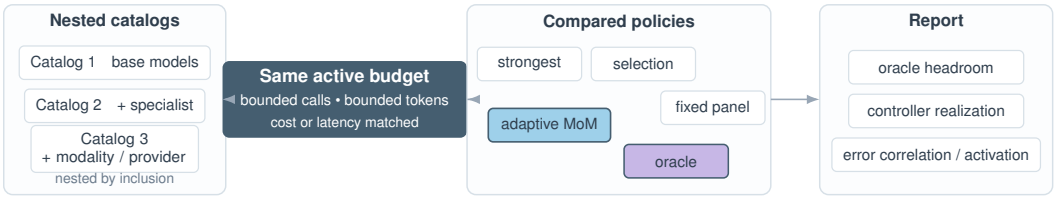

Figure 8: **Matched-budget protocol for MoM scaling.** Nested catalogs are evaluated under one active-work envelope. Comparing a strongest constituent, learned selection, a fixed panel, an adaptive MoM, and an oracle separates catalog complementarity from controller realization and additional inference work. The diagram specifies an evaluation protocol, not a result.

7 Related Architectures

Conditional computation inside a model. Adaptive mixtures and conditional-computation methods established the use of a learned gate to allocate examples to specialized computation (Jacobs et al., 1991; Bengio et al., 2013). Sparsely gated Mixture-of-Experts (MoE), GShard, and Switch Transformers scale that principle while activating only a subset of parameters for each input (Shazeer et al., 2017; Lepikhin et al., 2020; Fedus et al., 2021). Their experts typically share a checkpoint, training loop, tensor interface, and deployment boundary. MoM retains conditional computation but moves the boundary outward: a constituent can be an independently versioned local model, a private specialist, or a closed remote service. MoM does not require joint differentiability and often

forgoes it; in return, it permits independent evolution, heterogeneous placement, and explicit policy and failure boundaries.

Cascades and model routing. Cost-aware cascades and learned routers choose among models under quality and resource objectives (Aggarwal et al., 2023; Chen et al., 2023b; Ding et al., 2024; Ong et al., 2024); RouterBench supplies a routing evaluation framework (Hu et al., 2024). ModelGLUE instead studies model selection and parameter or output aggregation over a model zoo (Zhao et al., 2024). These methods provide important controllers and protocols. A general MoM also admits parallel calls, verification, synthesis, and bounded workflows, and places contracts and deployment identity inside the model lifecycle.

Closer whole-model precedents route among domain specialists, train asynchronous language-model mixtures, form domain-specific consensus, or orchestrate models and agents (Simonds et al., 2024; Filippova et al., 2025; Wang et al., 2025; Guo et al., 2025; Pecerskis & Smirnovs, 2026). Brick and NSED independently use the Mixture-of-Models name; HyDRA and RouteBalance study closely related capability routing and serving-aware scheduling (Massa & Cristofanilli, 2026; Garg et al., 2026; Da & Kalyvianaki, 2026). Fugu is a particularly close contemporary system: trained orchestrator models dynamically devise agentic scaffolds over a model team and expose the result through a model-like interface (Tang et al., 2026a). Our focus is the open infrastructure boundary around such behavior—a finite execution algebra, portable identity, environment binding and lock, attributable traces, and one lifecycle across training, evaluation, and serving.

Ensembles and inference-time collaboration. Ensembling, response fusion, and Mixture-of-Agents exploit complementary outputs from several models (Jiang et al., 2023; Wang et al., 2024; Li et al., 2025). The cited methods evaluate a fixed composition topology. In our formulation, parallel deliberation is one topology in a larger trace space, and a controller may choose another under explicit resource and policy constraints. The trace also makes its additional inference compute visible.

Compound AI systems and language-model programming. Compound AI systems frame application intelligence as the interaction of models, retrieval, tools, and control logic (Kandogan et al., 2024). Language-model programming systems likewise provide constraints, control flow, optimization, and efficient execution for multi-call programs (Beurer-Kellner et al., 2022; Khattab et al., 2023; Zheng et al., 2023b). MoM narrows that broad systems view into a model-facing abstraction with a bounded graph, explicit optimization variables, and portable identity. An agent can invoke a MoM or request a bounded workflow, while the infrastructure layer retains authority over admissibility and accounting. Programming systems can be authoring frontends or execution backends; they do not by themselves define the deployment identity proposed here.

