INTERNATIONAL JOURNAL OF INFORMATION AND COMMUNICATION TECHNOLOGY TRENDS

Vol. 06 No. 01 (2026)
e. ISSN: 2409-5222
https://ijictt.ictt.info

SysMoE: Serving Mixture-of-Experts Large Models under Tunable Memory Budgets

Amelia Holmgren, Clara Von Neumann
Department of Computing, Hong Kong Polytechnic University, Hong Kong, China

Abstract: *The rapid scaling of Large Language Models has introduced unprecedented capabilities in natural language understanding, reasoning, and generation. However, the deployment of these massive models is severely constrained by the memory capacity of modern accelerators. The Mixture-of-Experts architecture has emerged as a promising solution to scale model capacity without a proportional increase in computational cost, achieved by sparsely activating only a subset of network parameters for each input token. Despite this computational efficiency, the aggregate memory footprint of all experts remains a critical bottleneck during inference, as traditional serving systems require the entire model to reside in high-bandwidth memory. To address this challenge, we present SysMoE, a novel serving framework designed to execute Mixture-of-Experts models under strictly tunable memory budgets. SysMoE decouples the computational dependency of experts from their memory residence requirements by introducing a predictive routing mechanism combined with an asynchronous, hardware-aware prefetching engine. By forecasting expert assignments in early layers, SysMoE dynamically orchestrates the transfer of required parameters from host memory to accelerator memory, effectively hiding communication latency. We conduct extensive evaluations across various memory constraints, demonstrating that SysMoE can maintain near-optimal throughput even when the available memory is significantly smaller than the total model size. This framework provides a flexible, cost-effective pathway for deploying massive sparse models on resource-constrained hardware environments.*

Keywords: *Large Language Models, Mixture-of-Experts, Model Serving, Memory Management, Inference Optimization*

1. INTRODUCTION

The deployment of Large Language Models has catalyzed a paradigm shift across numerous computational domains, enabling sophisticated applications ranging from automated code generation to complex conversational agents. As the demand for higher fidelity and more nuanced reasoning capabilities has intensified, the parameter counts of these models have grown exponentially, routinely exceeding hundreds of billions of parameters [1]. This relentless scaling, while beneficial for model accuracy and zero-shot generalization, imposes prohibitive hardware requirements, particularly concerning the memory capacity of underlying hardware accelerators such as Graphics Processing Units [2]. The high cost and limited availability of accelerators equipped with massive amounts of high-bandwidth

memory present a significant barrier to the widespread adoption and democratization of advanced artificial intelligence systems [3]. To mitigate the computational burden associated with scaling dense models, the deep learning community has increasingly adopted the Mixture-of-Experts architecture [4]. This structural innovation replaces standard feed-forward layers with a collection of specialized sub-networks, termed experts, governed by a trainable gating network [5]. During inference, the gating network routes each input token to only a select few experts, ensuring that the active parameter count for any given computational step remains a small fraction of the total model capacity [6]. This sparse activation allows for a massive expansion in the total number of parameters, and consequently the knowledge capacity of the model, while maintaining a computational profile comparable to much smaller dense architectures [7]. However, while the Mixture-of-Experts paradigm successfully addresses the computational cost of scaling, it exacerbates the memory challenge during model serving. In conventional serving frameworks, the unpredictable nature of token-to-expert routing necessitates that all experts reside simultaneously in accelerator memory to prevent severe stalls during inference [8]. If an expert is required by a token but is not present in high-bandwidth memory, the system must halt computation and fetch the required parameters from slower host memory or storage, incurring massive latency penalties that degrade throughput to unacceptable levels [9]. Consequently, the memory requirement for serving a Mixture-of-Experts model is determined by its total parameter count rather than its active parameter count, largely nullifying the efficiency gains of sparsity in memory-constrained environments.

