International Journal of Computer Information Systems and Industrial Management Applications
ISSN 2150-7988 Volume 18(18s) 2026 pp. 232-242
https://doi.org/10.70917/ijcisim-2026-4862
Article

Intelligent Inference Endpoint Scheduling for Heterogeneous Large Language Model Deployments: A Six-Dimensional Routing Framework

Rajalakshmi Srinivasaraghavan

Independent Researcher, Leander, Texas
reachsrajalakshmi@gmail.com

Abstract: Large Language Model (LLM) serving infrastructure has evolved from single-model deployments into heterogeneous fleets combining general-purpose models, domain-specialised variants, multimodal models, and quantised derivatives with widely differing context capacities. In such environments, the decision of which endpoint should serve a given request materially affects latency, throughput, monetary cost, and output quality. Despite growing industrial adoption of inference routers, the literature lacks a consolidated account of the decision dimensions such systems must reason over, or of how those dimensions interact when their objectives conflict. This paper addresses that gap. Through an analysis of inference-serving mechanisms and model-specialisation results, we identify and characterise six dimensions governing endpoint selection: load distribution, context-length requirement, input modality, task category, domain specialisation, and prefix-cache locality. For each dimension we formalise the decision criterion and describe the signals available at request-admission time, distinguishing hard feasibility constraints from soft optimisation preferences. We further show that these dimensions are not independent - notably, cache-locality routing and load-balancing objectives are structurally opposed, since the former concentrates traffic while the latter disperses it - and we propose a bounded-affinity policy and a priority-ordered evaluation sequence to resolve such conflicts. Finally, we specify an evaluation protocol comprising workload definitions, baselines, and metrics, by which implementations of the framework may be empirically validated and compared. The framework is intended as a design reference for practitioners building inference gateways and as a structuring basis for future empirical work on multi-objective LLM request scheduling.
Keywords: large language model serving; inference optimisation; request routing; prefix caching; context-aware scheduling; multi-objective load balancing; model selection

1. INTRODUCTION

A. Background and Motivation

The operational cost of serving large language models is dominated by accelerator time, and accelerator time is consumed unevenly. A request comprising a ten-token prompt and a short completion may be served in well under a second, whereas a request carrying several thousand tokens of context and generating a comparably long output may occupy the same hardware for tens of seconds. This variance of two to three orders of magnitude within a single request stream distinguishes LLM serving from conventional stateless web workloads, where request cost is approximately uniform and round-robin distribution is near-optimal.

The heterogeneity of the serving fleet compounds this variance. A contemporary production deployment may concurrently host general-purpose models, domain-adapted models for code [5], biomedical [6], or financial [7] text, multimodal models accepting image input [3], [4], and quantised variants offering reduced precision at lower cost. Context capacities across such a fleet may span more than an order of magnitude [4]. Each endpoint therefore differs

not only in throughput characteristics but in capability: certain requests cannot be served by certain endpoints at all, and many requests that can be served by a highly capable endpoint would be served adequately, and more cheaply, by a smaller one.Directing a request to a suboptimal endpoint under these conditions produces one of three failure modes. Capability mismatch occurs where the selected endpoint cannot satisfy the request, whether for want of context capacity or of the required input modality. Over-provisioning occurs where an unnecessarily capable endpoint is selected and cost is expended without corresponding benefit. Queue imbalance occurs where load is distributed without regard to per-request cost, so that endpoints receiving disproportionately expensive requests develop queues while others remain underutilised, degrading tail latency. Intelligent inference routing is the middleware layer that seeks to avoid all three.

B. Research Gap

Individual routing-relevant mechanisms have received substantial attention. Memory-efficient batching and paged attention have been studied in the context of single-endpoint serving [1]. Reuse of computed key-value state across requests sharing a common prefix has been documented as a latency and cost optimisation [1], [2]. Reward-guided selection among an ensemble of models has been proposed as a means of directing queries to the most suitable model [8]. Evidence that domain-adapted models outperform general models within their domain is established across several fields [5], [6], [7], and multimodal architectures extending language models to image and video input are similarly documented [3], [4].

What this body of work does not provide is a consolidated account of these mechanisms as facets of a single admission-time decision. In practice a production router must evaluate them simultaneously, and their objectives are not mutually consistent: routing for cache locality concentrates traffic on endpoints already holding relevant state, whereas routing for load balance disperses it. Existing treatments consider each mechanism in isolation, and consequently neither enumerate the full set of decision dimensions nor address the resolution of conflicts among them.