Serving graphs, model packages, and runtimes. KServe InferenceGraph offers declarative sequence, switch, ensemble, and splitter nodes behind one endpoint (KServe Contributors, 2026). Hugging Face model repositories offer revisioned model coordinates; MLflow packages signatures and loading flavors; and OCI manifests provide content-addressed distribution (Hugging Face, 2026; MLflow Project, 2026; Open Container Initiative, 2025). ONNX most clearly separates a typed model IR from capability-negotiated execution providers (ONNX Project, 2026; Microsoft, 2026), while GGUF exemplifies a self-describing materialized edge artifact (GGML Project, 2026). None alone defines a composite model whose constituents may also be closed services. The proposed lifecycle retains native leaf formats while adding a preference-conditioned trace distribution, hard admissibility, separate logical and physical identities, and trace-linked evidence. Serving runtimes address complementary memory management, model multiplexing, scheduling and migration, and multi-tenant adapter serving; examples include vLLM, MuxServe, Llumnix, and Punica (Kwon et al., 2023; Duan et al., 2024; Sun et al., 2024; Chen et al., 2023a). A MoM artifact can bind to these systems without embedding mutable endpoints or credentials.

Evolution of vLLM Semantic Router. Prior project papers developed signal-driven routing, workload–router–pool optimization, a routing policy DSL, and agent orchestration (Liu et al., 2026a; Chen et al., 2026b; Liu et al., 2026b; Chen et al., 2026a). The earliest of these used "MoM" for mixture-of-modality deployments; here, modality is one dimension of the broader catalog of independently versioned models. Those mechanisms remain components of the reference engine. This paper changes the unit of analysis from a routing decision to a versioned logical model whose

learned and authored state spans a pool, a control policy, an execution topology, and contracts, together with an explicit deployment binding.

8 Limitations and Research Agenda

The architecture covers finite, typed, resource-bounded graphs; it does not make unbounded agency safe or guarantee a feasible binding. Mutable closed services limit exact replay, the current system results do not establish the matched-budget scaling hypothesis, and cross-provider or CPU–GPU–NPU portability still requires controlled validation. Security, sovereignty, and grounding are only as reliable as their evidence, enforcement, and checkers. Within those boundaries, the research agenda rests on four claims with concrete validation criteria.

Useful scale is complementarity at fixed active compute. A larger pool matters only when it adds capability, failure, resource, or deployment diversity that a learned policy can exploit within the same call, token, cost, energy, or latency budget. Matched-budget studies must therefore report the strongest constituent, oracle headroom, controller realization, repeated runs, error correlations, and topology ablations as specified in Section 6.3. Parallel panels and additional rounds count as active compute, not hidden model capacity. The scaling curve of interest is attainable utility against catalog diversity and active work; merely spending more inference is not conditional scaling.

Logical model identity must survive physical change. If a MoM is to move between a robot, an edge site, a sovereign cluster, and a managed service, its semantics cannot be identified with any one endpoint or runtime. The bundle therefore implies a shared conformance suite: canonicalization, signatures, operator capability negotiation, compatibility tests, and independent engine implementations. A binder and resolver may rebind logical references across CPU, GPU, NPU, and provider environments, but every substitution must remain visible in the binding, lock, and evaluation. This is especially important for mutable services whose weights, safety layers, prices, limits, or aliases can change without exposing a checkpoint hash; a lock records the best available evidence, while canaries and explicit revalidation measure the remaining drift.

Lifecycle optimization must coordinate logical and physical state. Signals, control policies, pools, prompts, topologies, contracts, and placements are optimized from different evidence and evolve on different timescales. Improving one in isolation can destabilize another: a more aggressive gate changes fleet load, a new binding changes latency and fallback behavior, and a new judge can change the apparent reward. MoM lifecycle optimization therefore needs artifact- and binding-level promotion, chronological evaluation, invariants, and rollback (Chen et al., 2026b). The research target is not convergence of one controller, but stable improvement of realized utility and constraint satisfaction under deployment drift.