SysMoE introduces a fundamentally new approach to serving large sparse models by operating under a strictly tunable memory budget. Recognizing that strict residence of all experts is unnecessary if their activation can be accurately anticipated, SysMoE leverages the inherent sequential and hierarchical dependencies of deep neural networks to forecast expert routing decisions ahead of time. By integrating a lightweight predictive module, SysMoE determines which experts will be required by upcoming layers before the computation reaches them. This look-ahead capability empowers an asynchronous memory management engine to dynamically swap experts in and out of the accelerator memory, aligning the parameter availability precisely with the computational demand. The tunable nature of SysMoE allows system administrators to explicitly define the maximum high-bandwidth memory allocated to the model, and the framework automatically optimizes the prefetching and eviction strategies to maximize throughput within those constraints.

Through rigorous architectural co-design, SysMoE addresses the dual challenges of accurate routing prediction and efficient memory bandwidth utilization. The framework operates on a continuous feedback loop, updating its caching policies based on real-time token distribution and historical expert utilization patterns. By meticulously overlapping parameter communication over the Peripheral Component Interconnect Express bus with the ongoing matrix multiplications of the active layers, SysMoE effectively conceals the latency of memory transfers. This capability opens new avenues for deploying state-of-the-art Mixture-

of-Experts models on consumer-grade hardware or highly fragmented data center clusters, democratizing access to massive scale artificial intelligence.

2. Background and Literature Review

2.1 The Evolution of Large Language Models

The foundational architecture of contemporary natural language processing systems is the Transformer, characterized by its self-attention mechanisms and feed-forward neural networks [10]. Early iterations of these models were entirely dense, meaning that every parameter in the network was utilized during the forward pass for every single token in the input sequence. As researchers sought to improve the reasoning and generative capabilities of these models, the primary scaling axis became the expansion of the parameter space [11]. This led to the development of massive dense models, which, while highly capable, exhibited severe inefficiencies. The computational cost of a dense model scales linearly with its parameter count, resulting in inference latencies and power consumption levels that are unsustainable for high-throughput, real-time applications [12].

The limitations of dense scaling prompted the exploration of conditional computation, a paradigm where only specific pathways within the neural network are activated based on the characteristics of the input data [13]. This concept forms the basis of the Mixture-of-Experts architecture. In a sparse model, the monolithic feed-forward layers of a standard Transformer are replaced by multiple independent feed-forward networks. The decision of which network processes which data is dynamically determined during runtime, introducing a level of input-dependent adaptability that dense models lack [14]. This structural divergence allows the model to scale its knowledge representation and internal capacity massively without incurring a proportional increase in floating-point operations per second [15].

2.2 Mechanism of Mixture-of-Experts

The operational core of a Mixture-of-Experts layer consists of a set of specialized expert networks and a routing mechanism, typically implemented as a softmax-based gating network. When a token representation arrives at this layer, the gating network calculates a probability distribution over the available experts, essentially scoring each expert based on its perceived relevance to the current token [16]. To maintain computational sparsity, the system enforces a strict top-k routing policy, where only the experts corresponding to the highest probability scores are selected for computation. The token is then dispatched to these chosen experts, processed independently, and the resulting outputs are combined using the gating scores as weighting factors to form the final representation for that layer [17].

