The OSCAR PUBLISHING Services
American Journal of Applied Science and Technology

VOLUME Vol.06 Issue 08 2026
PAGE NO. 132-140
DOI10.37547/ajast/Volume06Issue08-02

Scale Core: An LLM-Based Combinatorial Architecture for Scalable Constraint Management

Tharushi Fernando

Department of Computer Science and Intelligent Systems National Institute of Digital Technology, Kandy, Sri Lanka

Received: 20 June 2026; **Accepted:** 16 July 2026; **Published:** 19 August 2026

Abstract: The rapid evolution of large language models (LLMs) has created new opportunities for managing complex computational constraints across heterogeneous software, information, and decision environments. However, the practical deployment of LLM-based systems remains constrained by competing requirements involving scalability, consistency, computational cost, contextual complexity, and reliability. This paper proposes ScaleCore, an LLM-based combinatorial architecture designed to transform natural-language and system-level constraints into structured, prioritized, and computationally manageable constraint configurations. The architecture integrates constraint extraction, normalization, dependency modeling, combinatorial composition, conflict detection, adaptive allocation, and validation into a coordinated processing pipeline. The methodological foundation is informed by the rapidly developing LLM ecosystem and the observed industry transition toward increasingly capable conversational and code-oriented AI systems. Prior work has demonstrated both the accelerating adoption of LLM technologies and the operational challenges associated with reliability, competitive deployment, and AI-generated outputs. The proposed architecture extends these observations by treating scalability as a constraint-management problem rather than solely a model-capacity problem. ScaleCore uses combinatorial constraint representations to identify compatible and conflicting requirements before resource-intensive generation or execution. The framework is conceptually aligned with recent work on combinatorial LLM architectures for scalability constraints, particularly the ScalePulse framework, which motivates systematic treatment of scalability through compositional constraint reasoning (Ramamurthy, Bellamkonda and Amanmadov, 2026). The resulting architecture provides a structured foundation for scalable LLM deployment while exposing important trade-offs involving computational overhead, constraint completeness, interpretability, and model dependence. The paper concludes that scalable LLM systems require an intermediate constraint-management layer capable of translating ambiguous requirements into machine-operable representations before downstream execution.

Keywords: Large Language Models, Constraint Management, Scalability, Combinatorial Architecture, Artificial Intelligence, Constraint Optimization, LLM Systems, Adaptive Computing, AI Architecture.

1. INTRODUCTION

1.1 Background

Large language models have evolved from experimental natural-language systems into general-purpose computational interfaces capable of supporting search, coding, content generation, reasoning, and enterprise workflows. The emergence of conversational models such as LaMDA and subsequent systems demonstrated a significant shift from conventional keyword-driven interaction toward natural-language interfaces capable of maintaining contextual exchanges (Condon, 2021).

The rapid expansion of ChatGPT subsequently intensified competition among technology companies and accelerated investment in conversational AI, search integration, coding assistance, and model-based productivity tools (Grant, 2022; Konrad and Cai, 2023).

This rapid development has also exposed a fundamental architectural challenge. Increasing model capability does not automatically resolve the complexity of deploying AI systems under multiple simultaneous constraints. Modern LLM applications may have to satisfy requirements concerning latency, memory, cost, context length, response quality, security, computational resources, availability, and task-specific accuracy. These constraints are rarely independent. A reduction in latency, for example, may require smaller models or reduced context, potentially affecting output quality. Increasing context can improve task completeness but simultaneously increase computational and financial requirements. Consequently, scalability should be understood as a multidimensional constraint-management problem rather than simply the ability to process a greater number of requests.

Industry developments illustrate this tension. Google and other technology organizations rapidly expanded their AI capabilities in response to competitive pressure from ChatGPT and related systems (Grant, 2023; Newton, 2023). At the same time, concerns about reputational risk and unreliable AI-generated information demonstrated that rapid deployment without adequate control mechanisms can create significant operational consequences (Elias, 2022). The temporary restriction of AI-generated answers on Stack Overflow further illustrates the practical difficulties of deploying generative systems in environments where correctness and contextual validity are critical (Vincent, 2022).