Sovereignty and governance are properties of the trace. Sovereignty is not a binary choice between a local model and a public cloud. Real deployments combine devices, jurisdiction-bound data centers, regional providers, open weights, and selectively approved global services. Their requirements span data and model control, residency, encryption-key control, operations, supply chain, audit, and exit paths (European Commission, 2026). A MoM compiler should translate such a profile into typed admissibility constraints over models, operators, data paths, and bindings; cloud fallback must never silently override them. The same principle applies to security, grounding, and preference governance. Signals can guide execution but do not prove confidentiality or factuality, and aggregate utility can hide systematically poor service for a group. Least-privilege operators, secret separation, explicit boundary crossings, per-group constraints, and worst-case evaluation make governance inspectable on the realized trace rather than aspirational at the endpoint.

Progress on these four claims would turn model composition from a collection of recipes into a model ecosystem with measurable scaling behavior, portable identity, and a durable lifecycle.

9 Conclusion

Mixture-of-Models moves conditional computation from parameters within one checkpoint to execution across independently versioned models. We define the served object as a preference-

conditioned distribution over finite, typed traces, with hard admissibility, portable logical identity, and environment-specific binding. Selection, cascades, fusion, multi-round collaboration, and bounded workflows are model topologies rather than application glue. The resulting intelligence is system-level: it arises from co-designing the trace distribution with its physical realization, not from any constituent or control component in isolation.

vLLM SEMANTIC ROUTER shows that an execution-facing subset of this abstraction can be declarative and executable without joint access to constituent weights. The portable bundle, lock, and conformance suite define its next implementation boundary. The central empirical question is whether added model diversity increases utility under matched active compute. If it does, model runtimes will need to execute and evolve composite models across providers, jurisdictions, devices, and fleets as reliably as they execute checkpoints across machines. The resulting mission is to make virtual composite models buildable, evaluable, portable, and servable with the same rigor as checkpoints.

REFERENCES

Pranjal Aggarwal, Aman Madaan, Ankit Anand, Srividya Pranavi Potharaju, Swaroop Mishra, Pei Zhou, Aditya Gupta, Dheeraj Rajagopal, Karthik Kappaganthu, Yiming Yang, Shyam Upadhyay, Manaal Faruqui, and Mausam. AutoMix: Automatically mixing language models, 2023. URL https://arxiv.org/abs/2310.12963.

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation, 2013. URL https://arxiv.org/abs/1308.3432.

Luca Beurer-Kellner, Marc Fischer, and Martin Vechev. Prompting is programming: A query language for large language models, 2022. URL https://doi.org/10.48550/arXiv.2212.06094.

Center for AI Safety, Scale AI, and HLE Contributors Consortium. A benchmark of expert-level academic questions to assess AI capabilities. *Nature*, 649:1139–1146, 2026. doi: 10.1038/s41586-025-09962-4. URL https://doi.org/10.1038/s41586-025-09962-4.

Huamin Chen, Xunzhuo Liu, Bowei He, and Xue Liu. From inference routing to agent orchestration: Declarative policy compilation with cross-layer verification, 2026a. URL https://arxiv.org/abs/2603.27299.

Huamin Chen, Xunzhuo Liu, Bowei He, Fuyuan Lyu, Yankai Chen, Xue Liu, Yuhan Liu, and Junchen Jiang. The workload-router-pool architecture for LLM inference optimization: A vision paper from the vLLM Semantic Router project, 2026b. URL https://arxiv.org/abs/2603.21354.

Lequn Chen, Zihao Ye, Yongji Wu, Danyang Zhuo, Luis Ceze, and Arvind Krishnamurthy. Punica: Multi-tenant LoRA serving, 2023a. URL https://arxiv.org/abs/2310.18547.

Lingjiao Chen, Matei Zaharia, and James Zou. FrugalGPT: How to use large language models while reducing cost and improving performance, 2023b. URL https://arxiv.org/abs/2305.05176.

Wei-Lin Chiang, Lianmin Zheng, Ying Sheng, Anastasios Nikolas Angelopoulos, Tianle Li, Dacheng Li, Hao Zhang, Banghua Zhu, Michael Jordan, Joseph E. Gonzalez, and Ion Stoica. Chatbot arena: An open platform for evaluating LLMs by human preference, 2024. URL https://arxiv.org/abs/2403.04132.