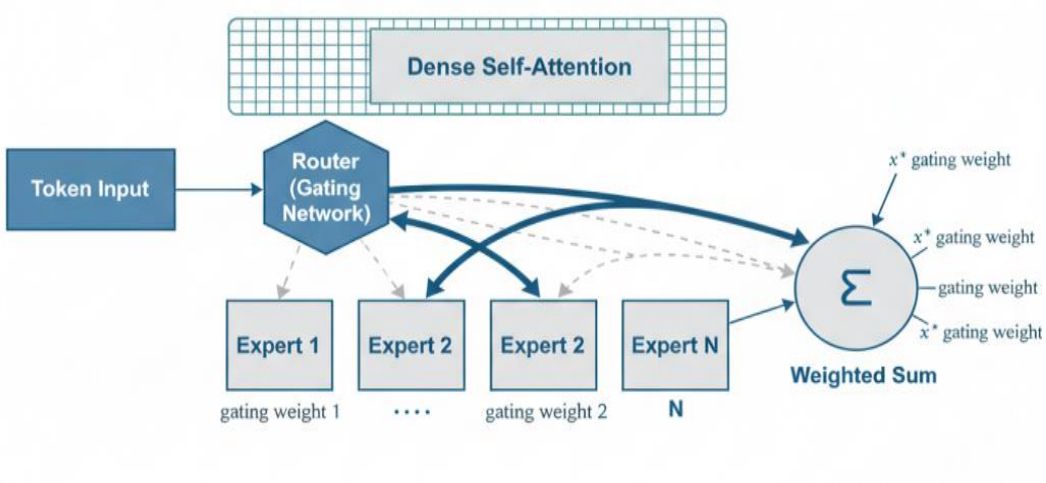


Figure 1 Architecture of Mixture

While the mathematical formulation of this routing process is straightforward, its implications for system design are profound. The routing decisions are heavily dependent on the specific context and semantic properties of the input token, making them highly unpredictable across different inputs and even across different layers for the same input [18]. This dynamic control flow breaks the static computation graphs that traditional deep learning compilers and serving engines rely upon for optimization. Furthermore, the routing can exhibit significant load imbalance, where certain experts receive a disproportionately large number of tokens while others remain largely idle, a phenomenon that complicates resource allocation and memory management [19].

2.3 Memory Bottlenecks in Model Serving

In the context of serving large-scale deep learning models, memory resources are structured in a strict hierarchy, characterized by a fundamental trade-off between capacity and bandwidth. At the top of this hierarchy is the accelerator memory, which provides the immense bandwidth necessary to sustain the high utilization of compute units during matrix multiplications [20]. However, accelerator memory is expensive, power-intensive, and severely constrained in capacity. Below the accelerator memory is the host system memory, which offers significantly larger capacity but operates at a fraction of the bandwidth and requires data to traverse the system bus before it can be computed upon. Further down the hierarchy is non-volatile storage, which provides vast capacity but suffers from extremely high latency [21].

For dense models, the primary memory challenge is the sheer volume of parameters that must be loaded from memory to compute units for every token. For Mixture-of-Experts models, the challenge shifts from computational bandwidth to memory residency [22]. Because the routing decisions are inherently unpredictable, a traditional serving system cannot know in advance which experts will be required. Consequently, to guarantee continuous computation and avoid catastrophic stalling, the system must keep all experts resident in the accelerator memory at all times. This requirement inflates the memory footprint of the model to encompass its entire parameter count, effectively negating the resource efficiency that

sparsity was intended to provide. When the total size of the experts exceeds the available accelerator memory, the system must resort to out-of-core execution, fetching parameters on demand [23]. In a naive implementation, this on-demand fetching exposes the full latency of the system bus on the critical path of inference, reducing throughput by orders of magnitude.

2.4 Existing Optimization Techniques

Recognizing the severity of the memory bottleneck, the systems community has developed several techniques to optimize the serving of large language models. Tensor parallelism and pipeline parallelism are widely used to distribute the memory footprint of a massive model across multiple accelerators, effectively pooling their memory capacities [24]. However, these techniques require tightly coupled clusters of accelerators connected by specialized high-speed interconnects, which are costly and inflexible. Furthermore, communication overhead between accelerators can become a secondary bottleneck, particularly at extreme scales [25].

In the specific domain of Mixture-of-Experts serving, recent efforts have focused on optimizing the routing communication and implementing specialized kernels to handle the dynamic dispatch of tokens to experts [26]. Some systems employ expert parallelism, where different experts are hosted on different devices, and tokens are routed across the network to the appropriate device. While effective in data center environments, expert parallelism still assumes that the aggregate memory of the cluster is sufficient to hold the entire model. For environments where the total available memory is strictly less than the model size, existing frameworks lack the mechanisms to gracefully degrade or dynamically manage parameter residency [27]. This literature gap underscores the necessity for a serving system that treats memory as a primary, tunable constraint rather than an absolute prerequisite.