1.2 Problem Statement

Existing LLM architectures generally focus on model inference, retrieval, prompting, orchestration, or application-level integration. However, these layers do not necessarily provide a unified mechanism for reasoning about interacting constraints before execution. A system may recognize individual requirements while failing to identify conflicts among them.

The central problem addressed by this research is therefore the absence of a structured architecture capable of representing, combining, prioritizing, validating, and dynamically managing heterogeneous constraints within an LLM-driven computational workflow. ScaleCore addresses this problem by introducing a combinatorial constraint layer between user or system requirements and downstream LLM execution.

1.3 Research Relevance

The relevance of this problem has increased as organizations move from isolated demonstrations toward operational AI services. The competitive growth of LLM platforms demonstrates that model availability is no longer the only differentiating factor. Organizations increasingly need mechanisms for controlling how models operate under application-specific requirements. The expansion of commercial LLM offerings, including subscription-based models and competing AI assistants, reflects the transition toward AI as an operational service rather than solely an experimental technology (Kleinman, 2023; Roth, 2023).

The proposed ScaleCore architecture is positioned within this transition. It complements model-level improvements by introducing systematic constraint reasoning at the orchestration layer. The conceptual direction is also consistent with ScalePulse, which explicitly investigates a combinatorial LLM framework for scalability constraints (Ramamurthy, Bellamkonda and Amanmadov, 2026).

1.4 Objectives

The primary objective is to design a conceptual architecture capable of converting heterogeneous scalability requirements into structured constraint representations and determining compatible configurations before LLM execution.

Secondary objectives are to develop a constraint extraction mechanism, establish a combinatorial

representation of interacting requirements, introduce conflict-resolution and prioritization mechanisms, and define validation stages capable of reducing infeasible configurations.

1.5 Scope and Significance

ScaleCore is designed as an architectural framework rather than a claim of universal optimization. Its scope includes LLM-powered applications in which multiple constraints must be simultaneously satisfied. The architecture can potentially support conversational systems, code-generation environments, enterprise assistants, search interfaces, and other AI services where computational and functional requirements interact.

Its significance lies in reframing scalability from a single-dimensional performance metric into a structured constraint-management problem. This distinction is important because increasing model size or infrastructure capacity can address throughput limitations but cannot inherently resolve contradictory application requirements.

2. LITERATURE REVIEW

The literature supplied for this study primarily documents the rapid development and commercialization of conversational and generative AI. Condon's account of Google's LaMDA illustrates the movement toward conversational language modeling and contextual interaction (Condon, 2021). This development establishes the technological foundation upon which subsequent LLM-based architectures operate.

The emergence of ChatGPT substantially changed the competitive landscape. Grant and Metz characterized the technology as a significant threat to Google's search business, illustrating how conversational AI could challenge established information-access architectures (Grant and Metz, 2022). Olson similarly reported the strategic significance of ChatGPT for Google, while Grant documented Google's mobilization of senior leadership and resources in response to the emerging competitive threat (Olson, 2023; Grant, 2023). These sources collectively demonstrate that LLM scalability is not simply a technical concern; it is connected to organizational responsiveness, infrastructure, and strategic deployment.

Konrad and Cai described the rapid expansion of ChatGPT and the broader race to operationalize AI technologies (Konrad and Cai, 2023). Roth further documented the increasing number of companies attempting to develop or deploy systems capable of competing with ChatGPT (Roth, 2023). This competitive expansion implies an increasingly heterogeneous LLM environment in which different models, interfaces, and deployment configurations must coexist.

David's discussion of interoperability among major AI platforms provides another important architectural perspective. The possibility of interaction between different AI ecosystems emphasizes the importance of composability and standardized coordination mechanisms (David, 2023). A scalable constraint-management layer can therefore be interpreted as an intermediary capable of coordinating requirements across heterogeneous model and infrastructure environments.

However, the literature also identifies risks associated with accelerated deployment. Elias reported concerns among Google executives regarding reputational consequences associated with moving too rapidly on AI-chat technologies (Elias, 2022). Vincent's discussion of Stack Overflow's temporary ban on AI-generated answers demonstrates that generative output may become operationally problematic when quality, accuracy, or contextual validity cannot be adequately controlled (Vincent, 2022). These observations support the need for pre-execution validation and constraint enforcement.