Wei Da and Evangelia Kalyvianaki. RouteBalance: Fused model routing and load balancing for heterogeneous LLM serving, 2026. URL https://arxiv.org/abs/2606.17949.

Dujian Ding, Ankur Mallick, Chi Wang, Robert Sim, Subhabrata Mukherjee, Victor Rühle, Laks V. S. Lakshmanan, and Ahmed Hassan Awadallah. Hybrid LLM: Cost-efficient and quality-aware query routing, 2024. URL https://arxiv.org/abs/2404.14618.

Jiangfei Duan, Runyu Lu, Haojie Duanmu, Xiuhong Li, Xingcheng Zhang, Dahua Lin, Ion Stoica, and Hao Zhang. MuxServe: Flexible spatial-temporal multiplexing for multiple LLM serving, 2024. URL https://arxiv.org/abs/2404.02015.

European Commission. Cloud sovereignty framework: Implementation guidance, June 2026. URL https://commission.europa.eu/document/download/2ad80a48-166f-4c77-a513-80c53ca2a128_en. Directorate-General for Digital Services.

William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity, 2021. URL https://arxiv.org/abs/2101.03961.

Anastasiia Filippova, Angelos Katharopoulos, David Grangier, and Ronan Collobert. No need to talk: Asynchronous mixture of language models. In *International Conference on Learning Representations*, 2025. doi: 10.48550/arXiv.2410.03529. URL https://openreview.net/forum?id=pHOH8FVrTp.

Aashna Garg, Siddharth Singha Roy, Jinu Jang, Federico Brancasi, and Shengyu Fu. HyDRA: Hybrid dynamic routing architecture for heterogeneous LLM pools, 2026. URL https://arxiv.org/abs/2605.17106.

GGML Project. GGUF Specification, 2026. URL https://github.com/ggml-org/ggml/blob/af97976c7810cdabb1863172f31c432dab767de7/docs/gguf.md. Specification at commit af97976c, accessed July 16, 2026.

Xiyu Guo, Shan Wang, Chunfang Ji, Xuefeng Zhao, Wenhao Xi, Yaoyao Liu, Qinglan Li, Chao Deng, and Junlan Feng. Towards generalized routing: Model and agent orchestration for adaptive and efficient inference, 2025. URL https://doi.org/10.48550/arXiv.2509.07571.

Qitian Jason Hu, Jacob Bieker, Xiuyu Li, Nan Jiang, Benjamin Keigwin, Gaurav Ranganath, Kurt Keutzer, and Shriyash Kaustubh Upadhyay. RouterBench: A benchmark for multi-LLM routing system, 2024. URL https://arxiv.org/abs/2403.12031.

Hugging Face. Models on the Hugging Face Hub, 2026. URL https://huggingface.co/docs/hub/models. Official documentation, accessed July 16, 2026.

Robert A. Jacobs, Michael I. Jordan, Steven J. Nowlan, and Geoffrey E. Hinton. Adaptive mixtures of local experts. *Neural Computation*, 3(1):79–87, 1991. doi: 10.1162/neco.1991.3.1.79. URL https://doi.org/10.1162/neco.1991.3.1.79.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. LiveCodeBench: Holistic and contamination free evaluation of large language models for code, 2024. URL https://arxiv.org/abs/2403.07974.

Dongfu Jiang, Xiang Ren, and Bill Yuchen Lin. LLM-Blender: Ensembling large language models with pairwise ranking and generative fusion. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 14165–14178. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.acl-long.792. URL https://aclanthology.org/2023.acl-long.792/.

Eser Kandogan, Sajjadur Rahman, Nikita Bhutani, Dan Zhang, Rafael Li Chen, Kushan Mitra, Sairam Gurajada, Pouya Pezeshkpour, Hayate Iso, Yanlin Feng, Hannah Kim, Chen Shen, Jin Wang, and Estevam Hruschka. A blueprint architecture of compound AI systems for enterprise, 2024. URL https://arxiv.org/abs/2406.00584.

Omar Khattab, Arnav Singhvi, Paridhi Maheshwari, Zhiyuan Zhang, Keshav Santhanam, Sri Vardhamanan, Saiful Haq, Ashutosh Sharma, Thomas T. Joshi, Hanna Moazam, Heather Miller, Matei Zaharia, and Christopher Potts. DSPy: Compiling declarative language model calls into self-improving pipelines, 2023. URL https://doi.org/10.48550/arXiv.2310.03714.