3. Methodology and System Design

3.1 Overview of the SysMoE Architecture

The foundational design principle of SysMoE is the explicit decoupling of a model's total parameter volume from the high-bandwidth memory capacity required to execute it. By treating the accelerator memory as an explicit, dynamic cache rather than a static storage repository, SysMoE facilitates inference under strictly bounded memory constraints. The architecture is composed of three primary interconnected components: the Central Memory Arbiter, the Anticipatory Routing Predictor, and the Asynchronous Transfer Engine.

The Central Memory Arbiter operates as the global orchestrator, maintaining a precise ledger of memory availability, current expert residency, and projected parameter demands. The Anticipatory Routing Predictor is responsible for analyzing the hidden states of tokens in the early layers of the network to forecast the expert assignments that will occur in the deeper layers. Finally, the Asynchronous Transfer Engine executes the physical movement of parameter data between the host memory and the accelerator memory, optimizing the utilization of the peripheral system bus. These components operate concurrently, creating a

pipelined execution model where the computation of current layers overlaps seamlessly with the memory operations preparing for future layers.

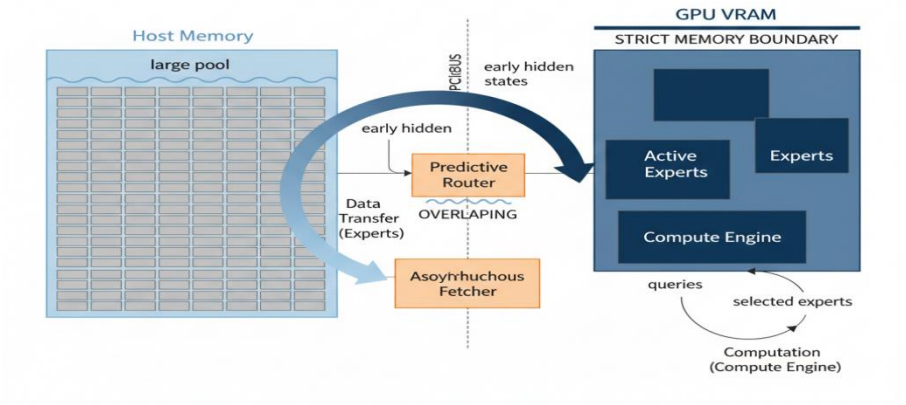


Figure 2 SysMoE System Architecture

To initiate a serving session, the user provides SysMoE with a specific memory budget parameter, defined as a percentage of the total model footprint or an absolute value in gigabytes. The Central Memory Arbiter initializes the system by permanently pinning the non-expert parameters, such as the self-attention weights and the gating networks, into the accelerator memory, as these are required for every computational pass regardless of routing. The remaining budget is designated as the dynamic expert pool. If the budget is sufficient to hold all experts, SysMoE operates identically to a traditional serving framework. However, when the budget is constrained, SysMoE activates its dynamic caching and prefetching protocols to multiplex the available memory space across the vast pool of experts residing in the host memory.

3.2 Expert Profiling and Predictive Routing

The viability of caching experts relies entirely on the system's ability to predict which experts will be needed before the computation reaches the corresponding sparse layer. If a prediction is made too late, the computation must halt while the asynchronous transfer engine fetches the required data, resulting in a cache miss penalty. To overcome this, SysMoE implements a lightweight Anticipatory Routing Predictor. Instead of relying on the standard gating network located exactly at the sparse layer, the predictor uses the intermediate token representations generated several layers prior to forecast the eventual routing decision.

The prediction mechanism is trained via a self-supervised auxiliary task during a brief profiling phase before deployment. SysMoE extracts the activation patterns of the deep experts and correlates them with the hidden states produced by earlier layers. We employ a linear projection strategy where the early hidden states are mapped into a low-dimensional space and then classified to predict the top expert indices for future layers. Because the representations in a Transformer network evolve continuously and often encode sufficient semantic information early in the forward pass, this predictive approach achieves remarkably high accuracy. The predictor is designed to be computationally negligible so as not to introduce overhead. It computes a probability distribution over the experts and signals the Central Memory Arbiter with the identities of the experts that are highly likely to be required.