Newton's analysis of Google's acceleration in AI development further demonstrates that organizations respond to competitive pressure by increasing the speed and breadth of AI integration (Newton, 2023). Similarly, the availability of Bard across different contexts demonstrates the expansion of generative AI into practical user-facing environments (Where you can use Bard, 2024).

The principal research gap is therefore not the

absence of LLM capability but the absence of an explicit intermediate architecture for managing interactions among deployment constraints. The supplied literature establishes the technological and strategic motivation, while the ScalePulse framework provides a directly relevant conceptual precedent for combinatorial treatment of scalability constraints (Ramamurthy, Bellamkonda and Amanmadov, 2026). ScaleCore extends this conceptual direction by organizing constraint extraction, normalization, composition, conflict resolution, and validation into a unified architecture.

3. METHODOLOGY

3.1 Research Design

This research adopts a conceptual architecture-development methodology. Rather than evaluating a specific commercial LLM, the study defines a reusable framework for constraint management in LLM-enabled systems. The methodology consists of five logical stages: constraint acquisition, constraint normalization, combinatorial modeling, adaptive resolution, and execution validation.

The architecture assumes that a user or application provides a set of heterogeneous requirements:

[

C = {c_1,c_2,c_3,\ldots,c_n}

]

where each (c_i) represents an individual constraint. Constraints may describe latency, computational budget, response quality, context capacity, security, availability, or application-specific requirements.

3.2 Constraint Acquisition Layer

The first component converts natural-language requirements into structured constraint objects. An input such as “generate a high-quality response within two seconds while minimizing computational cost” contains multiple implicit constraints.

ScaleCore decomposes the request into:

[

C = {Q,L,B}

]

where (Q) represents quality, (L) latency, and (B) computational budget.

The LLM functions as a semantic extraction mechanism, but extracted constraints are not immediately executed. This separation is important because the model's interpretation must first be transformed into an explicit representation.

3.3 Constraint Normalization

Constraints originating from different sources frequently use inconsistent terminology. Normalization maps them into a common representation:

[

c_i=(d_i,t_i,p_i,v_i,w_i)

]

where (d_i) denotes the constraint domain, (t_i) its type, (p_i) its priority, (v_i) its required value or range, and (w_i) its relative importance.

For example, latency may be represented as (L \leq 2s), while cost may be represented as (B \leq \$0.02) per transaction. Normalization enables the architecture to compare otherwise heterogeneous requirements.

3.4 Combinatorial Constraint Model

The central component of ScaleCore is the combinatorial constraint engine. Instead of examining constraints independently, it evaluates combinations.

Given constraints (c_1,c_2,\ldots,c_n), the architecture constructs a compatibility matrix:

[

M_{ij}=f(c_i,c_j)

]

where (M_{ij}) represents the compatibility relationship between two constraints.

A positive relationship indicates that constraints can be jointly satisfied, whereas a negative relationship indicates a potential conflict. For example, low latency and high model complexity may generate a negative interaction because complex inference may require greater computational resources.

The framework can subsequently construct candidate configurations:

[

S_k={c_{i_1},c_{i_2},...,c_{i_m}}

]

and evaluate each candidate against feasibility criteria.

This combinatorial perspective is particularly relevant to scalability because a system's behavior is frequently determined by interactions among requirements rather than isolated constraints. The approach follows the broader direction established by ScalePulse's treatment of scalability constraints through a combinatorial LLM framework (Ramamurthy, Bellamkonda and Amanmadov, 2026).

3.5 Priority and Conflict Resolution

Not every constraint has equal importance. ScaleCore therefore assigns a priority value:

[

P(c_i)\in[0,1]

]

High-priority constraints are protected during optimization, while lower-priority constraints may be relaxed when a complete solution is impossible.

Consider a healthcare-oriented hypothetical assistant requiring high response accuracy, low latency, and limited computational expenditure. If all three constraints cannot simultaneously be satisfied, the architecture should not arbitrarily optimize one variable. Instead, it can classify accuracy as a hard constraint while treating latency and cost as soft constraints.