C. Research Questions

This work is guided by three questions:

• RQ1. What decision dimensions must an inference router evaluate at request-admission time in a heterogeneous LLM deployment?

• RQ2. What signals are available at admission time to evaluate each dimension, and at what computational cost?

• RQ3. Where do these dimensions conflict, and how should conflicts be resolved?

D. Contributions

The contributions of this paper are as follows:

1. A six-dimensional characterisation of LLM inference routing decisions, distinguishing hard feasibility constraints from soft optimisation preferences.

2. A formalisation of the admission-time decision criterion for each dimension, including the signals required and the cost of extracting them.

3. An analysis of inter-dimensional conflict, in particular the structural tension between cache-locality routing and load balancing, together with a bounded-affinity resolution policy and a priority-ordered evaluation sequence.

4. An evaluation protocol specifying workloads, baselines, and metrics, intended to permit comparable empirical assessment of routing frameworks.

E. Organisation

Section 2 reviews related work and positions this contribution. Section 3 describes the analytical method. Sections 4 through 9 treat each routing dimension in turn. Section 10 presents a reference architecture and the conflict-resolution strategy. Section 11 specifies the evaluation protocol. Section 12 states limitations, and Section 13 concludes.

2. RELATED WORK

A. Inference Serving Systems

Work on LLM serving infrastructure has focused predominantly on maximising utilisation of a single endpoint or homogeneous cluster. PagedAttention introduced non-contiguous key-value cache memory management, allowing memory to be allocated in fixed-size blocks rather than contiguous per-sequence reservations, and thereby raising achievable batch sizes under memory pressure [1]. Such systems optimise scheduling within an endpoint and generally assume that request-to-endpoint assignment has already occurred. The present work addresses the complementary problem of that assignment.

B. Model Selection and Ensemble Routing

A second line of work treats model choice itself as the object of optimisation. Reward-guided ensemble routing directs a query to the model within an ensemble expected to produce the highest-reward response, treating selection as a learned prediction problem [8]. Such approaches address the quality-cost trade-off directly and provide a candidate mechanism for the task and domain dimensions described in Sections 7 and 8. They do not, however, incorporate the operational state of the serving fleet - queue depth, cache occupancy, or context capacity - and therefore address a subset of the decision treated here.

C. Caching and Prefix Reuse

Reuse of computed key-value state across requests sharing a common token prefix eliminates recomputation of the shared segment, and has been documented in both research systems [1] and commercial deployments [2]. Existing treatments largely consider caching as a single-endpoint optimisation concerned with cache admission and eviction. Its implications for routing policy, and the resulting conflict with load balancing, are analysed in Section 9.3.

D. Domain and Modality Specialisation

Evidence that domain-adapted models outperform general models of comparable scale within their target domain is established in the code [5], biomedical [6], and financial [7] settings. Multimodal architectures extending language models to image and video input are documented in [3] and [4], the latter also demonstrating substantially extended context capacity. This literature establishes that endpoint capability varies along domain and modality axes, and thereby motivates the corresponding routing dimensions, but it addresses model construction and evaluation rather than the assignment decision that capability heterogeneity necessitates.

E. Positioning

Table 1 positions this work against the categories above. Each cell records the extent to which a body of work treats the corresponding dimension as an endpoint-selection criterion, which is distinct from establishing that the underlying factor affects serving cost or output quality. The distinction matters for the domain and modality dimensions in particular: the specialisation literature establishes that capability differs across endpoints, but does not address how a request should be assigned in consequence.

Prior work category	Load	Context	Modality	Task	Domain	Cache	Unified analysis

Serving systems (§2.1)	◐	○	○	○	○	◐	○
Ensemble routing (§2.2)	○	○	○	●	◐	○	○
Caching work (§2.3)	○	○	○	○	○	●	○
Specialisation studies (§2.4)	○	○	◐	○	◐	○	○
This work	●	●	●	●	●	●	●

Table 1. Coverage of endpoint-selection dimensions in prior work. ● = treated explicitly as a routing criterion; ◐ = addressed partially, or established as a capability difference without an accompanying assignment policy; ○ = not addressed.

3. METHOD

This paper develops an analytical framework rather than reporting original measurements. The method proceeded in three stages.