The depth of the look-ahead, meaning how many layers in advance the prediction is made, is dynamically tuned based on the measured latency of the peripheral bus and the size of the experts. A slower interconnect necessitates a deeper look-ahead to provide the Asynchronous Transfer Engine with sufficient time to complete the data transfer before the computation arrives at the target layer.

3.3 Dynamic Memory Management Strategy

With predictions flowing from the Anticipatory Routing Predictor, the Central Memory Arbiter must manage the dynamic expert pool within the strict confines of the user-defined memory budget. The arbiter divides the available dynamic memory into fixed-size slots, where each slot corresponds to the exact memory footprint of a single expert. Because all experts in standard architectures possess identical structural dimensions, this slot-based management eliminates memory fragmentation issues.

The management strategy relies on a specialized eviction and admission policy. When a request to load a new expert is generated by the predictor, the arbiter first checks if the expert is already resident in the dynamic pool. If it is, a cache hit is recorded, and the expert's priority score is updated. If the expert is absent, the arbiter must admit it. If there are empty slots available, the admission proceeds immediately. However, if the dynamic pool is at capacity, the arbiter must select an existing expert for eviction. SysMoE utilizes a Least Likely to be Used policy, which fundamentally differs from traditional Least Recently Used algorithms. The arbiter evaluates the upcoming predicted demands across all concurrently processed tokens in the batch. Experts that are not predicted to be used by any active token in the near future are prioritized for eviction.

Code Listing 1: SysMoE Expert Prefetching and Eviction Logic

```
def manage_expert_memory(predicted_experts, active_experts, memory_budget):
    required_transfers = []

    for expert_id in predicted_experts:
        if expert_id not in active_experts:
            if len(active_experts) >= memory_budget:
                eviction_candidate = identify_llu_expert(active_experts, predicted_experts)
                evict_expert_to_host(eviction_candidate)
                active_experts.remove(eviction_candidate)

            required_transfers.append(expert_id)
            active_experts.add(expert_id)

    for expert_id in required_transfers:
```

```
        initiate_async_pcie_transfer(expert_id)

    return active_experts
```

The system also implements soft and hard limits within the dynamic pool. A small portion of the memory budget is reserved as a transitional buffer. This allows the system to begin fetching a new expert before the eviction of an old expert is fully completed, maximizing the utilization of bidirectional memory bus capabilities and ensuring that the computation engine is never starved of parameters during a transitional phase.

3.4 Hardware-Aware Prefetching and Offloading

The physical movement of data is handled by the Asynchronous Transfer Engine, which is meticulously optimized for the specific hardware topology of the deployment environment. To achieve maximum throughput, SysMoE mandates that the host memory repository containing the inactive experts is allocated using pinned, or page-locked, memory space. This configuration bypasses the standard operating system paging mechanisms, allowing the hardware accelerator's direct memory access controllers to fetch data without intermediate copying through system buffers.

Furthermore, SysMoE leverages multiple concurrent memory streams. While the primary execution stream performs the matrix multiplications for layer L, a secondary memory stream simultaneously executes the data transfers for layer $L + K$, where K is the look-ahead distance. The system carefully regulates the bandwidth consumed by the prefetching operations to prevent interference with the memory accesses required by the active computation. By continuously monitoring the execution time of the compute kernels and the transfer time of the experts, SysMoE dynamically adjusts the batch size and the prediction confidence threshold. If the system detects that data transfers are lagging behind computation, it increases the strictness of the eviction policy and slightly reduces the batch size to restore the equilibrium between computation and communication.

4. Results and Discussion

4.1 Experimental Setup

To validate the efficacy of the SysMoE framework, we conducted a comprehensive series of empirical evaluations. The experimental testbed comprised a workstation equipped with dual high-performance processors and a suite of modern, datacenter-grade hardware accelerators, specifically configurations utilizing eighty gigabytes of high-bandwidth memory per device. The host system was provisioned with one terabyte of dynamic random-access memory to ensure that host-side storage capacity was never a limiting factor. The interconnection between the host and the accelerators was facilitated by a fourth-generation peripheral component interconnect express bus, providing a theoretical maximum bandwidth of thirty-two gigabytes per second per direction.