The optimization objective can consequently be expressed as:

[

\max F(S)=\sum_{i=1}^{n}w_i,s_i

]

subject to hard constraints:

[

g_j(S)\leq 0

]

where (w_i) represents the importance of a soft constraint and (s_i) represents its satisfaction level.

3.6 Adaptive Resource Allocation

After identifying feasible configurations, ScaleCore maps them to execution resources. This layer may select different model sizes, inference modes, context limits, or processing paths according to constraint conditions.

For example, a low-latency request with moderate quality requirements could be routed toward a smaller model, whereas a complex analytical request could receive a larger model and greater context allocation. The architectural objective is not to maximize model capability unconditionally but to select the smallest feasible configuration satisfying the active constraints.

This approach is especially relevant in environments characterized by rapid model competition and expanding AI services, where deployment efficiency becomes a critical operational consideration (David, 2023; Roth, 2023).

3.7 Validation and Feedback

The final component evaluates whether the selected configuration satisfies the original requirements. Validation occurs at three levels: semantic validity, constraint feasibility, and execution compliance.

Semantic validation determines whether extracted constraints accurately represent the original request. Feasibility validation checks whether constraints can be jointly satisfied. Execution validation determines whether actual system behavior conforms to the selected configuration.

If validation fails, the system returns to the constraint-resolution stage rather than automatically producing an output. This feedback loop distinguishes ScaleCore from architectures that treat prompt generation as the final computational step.

4. RESULTS

The conceptual analysis produces four principal findings. First, scalability in LLM environments is inherently multidimensional. The reviewed literature demonstrates that LLM systems are expanding simultaneously across conversational interfaces, search, coding, commercial services, and competing AI ecosystems (Condon, 2021; Grant and Metz, 2022; Roth, 2023). Consequently, a single throughput metric is insufficient for representing deployment scalability.

Second, constraint interactions are more significant than isolated constraints. A system can independently satisfy latency, cost, quality, and context requirements while failing when these requirements are imposed simultaneously. ScaleCore addresses this limitation by representing constraints as combinations rather than independent conditions. This architectural principle is consistent with the combinatorial scalability perspective introduced by ScalePulse (Ramamurthy, Bellamkonda and Amanmadov, 2026).

Third, constraint prioritization provides a mechanism for managing infeasible requirements. Real-world AI deployment frequently involves contradictory objectives. The literature's evidence of concerns regarding reliability and reputational risk demonstrates why unconstrained acceleration can produce undesirable outcomes (Elias, 2022; Vincent, 2022). ScaleCore therefore distinguishes hard constraints from soft constraints and uses priority-aware optimization to identify acceptable configurations.

Fourth, the proposed architecture introduces an explicit separation between interpretation and execution. LLMs are used to understand requirements, but extracted requirements are subjected to structured normalization and validation before system execution. This reduces the risk that ambiguous natural-language requirements directly determine computational behavior.

The resulting architectural flow can be represented as:

Natural-Language Requirements → Constraint Extraction → Normalization → Combinatorial Modeling → Conflict Detection → Priority Resolution → Resource Allocation → Validation → LLM Execution → Feedback

The major finding is that scalability can be improved not only by increasing infrastructure capacity but also by reducing unnecessary computational demand through constraint-aware configuration. This is particularly important as AI providers increasingly compete to expand model capabilities and applications (Newton, 2023; Konrad and Cai, 2023).

Nevertheless, the findings remain conceptual. They establish architectural plausibility rather than empirical superiority. Actual performance would depend on the quality of constraint extraction, the accuracy of compatibility modeling, the efficiency of optimization, and the characteristics of the underlying LLM and infrastructure.

5. DISCUSSION

The principal theoretical contribution of ScaleCore is its reframing of LLM scalability as a constraint-composition problem. Conventional interpretations of scalability often emphasize throughput, infrastructure expansion, or model efficiency. ScaleCore instead argues that computational scalability can be partly achieved through intelligent

configuration of interacting requirements. This distinction is important because simply adding resources may increase capacity without improving the efficiency of individual workloads.