KServe Contributors. InferenceGraph, 2026. URL https://kserve.github.io/website/docs/concepts/resources/inferencegraph. KServe documentation, version 0.18, accessed July 15, 2026.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with PagedAttention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, pp. 611–626. ACM, 2023. doi: 10 . 1145 / 3600006 . 3613165. URL https://doi.org/10.1145/3600006.3613165.

Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. GShard: Scaling giant models with conditional computation and automatic sharding, 2020. URL https://arxiv.org/abs/2006.16668.

Wenzhe Li, Yong Lin, Mengzhou Xia, and Chi Jin. Rethinking mixture-of-agents: Is mixing different large language models beneficial?, 2025. URL https://arxiv.org/abs/2502.00674.

Xunzhuo Liu, Huamin Chen, Samzong Lu, Yossi Ovadia, Guohong Wen, Hao Wu, Zhengda Tan, Jintao Zhang, Senan Zedan, Yehudit Kerido, Liav Weiss, Haichen Zhang, Bishen Yu, Asaad Balum, Noa Limoy, Abdallah Samara, Baofa Fan, Brent Salisbury, Ryan Cook, Zhijie Wang, Qiping Pan, Rehan Khan, Avishek Goswami, Houston H. Zhang, Shuyi Wang, Ziang Tang, Fang Han, Zohaib Hassan, Jianqiao Zheng, Avinash Changrani, Xue Liu, and Bowei He. vLLM Semantic Router: Signal driven decision routing for mixture-of-modality models, 2026a. URL https://arxiv.org/abs/2603.04444.

Xunzhuo Liu, Hao Wu, Huamin Chen, Bowei He, and Xue Liu. Conflict-free policy languages for probabilistic ML predicates: A framework and case study with the semantic router DSL, 2026b. URL https://arxiv.org/abs/2603.18174.

Francesco Massa and Marco Cristofanilli. Brick: Spatial capability routing for the mixture-of-models (MoM) paradigm, 2026. URL https://arxiv.org/abs/2606.13241.

Microsoft. ONNX Runtime Execution Providers, 2026. URL https://onnxruntime.ai/docs/execution-providers/. Official documentation, accessed July 16, 2026.

MLflow Project. MLflow Model Format, 2026. URL https://mlflow.org/docs/latest/ml/model/index.html. Official documentation, accessed July 16, 2026.

Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E. Gonzalez, M. Waleed Kadous, and Ion Stoica. RouteLLM: Learning to route LLMs with preference data, 2024. URL https://arxiv.org/abs/2406.18665.

ONNX Project. ONNX Intermediate Representation Specification, 2026. URL https://onnx.ai/onnx/repo-docs/IR.html. Official specification, accessed July 16, 2026.

Open Container Initiative. OCI Image Format Specification, November 2025. URL https://specs.opencontainers.org/image-spec/. Standards Track.

Tims Pecerskis and Aivars Smirnovs. Mixture-of-models: Unifying heterogeneous agents via N-way self-evaluating deliberation, 2026. URL https://doi.org/10.48550/arXiv.2601.16863.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. GPQA: A graduate-level google-proof Q&A benchmark, 2023. URL https://arxiv.org/abs/2311.12022.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc V. Le, Geoffrey E. Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer, 2017. URL https://arxiv.org/abs/1701.06538.

Toby Simonds, Kemal Kurniawan, and Jey Han Lau. MoDEM: Mixture of domain expert models. In *Proceedings of the 22nd Annual Workshop of the Australasian Language Technology Association*, pp. 75–88. Association for Computational Linguistics, December 2024. URL https://aclanthology.org/2024.alta-1.6/.

Biao Sun, Ziming Huang, Hanyu Zhao, Wencong Xiao, Xinyi Zhang, Yong Li, and Wei Lin. Llumnix: Dynamic scheduling for large language model serving, 2024. URL https://arxiv.org/abs/2406.03243.

Yujin Tang, Edoardo Cetin, Jinglue Xu, Qi Sun, Stefan Nielsen, Vincent Richard, Haruto Goda, Iaroslav Tymchenko, Nhan Nguyen, Hyunin Lee, Mari Ashiga, Shashank Kotyan, So Kuroki, and Tarin Clanuwat. Sakana Fugu technical report, 2026a. URL https://arxiv.org/abs/2606.21228.