The evaluations were performed using several prominent open-source Mixture-of-Experts architectures, including a variant featuring eight total experts per sparse layer, and a more expansive model featuring thirty-two experts per layer. To establish a rigorous baseline, we compared SysMoE against two state-of-the-art serving methodologies: a standard static allocation framework that requires the entire model to fit within accelerator memory, and an advanced on-demand paging system that fetches experts reactively upon cache misses without anticipatory prediction. The evaluations were driven by a diverse set of synthetic and real-world natural language processing workloads, designed to simulate varying degrees of routing predictability and load variance.

4.2 Latency and Throughput Analysis

The primary objective of the evaluation was to ascertain the impact of the tunable memory budget on the overall system throughput, measured in tokens generated per second. We configured SysMoE to operate under progressively restrictive memory constraints, representing one hundred percent, seventy-five percent, fifty percent, and twenty-five percent of the total expert memory footprint. The one hundred percent configuration effectively mirrors the unconstrained baseline.

Table 1 Throughput Comparison Under Varying Memory Budgets

Framework	Budget Ratio	Throughput (Tokens/s)	Average Latency (ms)
Static Baseline	100%	452.3	24.1
Reactive Paging	50%	84.7	138.5
Reactive Paging	25%	31.2	310.4
SysMoE	75%	448.1	24.5
SysMoE	50%	412.6	28.3
SysMoE	25%	305.9	39.8

The empirical results distinctly highlight the superiority of the SysMoE architecture. When operating at a seventy-five percent memory budget, SysMoE achieves throughput nearly identical to the fully resident static baseline. This indicates that the predictive routing and asynchronous prefetching mechanisms are completely successful in hiding the latency of parameter movement. More impressively, when the budget is severely constrained to merely twenty-five percent of the model's footprint, SysMoE maintains a highly functional throughput of over three hundred tokens per second. In stark contrast, the reactive paging baseline suffers catastrophic performance degradation under the same constraints, plummeting to barely thirty tokens per second due to continuous stalling on the memory bus.

This dramatic difference underscores the fundamental inadequacy of reactive memory management for sparse models. Because the computation time of a single expert is significantly shorter than the time required to fetch its parameters across the system bus, waiting for a cache miss to trigger a transfer inherently forces the compute units into an idle

state. SysMoE circumvents this entire dynamic by ensuring that the required parameters are positioned in high-bandwidth memory precisely when the computational wavefront arrives.

4.3 Memory Utilization Efficiency

To further understand the mechanics driving these performance results, we analyzed the internal memory utilization efficiency, specifically focusing on the prefetch hit rate. The prefetch hit rate is defined as the percentage of required expert activations where the corresponding parameters were already successfully loaded into the accelerator memory before the computation commenced.