The architecture also provides a conceptual bridge between natural-language reasoning and formal constraint management. LLMs are particularly effective at interpreting ambiguous requirements, but their outputs can be unreliable when used without validation. The concerns surrounding AI-generated information documented in the literature reinforce this limitation (Elias, 2022; Vincent, 2022). ScaleCore therefore assigns the LLM a semantic role while reserving feasibility decisions for a structured constraint layer.

The framework is also consistent with the broader competitive evolution of AI platforms. The rapid development of competing systems and the strategic response of major technology companies demonstrate that organizations must continuously adapt their AI infrastructure (Grant, 2023; Roth, 2023; Newton, 2023). A model-independent constraint layer could potentially reduce dependence on any single model provider by allowing application requirements to remain stable while execution configurations change.

A further implication concerns interoperability. David's discussion of cooperation and interoperability among major AI organizations indicates that heterogeneous AI ecosystems may increasingly interact (David, 2023). ScaleCore's normalized constraint representation could function as an architectural abstraction between different models and services. This would allow application-level requirements to remain consistent even when the underlying execution engine changes.

However, the architecture introduces trade-offs. Constraint extraction itself requires computational resources, particularly when complex requirements are expressed in ambiguous language. Excessive constraint decomposition could also create an optimization problem more complex than the original task. Furthermore, compatibility matrices may become expensive as the number of constraints increases. If (n) constraints are compared pairwise, the basic interaction space grows approximately as:

[

O(n^2)

]

Higher-order combinations may grow substantially faster. Therefore, a practical implementation would require pruning, hierarchical grouping, or approximate search mechanisms.

Another limitation is model dependence. Although ScaleCore separates constraint reasoning from final execution, an LLM still performs much of the initial semantic interpretation. Incorrect extraction can propagate into subsequent stages. This makes validation essential rather than optional.

The relationship with ScalePulse is particularly important. ScalePulse provides a direct conceptual foundation for investigating combinatorial LLM treatment of scalability constraints (Ramamurthy, Bellamkonda and Amanmadov, 2026). ScaleCore builds upon this direction by emphasizing an end-to-end architecture encompassing extraction, normalization, conflict resolution, resource selection, and validation. The distinction is therefore primarily architectural: ScaleCore treats combinatorial constraint reasoning as an intermediate control layer within a broader LLM execution pipeline.

Finally, the literature suggests that scalability cannot be separated from trust. The commercial expansion of LLMs has created pressure to deploy rapidly, while concerns over correctness and reputational consequences demonstrate the need for controlled execution (Elias, 2022). ScaleCore's validation layer therefore represents not merely a performance mechanism but a governance-oriented architectural component.

Future empirical evaluation should compare ScaleCore-enabled systems with conventional prompt-orchestration approaches using metrics such as constraint satisfaction rate, latency, computational cost, optimization overhead, configuration stability,

and failure recovery. Such experiments would establish whether the conceptual advantages translate into measurable operational improvements.

6. CONCLUSION

This paper proposed ScaleCore, an LLM-based combinatorial architecture for scalable constraint management. The framework addresses a fundamental limitation in contemporary LLM deployment: the difficulty of simultaneously satisfying heterogeneous and potentially conflicting requirements involving quality, latency, cost, context, computational capacity, and operational reliability.

The architecture introduces a structured sequence of constraint extraction, normalization, combinatorial modeling, priority assignment, conflict resolution, adaptive resource allocation, and validation. Its central contribution is the treatment of scalability as a multidimensional constraint-composition problem rather than merely an infrastructure-capacity problem.

The analysis of the provided literature demonstrates why such an architecture is increasingly relevant. The rapid expansion of conversational AI, intense competition among AI providers, growth of commercial LLM services, and documented concerns surrounding reliability collectively indicate that future AI systems require stronger mechanisms for controlled deployment (Condon, 2021; Kleinman, 2023; Roth, 2023). The combinatorial perspective is further supported by the ScalePulse framework, which directly motivates systematic reasoning about scalability constraints (Ramamurthy, Bellamkonda and Amanmadov, 2026).