Yujin Tang, Edoardo Cetin, Jinglue Xu, Qi Sun, Stefan Nielsen, Vincent Richard, Haruto Goda, Iaroslav Tymchenko, Nhan Nguyen, Hyunin Lee, Mari Ashiga, Shashank Kotyan, So Kuroki, and Tarin Clanuwat. Sakana Fugu technical report, 2026b. URL https://arxiv.org/abs/2606.21228v1. Version 1; benchmark protocols used by the released scorecards.

vLLM Semantic Router Team. Micro-agent: Beat frontier models with collaboration inside model API. vLLM Blog, 2026. URL https://vllm-project.github.io/2026/06/29/micro-agent-frontier-models.html.

Junlin Wang, Jue Wang, Ben Athiwaratkun, Ce Zhang, and James Zou. Mixture-of-agents enhances large language model capabilities, 2024. URL https://arxiv.org/abs/2406.04692.

Xinquan Wang, Fenghao Zhu, Chongwen Huang, Zhaohui Yang, Zhaoyang Zhang, Sami Muhaidat, Chau Yuen, and Mérouane Debbah. TeleMoM: Consensus-driven telecom intelligence via mixture of models, 2025. URL https://doi.org/10.48550/arXiv.2504.02712.

Xinyu Zhao, Guoheng Sun, Ruisi Cai, Yukun Zhou, Pingzhi Li, Peihao Wang, Bowen Tan, Yexiao He, Li Chen, Yi Liang, Beidi Chen, Binhang Yuan, Hongyi Wang, Ang Li, Zhangyang Wang, and Tianlong Chen. Model-GLUE: Democratized LLM scaling for a large model zoo in the wild, 2024. URL https://arxiv.org/abs/2410.05357.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Tianle Li, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zhuohan Li, Zi Lin, Eric P. Xing, Joseph E. Gonzalez, Ion Stoica, and Hao Zhang. LMSYS-Chat-1M: A large-scale real-world LLM conversation dataset, 2023a. URL https://arxiv.org/abs/2309.11998.

Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Sun, Jeff Huang, Cody Hao Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E. Gonzalez, Clark Barrett, and Ying Sheng. SGLang: Efficient execution of structured language model programs, 2023b. URL https://doi.org/10.48550/arXiv.2312.07104.

A Experimental Protocol

This appendix records the protocol behind Section 6 and the evidence needed to audit an end-to-end system result. The current versioned evaluation manifest covers GPQA-Diamond and LiveCodeBench; the HLE description corresponds to the prior released scorecard retained in Figure 7. The open reference implementation is available at https://github.com/vllm-project/semantic-router. The evaluated configurations were served through an OpenAI-compatible endpoint under the shared alias `vllm-sr/auto`; each benchmark configuration nevertheless has a distinct pool and topology and should be released under its own immutable coordinate and digests.

A.1 Constituent pools and trace budgets

GPQA-Diamond. The benchmark is GPQA-Diamond (Rein et al., 2023). The evaluated MoM uses Gemini 3.1 Pro Preview, GPT-5.5, and Claude Opus 4.8. Each request launches one reasoning call to each constituent and then invokes Gemini for synthesis, for four nominal model calls. The output contract admits exactly one answer in {A,B,C,D}. The reported run contains 50 questions and is labeled *formal-50* to distinguish it from the complete 198-question benchmark.

LiveCodeBench. The evaluated window uses LiveCodeBench (Jain et al., 2024) v6, January–April 2025, with 175 problems. The pool again contains Gemini 3.1 Pro Preview, GPT-5.5, and Claude Opus 4.8. Request signals select one of two principal traces. The first uses three analysis calls followed by GPT-5.5 synthesis, for four calls in total; the second uses five breadth calls followed by GPT-5.5 synthesis, for six calls in total. The contract requires one complete Python solution with the problem's function or standard-input interface preserved. Sandboxed execution provides the task score. Five missing generations are assigned zero credit, so the denominator remains 175.