Stage 1 - Mechanism identification. Mechanisms affecting request-to-endpoint assignment were identified from the serving-systems, model-selection, caching, and specialisation literature reviewed in Section 2. A mechanism was admitted where it satisfied two criteria: that it be evaluable from request metadata or observable system state at the moment of admission, and that the underlying factor be documented as affecting serving cost, feasibility, or output quality.

Stage 2 - Dimensional characterisation. Admitted mechanisms were grouped by the signal they consume, yielding the six dimensions presented in Sections 4 through 9. For each, the decision criterion was formalised, the extraction cost of the required signal was assessed, and the dimension was classified as a hard feasibility constraint or a soft optimisation preference according to whether its violation produces request failure or merely inefficiency.

Stage 3 - Interaction analysis. Each pair of dimensions was examined for objective compatibility, that is, whether optimising one moves the assignment toward or away from the optimum of the other. This analysis yields the conflict identified in Section 9.3 and the evaluation ordering proposed in Section 10.2.

Scope of claims. The framework is analytical and is not validated by measurement in this paper. Claims regarding the relative cost of mechanisms are argued from their computational structure, and claims regarding capability differences between endpoint classes are attributed to the cited sources. No performance figure is asserted on the author's own authority, and Section 11 specifies the protocol by which the framework's claims may be tested.

4. DIMENSION I: LOAD DISTRIBUTION

A. Why Conventional Load Balancing Fails

Conventional load balancers assume approximately uniform request cost, which permits round-robin or least-connections distribution to approximate optimal utilisation. Three properties of LLM inference violate this assumption.

First, cost variance: service time is a function of prompt length and generated length, and varies by two to three orders of magnitude within a typical request stream. Second, statefulness: a request occupies key-value cache memory for the full duration of its generation, so endpoint capacity is bounded by memory as well as by compute, and the binding constraint may differ between endpoints [1]. Third, deferred cost revelation: output length is unknown at

admission time and must be estimated, making exact cost-aware assignment impossible in principle rather than merely difficult in practice.

B. Token-Weighted Cost Estimation

Let a request r have input length n_"in" (r) tokens and estimated output length n^_"out" (r). The admission-time cost estimate is

$$\hat{C}(r) = \alpha \cdot n_{\text{in}}(r) + \beta \cdot \hat{n}_{\text{out}}(r)$$

where α and β are endpoint-specific coefficients reflecting the differing computational profiles of the prefill and decode phases. Prefill processes the input sequence in parallel and is compute-bound; decode generates tokens sequentially, each step requiring a full pass over model weights, and is bound by memory bandwidth. Consequently the per-token cost of decode exceeds that of prefill on typical accelerator hardware, so that $\beta > \alpha$. Both coefficients are properties of a specific model and hardware configuration and must be calibrated by regression of observed service times against the corresponding token counts.

The estimator n^_"out" may be derived from an explicit output-limit parameter where the client supplies one, from a historical mean conditioned on task category, or from a learned predictor. The accuracy of this estimate bounds the achievable quality of cost-aware assignment and is a principal source of error in deployment.

C. Batch-Aware Admission

Continuous batching permits arriving requests to join an in-flight batch rather than waiting for the current batch to complete [1]. This creates a deliberate admission-delay decision: briefly holding a request so that it may join a forming batch can raise aggregate throughput at the cost of added latency for that request. The optimal hold window depends on arrival rate and batch composition, and represents a tunable throughput-latency trade-off rather than a fixed value.

D. Tiered Assignment

Where the fleet is organised into capability tiers, the router may direct each request to the lowest tier expected to satisfy it, reserving high-capability capacity for requests requiring it. Tier assignment depends on the task and domain dimensions of Sections 7 and 8, and on a quality threshold that the operator must set explicitly, since that threshold encodes an organisational tolerance for quality degradation in exchange for cost reduction and cannot be derived from system state.

5. DIMENSION II: CONTEXT-LENGTH REQUIREMENT

A. The Cost Structure of Context

Context capacity varies substantially across a heterogeneous fleet, with extended-context models supporting sequence lengths more than an order of magnitude beyond conventional deployments [4]. The cost of context is not linear in its length. Self-attention exhibits quadratic time complexity in sequence length during prefill, while key-value cache memory grows linearly per concurrent sequence, bounding the number of long-context requests an endpoint may serve simultaneously [1].

This yields a consequence relevant to routing but not to single-endpoint scheduling: serving a short request on a long-context endpoint consumes memory that could otherwise support several concurrent short requests, degrading aggregate fleet throughput even though the request itself completes without incident. Context-based assignment is therefore an optimisation over the fleet, not merely a feasibility check on the individual request.