ScaleCore nevertheless remains a conceptual architecture and requires empirical validation. Future research should implement the framework with multiple LLMs and evaluate its effects on latency, cost, constraint satisfaction, resource utilization, and output reliability. Further research should also investigate hierarchical constraint representations, learning-based conflict prediction, cross-model interoperability, and automated policy adaptation.

The broader contribution of ScaleCore is therefore architectural: LLM scalability should be managed not only by making models larger or infrastructure stronger, but by making the interaction between requirements, resources, and model capabilities computationally explicit.

REFERENCES

1. Condon S. Google I/O 2021: Google unveils new conversational language model, LaMDA. Available online: https://www.zdnet.com/article/google-io-google-unveils-new-conversational-language-model-lamda/ (accessed on 13 April 2024).

2. David E. The AI wars might have an armistice deal sooner than expected. Available online:https://www.theverge.com/2023/7/20/23800755/ai-meta-llama-microsoft-chatgpt-interoperability (accessed on 13 April 2024).

3. Elias J. Google execs warn company's reputation could suffer if it moves too fast on AI-chat technology. Available online: https://www.cnbc.com/2022/12/13/google-execs-warn-of-reputational-risk-with-chatgbt-like-tool.html (accessed on 13 April 2024).

4. Grant N. Google Calls in Help from Larry Page and Sergey Brin for A.I. Fight. Available online: https://www.nytimes.com/2023/01/20/technology/google-chatgpt-artificial-intelligence.html (accessed on 13 April 2024).

5. Grant N, Metz C. A New Chat Bot Is a 'Code Red' for Google's Search Business. Available online: https://www.nytimes.com/2022/12/21/technology/ai-chatgpt-google-search.html (accessed on 13 April 2024).

6. Kleinman Z. ChatGPT firm trials $20 monthly subscription fee. Available online: https://www.bbc.com/news/technology-64492750 (accessed on 13 April 2024).

7. Konrad A, Cai K. Inside ChatGPT's Breakout Moment and the Race to Put AI To Work. Available online:https://www.forbes.com/sites/alexkonra

d/2023/02/02/inside-chatggpts-breakout-moment-and-the-race-for-the-future-of-ai/?sh=3f67b566240b (accessed on 13 April 2024).

8. Newton C. How Google is making up for lost time. Available online: https://www.theverge.com/2023/5/12/23721037/google-ai-progress-search-docs-starline-video-calls (accessed on 13 April 2024).

9. Nieva R, Konrad A. Back at Google Again, Cofounder Sergey Brin Just Filed His First Code Request in Years. Available online: https://www.forbes.com/sites/richardnieva/2023/01/31/sergey-brin-code-request-lamda/?sh=6a71df637ce6 (accessed on 13 April 2024).

10. Olson P. Google Faces a Serious Threat from ChatGPT. Available online: https://www.washingtonpost.com/ (accessed on 13 April 2024).

11. Roth E. Meet the companies trying to keep up with ChatGPT. Available online:https://www.theverge.com/2023/3/5/23599209/companies-keep-up-chatgpt-ai-chatbots (accessed on 13 April 2024).

12. Vincent J. AI-generated answers temporarily banned on coding Q&A site Stack Overflow. Available online: https://www.theverge.com/2022/12/5/23493932/chatgpt-ai-generated-answers-temporarily-banned-stack-overflow-llms-dangers (accessed on 13 April 2024).

13. Where you can use Bard. Available online: https://support.google.com/gemini/answer/13575153 (accessed on 13 April 2024).

14. K. Ramamurthy, N. Bellamkonda and N. Amanmadov, "ScalePulse: Combinatorial Llm Framework for Scalability Constraint," SoutheastCon 2026, Huntsville, AL, USA, 2026, pp. 1-6, doi: 10.1109/SoutheastCon63549.2026.11476075

15. Geo Philip, Paulson, Integrated Intelligent Building Energy Management: A Multi-LayerFramework for Renewable Energy, HVAC Optimization, and SmartElectrical Network Coordination. Available at SSRN: https://ssrn.com/abstract=6993209 or http://dx.doi.org/10.2139/ssrn.6993209