Humanity's Last Exam. The released scorecard reports a text-only filter over the 2,500-item release of Humanity's Last Exam (HLE) (Center for AI Safety et al., 2026), with 2,158 retained items. The closed realization contains Claude Opus 4.8 and Gemini 3.1 Pro Preview. Depending on request risk, it obtains two, three, or four candidate responses and then invokes Gemini for synthesis, for three to five nominal calls; its scoring judge is also Gemini 3.1 Pro Preview. The hybrid realization contains two independent GLM-5.2 sampling roles, GPT-5.5, Claude Opus 4.8, and Gemini 3.1 Pro Preview. It obtains three or four candidate responses and then invokes Gemini for synthesis, for four or five nominal calls. The contract requires an explanation, exact answer, and calibrated confidence field. The hybrid scoring path uses GPT-5.5. In both realizations, the judge identity may also participate in the execution graph, so the reported numbers are end-to-end system scores; comparisons intended to isolate constituent quality should instead use a disjoint judge pool and report multi-judge agreement.

For archival reproduction, each dataset must be pinned by repository revision, configuration, filter predicate, ordered item identifiers, and an identifier digest. In particular, *LiveCodeBench v6* here denotes the 175-item January–April 2025 configuration rather than the cumulative `release_v6`; the HLE count is produced by the text-only filter rather than a benchmark-defined split; and the GPQA formal-50 item list requires an explicit selection record. These public benchmarks predate the evaluated 2026 constituents, so no result is interpreted as contamination-free generalization.

All constituent calls in these three experiments use the same managed model gateway. Thus the experiments test heterogeneous model identities and trace topologies, while cross-provider and cross-hardware portability remain separate evaluation dimensions.

A.2 Request, trace, and scoring records

A reproducible and independently auditable MoM evaluation requires more than final predictions. Each request record should contain:

- dataset version, split, immutable example identifier, and input digest;
- logical MoM identity, behavior variant, request preference, bundle digest, and deployment-lock digest;
- signals, projection values, matched decision, selected topology, and hard-constraint outcomes;
- every constituent model and service revision, prompt digest, sampling parameters, retry index, stop reason, and terminal status;
- raw response or policy-compliant encrypted/redacted equivalent, parsed answer, contract-validation result, repair, fallback, and abstention events;
- input, cached-input, reasoning, and output token counts when available;
- per-call and end-to-end latency, timeouts, rate limits, and failures;
- realized monetary and, where estimable, energy cost under the declared accounting model; and
- scorer and judge versions, raw score, aggregation rule, and uncertainty.

Run-level metadata should additionally record engine and dependency versions, model inventory, start and end times, concurrency, seeds, counts of attempted, completed, and missing items, and checksums for every output shard. A run is complete only when the expected item identifiers and denominator match; the presence of an aggregate score file alone is insufficient.

A.3 Routing-aware fleet-planning protocol

The capacity example uses the empirical request-length distribution from LMSYS-Chat-1M, collected partly through Chatbot Arena (Zheng et al., 2023a; Chiang et al., 2024), at an arrival rate of 100 requests per second and a 500 ms time-to-first-token target. The modeled worker serves a 70B model with eight-way tensor parallelism on A100 GPUs. The homogeneous baseline assigns every request to a 65K-context pool. The routed plan assigns short requests to a 4K-context pool and long requests to the 65K pool, then sweeps the threshold subject to the same latency target. GPU cost is fixed at $2.21 per hour. Both reported plans are continuous analytical capacity estimates before rounding to whole eight-GPU workers; they are not deployable cluster counts or measurements from a production cluster.

For deployment use, the planner should first be calibrated against observed prefill/decode throughput and tail latency, then validated on held-out traces. A complete study should repeat the sweep over arrival processes, context distributions, SLOs, accelerator types, model profiles, failures, and price assumptions, and should compare analytical predictions with discrete-event and live-cluster measurements.

A.4 Release contract

An independently checkable release should publish the canonical MoM bundle, deployment binding and lock, constituent inventory, prompts, contracts, benchmark manifests, per-item traces, and deterministic report generator. Credentials, authorization headers, and private request content must never be embedded; secret references and policy-preserving redaction retain the logical identity needed for verification. All tables and figures should be regenerated from immutable request and run records, and their reported digests should match the released artifacts.