B. Context Estimation

Total context requirement is estimated at admission as

n_"total" (r)=n_"sys" +n_"user" +n_"hist" +n^_"out"

comprising system prompt, user input, retained conversation history, and estimated output. Where retrieval augmentation is used, retrieved passage length must be included and is frequently the dominant term. All components except n^_"out" are known exactly at admission, so estimation error in this dimension derives entirely from the output-length estimator of Section 4.2.

C. Threshold-Based Assignment

Requests are assigned to the smallest-capacity endpoint class satisfying n_"total" (r)≤γ·"capacity" (e), where γ<1 is a safety margin absorbing estimation error. Representative thresholds are given in Table 2. These are configuration parameters reflecting a particular fleet composition rather than universal constants, and must be re-derived for each deployment from the capacities actually present.

Estimated context	Target endpoint class
Below small-context capacity	Any available; prefer smallest capacity
Within medium-context capacity	Medium-context class
Within large-context capacity	Large-context class
Exceeding large-context capacity	Extended-context class only

Table 2. Structure of context-based assignment thresholds.

The consequences of estimation error are asymmetric. Under-estimation of n_"total" causes truncation or outright failure once generation exceeds the endpoint's capacity, whereas over-estimation causes only unnecessary cost. This asymmetry generally justifies a conservative choice of γ.

6. DIMENSION III: INPUT MODALITY

A. Modality as a Feasibility Constraint

Modality differs from the preceding dimensions in being a hard constraint rather than an optimisation: a text-only endpoint cannot serve a request containing an image, irrespective of its load or cost advantage. Modality is therefore evaluated first in the decision sequence of Section 10.2, as a feasibility filter that reduces the candidate endpoint set before any optimisation criterion is applied.

Detection is inexpensive, requiring only inspection of the request payload for non-text content blocks, and adds negligible latency to admission. This combination of hard constraint and cheap evaluation is what justifies its position at the head of the sequence.

B. Token Accounting for Visual Input

Multimodal architectures convert visual input into token-equivalent representations before processing, with the resulting count scaling with image resolution and the tiling strategy employed [3], [4]. A high-resolution image may therefore contribute a token count comparable to a substantial body of text, with corresponding effect both on cost and on the context estimate of Section 5.

Two pre-routing transformations follow. Resolution reduction to the minimum sufficient for the task lowers the token contribution directly, and format normalisation ensures compatibility across endpoints that accept differing encodings. Both are performed by the router before assignment, and both reduce the context requirement subsequently passed to the assignment of Section 5.3.

C. Avoiding Unnecessary Multimodal Assignment

Because multimodal endpoints are generally more expensive per request than text-only endpoints of comparable scale, two checks are worthwhile before committing to multimodal assignment. The first determines whether visual input is material to the request or merely incidental, as where an image is attached but not referenced by the accompanying instruction. The second determines whether the visual task is constrained enough - classification or optical character recognition, rather than open-ended visual reasoning - to be served by a smaller vision-capable endpoint. Both checks reduce the candidate set to the cheapest endpoint class that remains feasible.

7. DIMENSION IV: TASK CATEGORY

A. Task-Dependent Model Suitability

Task categories differ in their sensitivity to model capability. Extractive tasks with well-defined output schemas, such as entity extraction or classification, are frequently served adequately by smaller fine-tuned models. Generative tasks requiring extended coherent reasoning, such as complex code synthesis or multi-step analysis, degrade more sharply as model capability is reduced. Categories of practical relevance in production deployments include summarisation, entity extraction, code generation, creative generation, and question answering.

The routing implication is that the marginal quality gain obtained from a higher-capability endpoint is itself a function of task category. Tier assignment as described in Section 4.4 should therefore be conditioned on task category rather than applied uniformly across the request stream.

B. Classification Mechanisms

Three mechanisms are available, with differing cost and reliability profiles.

Explicit declaration, in which the client supplies a task hint as a request parameter, is exact and imposes no computational cost, but requires client cooperation and cannot be relied upon in open deployments where clients are untrusted or unmodified.

Lexical heuristics, matching keywords and patterns against the prompt, are computationally negligible but brittle under paraphrase, and are prone to misclassifying prompts that contain task-adjacent vocabulary incidentally.

Lightweight learned classification, applying a compact encoder to a prefix of the prompt, offers greater robustness at the cost of an inference pass on every admitted request. The governing design constraint is that this cost is incurred unconditionally, whereas the routing error it prevents occurs only on some fraction of requests; the classifier must therefore be cheap relative to the expected cost of the misroutes it avoids, which in practice restricts the choice to models far smaller than any serving endpoint.

C. Handling Misclassification

Because no classifier is exact, routing policy must specify behaviour under low classification confidence. The conventional resolution is to fall back to a general-purpose endpoint, accepting higher cost in exchange for bounded quality risk. This is the same confidence-thresholded structure set out for the domain dimension in Section 8.2, and in an implementation the two are naturally handled by a common mechanism.

8. DIMENSION V: DOMAIN SPECIALISATION

A. Evidence for Domain Adaptation

Domain-adapted models have been shown to outperform general models within their target domain in the code [5], biomedical [6], and financial [7] settings. Domains of practical relevance to production deployments include mathematics, programming, medicine, law, and finance. Where specialised endpoints are present in the fleet, directing

domain-matched requests to them offers quality improvement, cost reduction, or both, since the specialised model is frequently smaller than the general model it displaces.

B. Confidence-Thresholded Assignment

Let $c_d(r) \in [0,1]$ denote classifier confidence that request r falls within domain d. Assignment follows a three-band policy governed by thresholds $\tau_{\text{"low"}}$ and $\tau_{\text{"high"}}$:

$c_d(r) > \tau_{\text{"high"}}$: route to the specialised endpoint for domain d;

$\tau_{\text{"low"}} \le c_d(r) \le \tau_{\text{"high"}}$: route to a high-capability general endpoint;

$c_d(r) < \tau_{\text{"low"}}$: route to a general endpoint by default policy.

Both thresholds are deployment-specific and should be set by weighing the quality cost of misrouting against the cost saving obtained by correct specialised assignment. The two error directions are not symmetric: directing a general request to a narrowly specialised endpoint typically degrades output quality more severely than directing a specialised request to a capable general endpoint, since the latter merely forgoes an improvement while the former may fall outside the specialised model's competence altogether. This asymmetry argues for setting $\tau_{\text{"high"}}$ conservatively high.

9. DIMENSION VI: PREFIX-CACHE LOCALITY

A. Mechanism

Key-value state computed during prefill may be retained in accelerator memory and reused by subsequent requests sharing a common token prefix, eliminating recomputation of the shared segment [1], [2]. Because prefill is compute-bound and quadratic in sequence length, the saving is greatest precisely where the cost is greatest, namely for long shared prefixes.

The property relevant to routing is that this state is endpoint-local. A cache entry residing on one endpoint confers no benefit on a request assigned to another, which makes cache occupancy a property of the assignment decision rather than solely of the serving endpoint's internal scheduling.

B. Exploitable Patterns

Cache reuse is available wherever request structure is repetitive. Fixed system prompts are shared across all requests issued by a given application and yield reuse proportional to their length. Templated requests share a structural prefix with a variable suffix. Multi-turn conversations share the accumulated dialogue history with all preceding turns, so that reuse grows as the conversation lengthens. Document-grounded question answering shares the document across all queries issued against it, and typically offers the greatest absolute saving owing to prefix length. Few-shot prompts share the exemplar block preceding the query.

In every case reuse is available only after the prefix has been computed once and while the corresponding entry survives eviction, so achievable reuse is bounded by both cache capacity and the temporal distribution of requests sharing the prefix.

C. The Cache-Load Conflict

Cache-aware routing directs requests sharing a prefix to the endpoint already holding the corresponding state. This is affinity routing, and its objective is directly opposed to that of Section 4: load balancing seeks to disperse requests across endpoints, while cache locality seeks to concentrate them.

The conflict is structural rather than incidental. A cache-optimal policy would direct all traffic sharing a common system prompt to a single endpoint, saturating it while others idle. A load-optimal policy would distribute

that same traffic evenly, reducing every endpoint's hit rate toward the level achievable with no affinity at all. Neither extreme is desirable, and the optimum lies at an interior point determined by the workload's prefix-sharing rate and the fleet's spare capacity.

This paper proposes bounded affinity as the resolution. Cache affinity is honoured only while the queue depth of the preferred endpoint remains below a threshold θ; above that threshold the request overflows to the next-best feasible endpoint and accepts a cache miss. The threshold encodes the operator's exchange rate between latency saved by a cache hit and latency added by queueing, and is therefore a property of the deployment rather than of the framework. Determination of θ under realistic workloads is an open question and a natural subject for the empirical work outlined in Section 11.