B Proposed MoM Bundle Format

This appendix instantiates the artifact contract in Section 4. The format is a versioned proposal; a machine-readable companion supplies schemas for the manifest, typed IR, realization profile, binding, lock, evaluation protocol, evaluation attestation, and run record, together with one verified example.

B.1 Bundle contents and evidence boundary

The manifest is an immutable index over the human model coordinate, behavior variant, semantic digest, typed entrypoints, canonical specification, semantic assets, contracts, evaluation protocols, and optional realization templates. Canonical IR remains the sole semantic authority. Authoring source and model cards are optional distribution objects rather than executable authorities. Evaluation results, signatures, SBOMs, and build provenance are separate digest-bound attestations: publishing new evidence does not change the model being evaluated.

Every object descriptor contains a normalized relative path, media type, SHA-256 digest, and byte size. Absolute, backslash, empty, dot, repeated, traversal, and symlink paths are rejected before parsing. A production archive importer must additionally bound expanded size, file count, nesting, and compression ratio and reject hard links, devices, and other non-regular files.

The example exposes one accuracy-first model and two realization profiles: ROCm plus cloud and CUDA plus cloud. Behavior variants belong to logical identity; realization profiles contain endpoint-free compatibility requirements and do not. A binding chooses one realization profile and supplies concrete candidates.

B.2 Four identities without circular hashes

Let C be canonical JSON and H be SHA-256. Let I be the fully defaulted and typed IR, whose semantic assets are themselves content-addressed. Let M be the manifest, whose object descriptors commit to the distributed byte closure. Let $N(B_s)$ be the normalized typed representation obtained from binding source B_s, and let L be the completed normalized lock. The four identities are

$$\begin{aligned} d_{\text{semantic}} &= H(C(I)), & d_{\text{bundle}} &= H(C(M)), \\ d_{\text{binding}} &= H(C(N(B_s))), & d_{\text{lock}} &= H(C(L)). \end{aligned} \tag{18}$$

No object stores its own digest. The manifest stores d_{semantic}; the binding stores semantic and bundle subjects; the lock stores that subject and d_{binding}; traces and attestations then store all four. Formatting a YAML binding therefore cannot create a new environment identity.

B.3 Target compatibility and locking

A logical constituent declares an identity and required or preferred capabilities, including protocol, modality, interface features, input and output limits, and data policy. A realization profile may additionally require a wire transport, API adapter, serving runtime, leaf-artifact format, accelerator API, vendor, architecture, or deployment driver. Environment policy may strengthen but never relax semantic constraints. Resolution is thus the intersection of logical admissibility, realization-profile requirements, and environment policy.

The lock records the complete candidate inventory available to an execution, not the one candidate selected by a particular request. For open artifacts it may pin a content digest or repository revision together with tokenizer, runtime image, protocol adapter, and accelerator evidence. For a closed service it marks identity as observed-opaque, records an observation time and probe digest, and makes no bitwise-reproducibility claim. Floating dependencies are excluded from a production lock.

B.4 Distribution closures and evaluation

A thin export retains immutable external references; a materialized export vendors leaf artifacts whose licenses permit redistribution; and a targeted export additionally carries runtime objects for selected realization profiles. Materialization fails for a closed provider rather than treating an API alias as an offline model. A canonical directory and deterministic archive define local exchange; OCI descriptors, manifests, and referrers provide a natural registry transport and evidence attachment.

An evaluation protocol is part of the bundle because it defines an assessment procedure. A result is an external attestation over semantic, bundle, binding, lock, protocol, dataset, and harness digests, plus sampling and concurrency settings. Serving and evaluation therefore consume the same resolved model identity, while later measurements remain independently publishable.

B.5 Conformance

The companion verifier checks all supplied schemas; stable canonical identities; traversal-safe paths, byte sizes, and digests; manifest/IR semantic closure; behavior variants and realization profiles; graph reachability, explicit contract branches, universal termination, and finite bounds; fail-closed hard constraints; constituent identity and capability satisfaction; complete lock inventory; pinned and observed-opaque evidence; secret non-inclusion; and run and evaluation attribution to all four identities.

A complete cross-engine suite must additionally test deterministic compilation, source-to-IR semantic preservation, operator capability negotiation, archive hardening, license and extension trust, trace conformance, and portability across at least two disclosed bindings while fixing semantic and bundle digests.