10. ARCHITECTURE

A. Components

A router implementing the six dimensions comprises six components.

The request analyser parses the incoming payload and extracts routing features: modality, token counts, task and domain classification, and the prefix hash used for cache lookup. The endpoint registry maintains per-endpoint metadata covering modality support, context capacity, domain specialisation, capability tier, cost coefficients, and health status. The decision engine applies the dimensional criteria in the order established below and emits an assignment. The cache index maps prefix hashes to the endpoints holding the corresponding key-value state, and tracks eviction so that stale affinity is not honoured. The load monitor maintains per-endpoint queue depth, in-flight token count, and memory occupancy. Telemetry records assignment decisions and realised outcomes for subsequent policy calibration.

B. Decision Ordering and Conflict Resolution

The dimensions are not commutative, and the framework specifies the following evaluation order:

Modality (§6) - hard feasibility constraint; eliminates endpoints that cannot accept the input.

Context capacity (§5) - hard feasibility constraint; eliminates endpoints of insufficient capacity.

Domain and task (§7, §8) - quality constraints; restrict the candidate set to endpoints meeting the quality threshold for the request class.

Cache locality (§9) - soft preference over the surviving candidates.

Load (§4) - soft preference and final tiebreaker; additionally overrides the cache preference where queue depth exceeds θ.

Hard constraints precede soft preferences because their violation produces request failure rather than inefficiency, and because they are the cheapest to evaluate, so that applying them first minimises the candidate set over which more expensive criteria must be computed. Quality constraints precede optimisation preferences because they bound the acceptable outcome rather than merely ranking outcomes within it. Cache preference precedes load preference in expression but yields to it under saturation, as established in Section 9.3.

This ordering is derived from the constraint analysis of Sections 4 through 9. Its optimality is not established empirically in this paper, and it constitutes a specific hypothesis to be tested under the protocol of Section 11.

C. Observability Requirements

Policy calibration requires that realised outcomes be recorded against the decisions that produced them. The minimum instrumentation comprises the assignment decision together with the dimension that determined it; the realised output length alongside the estimate n^_"out" , permitting calibration of the estimator; the cache hit or miss

outcome; end-to-end and queueing latency, separated so that admission delay may be distinguished from service time; and, where obtainable, a quality signal such as task success or user acceptance.

Without the second and third of these, neither the cost model of Section 4.2 nor the threshold of Section 9.3 can be tuned, and the framework degenerates to a set of fixed heuristics.

11. EVALUATION PROTOCOL

The framework presented above is analytical and is not empirically validated in this paper. This section specifies a protocol by which it may be evaluated, in order that results obtained from independent implementations be mutually comparable.

Workloads. Evaluation should span at least three request mixes: a homogeneous short-prompt mix representing interactive conversational traffic; a mixed-length mix combining short conversational requests with long document-grounded requests; and a multimodal mix incorporating image input. Each mix should be characterised by its input-length distribution, output-length distribution, prefix-sharing rate, and arrival process, since results are not interpretable without them.

Baselines. Proposed policies should be compared against round-robin assignment, least-connections assignment, and single-dimension policies - cache-affinity-only and load-only - so that the marginal contribution of combining dimensions may be distinguished from the contribution of any dimension alone.

Metrics. Latency at the 50th, 95th, and 99th percentiles; sustained throughput measured in both requests and tokens per second; cost per thousand requests under an explicitly stated price model; cache hit rate; and routing accuracy, defined as the proportion of requests assigned to the cheapest endpoint meeting the applicable quality threshold, evaluated against a labelled sample.

Ablation. Each dimension should be disabled in turn to quantify its marginal contribution, and the decision ordering proposed in Section 10.2 should be tested against alternative orderings, since that ordering is a hypothesis of this work rather than an established result.

Environment reporting. Results should state accelerator type and count, serving framework and version, the composition of the model fleet, and whether measurements derive from simulation or live serving. Simulation is acceptable for the load and cache dimensions provided the service-time model is calibrated against measurement and the calibration procedure is reported.

12. LIMITATIONS

Four limitations bound the claims of this paper.

First, the framework is not empirically validated. The dimensional characterisation is derived from analysis of documented mechanisms, and the decision ordering of Section 10.2 is argued from the character of the constraints rather than demonstrated by measurement. Section 11 exists precisely because this validation remains outstanding.

Second, the analysis rests on a small body of primary sources. Broader synthesis, particularly of production deployment experience, would strengthen the dimensional characterisation and might reveal dimensions not identified here.

Third, the six dimensions are treated as approximately separable for the purpose of establishing a decision ordering. Section 9.3 demonstrates that at least one pair interacts materially, and further interactions may exist that the present analysis has not identified. The proposed ordering should therefore be regarded as a starting point for empirical refinement rather than a settled result.

Fourth, the scope is confined to admission-time routing across a static endpoint fleet. Dynamic capacity adjustment, cross-region placement, speculative decoding, disaggregated prefill-decode serving, and heterogeneous accelerator scheduling all interact with the routing decision and lie outside the present treatment.

13. CONCLUSION AND FUTURE WORK

This paper has presented a six-dimensional framework for inference endpoint selection in heterogeneous LLM deployments, addressing the three questions posed in Section 1.3.

In answer to RQ1, six dimensions were identified: load distribution, context requirement, modality, task category, domain specialisation, and prefix-cache locality. In answer to RQ2, the admission-time signals for each were characterised, distinguishing hard feasibility constraints, which are cheap to evaluate from request metadata alone, from quality constraints, which require classification and therefore impose an unconditional latency cost on every admitted request. In answer to RQ3, the dimensions were shown not to be independent: cache-locality routing and load balancing are structurally opposed, and a bounded-affinity policy governed by a queue-depth threshold was proposed to resolve the conflict, together with a priority ordering placing feasibility constraints before quality constraints and quality constraints before optimisation preferences.

The principal contribution is the consolidation of mechanisms previously studied in isolation into a single admission-time decision, together with an explicit ordering of that decision and an evaluation protocol permitting comparable measurement across implementations.

Three directions follow. The most immediate is empirical validation of the proposed ordering against the protocol of Section 11, and in particular the determination of the affinity threshold θ under realistic prefix-sharing rates, since that threshold governs the one interaction the framework identifies as structural. The second is the replacement of the hand-set thresholds appearing throughout the framework with policies learned from the telemetry described in Section 10.3, framing endpoint selection as a contextual bandit problem over the constraint-filtered candidate set. The third is extension to heterogeneous accelerator fleets and to disaggregated architectures in which prefill and decode are served by distinct hardware, both of which alter the cost model of Section 4.2 and therefore the entire routing calculus that depends upon it.

REFERENCES

1. W. Kwon, Z. Li, S. Zhuang, Y. Sheng, L. Zheng, C. H. Yu, J. E. Gonzalez, H. Zhang, and I. Stoica, “Efficient memory management for large language model serving with PagedAttention,” in Proc. 29th ACM Symp. Operating Systems Principles (SOSP), Koblenz, Germany, 2023, pp. 611–626.
2. Anthropic, “Prompt caching with Claude,” Technical Documentation, 2024.
3. OpenAI, “GPT-4 technical report,” arXiv preprint arXiv:2303.08774, 2023.
4. Google DeepMind, “Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context,” Technical Report, 2024.
5. Meta AI, “Code Llama: Open foundation models for code,” arXiv preprint arXiv:2308.12950, 2023.
6. K. Singhal, S. Azizi, T. Tu, S. S. Mahdavi, J. Wei, H. W. Chung, N. Scales, A. Tanwani, H. Cole-Lewis, S. Pfohl, P. Payne, M. Seneviratne, P. Gamble, C. Kelly, N. Schärli, A. Chowdhery, P. Mansfield, B. Aguera y Arcas, D. Webster, G. S. Corrado, Y. Matias, K. Chou, J. Gottweis, N. Tomasev, Y. Liu, A. Rajkomar, J. Barral, C. Semturs, A. Karthikesalingam, and V. Natarajan, “Large language models encode clinical knowledge,” Nature, vol. 620, no. 7972, pp. 172–180, 2023.
7. S. Wu, O. Irsoy, S. Lu, V. Dabravolski, M. Dredze, S. Gehrmann, P. Kambadur, D. Rosenberg, and G. Mann, “BloombergGPT: A large language model for finance,” arXiv preprint arXiv:2303.17564, 2023.
8. L. Yu, et al., “Routing to the expert: Efficient reward-guided ensemble of large language models,” arXiv preprint arXiv:2311.08692, 2023.