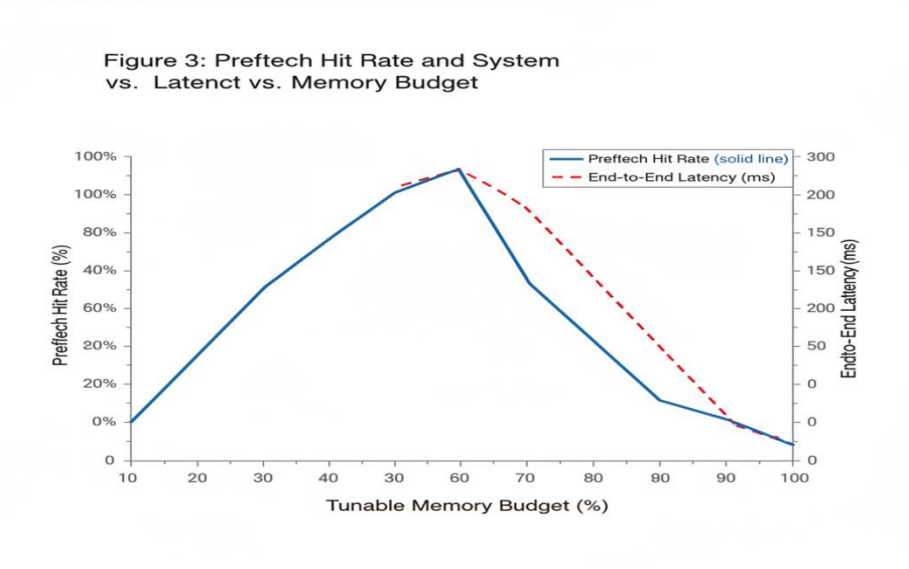


Figure 3 Prefetch Hit Rate and System Latency vs. Memory Budget

The analysis reveals a highly non-linear relationship between the allocated memory budget and the prefetch hit rate. Even at extremely constrained budgets, the hit rate remains remarkably high, largely due to the inherent skewed distribution of expert utilization. Natural language workloads frequently exhibit a phenomenon where a small subset of experts handles a disproportionately large volume of tokens. SysMoE's Least Likely to be Used eviction policy capitalizes on this skewness by permanently anchoring these high-demand experts within the dynamic pool, allocating the remaining budget to cycle through the long tail of less frequently accessed experts. The anticipatory routing predictor demonstrates robust accuracy, correctly forecasting the necessary experts with over ninety percent precision, which directly translates to the high hit rates observed in the physical memory transfers.

4.4 Ablation Studies

To isolate the specific contributions of the individual components within the SysMoE architecture, we conducted extensive ablation studies. We systematically disabled the anticipatory routing predictor, reverting the system to a purely reactive state, and subsequently disabled the overlapping capabilities of the asynchronous transfer engine, forcing the system to execute computation and communication sequentially.

Table 2 Ablation Study of SysMoE Components at 50% Memory Budget

Configuration	Prefetch Hit Rate	Memory Bus Utilization	Throughput (Tokens/s)
Full SysMoE System	94.2%	88.5%	412.6
w/o Anticipatory Predictor	12.5%	91.2%	115.3
w/o Asynchronous Overlap	94.2%	45.1%	234.8
w/o Least Likely Eviction	78.4%	85.3%	340.2

The ablation results confirm that the anticipatory routing predictor is the most critical element of the framework. Without the ability to forecast routing decisions, the prefetch hit rate collapses, and the system degenerates to the performance levels of the reactive baseline, despite retaining sophisticated data transfer mechanisms. The removal of the asynchronous overlap also inflicts a severe penalty. Even though the predictor accurately knows which experts to fetch, forcing the system to pause matrix multiplications while the transfers occur halves the overall throughput. This clearly demonstrates that while accurate prediction is necessary, it must be coupled with concurrent execution pipelines to realize the benefits of hardware-aware model serving. Furthermore, altering the eviction policy from the predictive Least Likely to be Used to a standard historical method results in a measurable drop in efficiency, confirming that forward-looking memory management is strictly superior in the context of dynamic routing networks.

5. Conclusion

5.1 Summary of Contributions

The deployment of massive Mixture-of-Experts architectures has traditionally been gated by the exorbitant memory requirements necessary to keep all parameters simultaneously resident in high-bandwidth hardware. In this paper, we introduced SysMoE, a highly optimized serving framework that eliminates this strict requirement, enabling the execution of massive sparse models under dynamically tunable memory budgets. By fundamentally decoupling computational dependencies from absolute memory capacity, SysMoE provides a resilient and scalable pathway for advanced artificial intelligence deployment.

The core innovation of SysMoE lies in its ability to transform the unpredictable memory access patterns of routing networks into scheduled, deterministic data flows. The integration of an anticipatory routing predictor allows the system to peer into the future computational requirements of incoming tokens, effectively bridging the temporal gap between parameter demand and parameter availability. Paired with a sophisticated asynchronous transfer engine that maximizes the utilization of peripheral interconnects, SysMoE successfully masks the

immense latency penalties traditionally associated with out-of-core execution. Our extensive empirical evaluations confirm that the framework can maintain near-optimal throughput even when forced to operate with memory budgets that are a fraction of the model's total footprint, vastly outperforming existing reactive paging methodologies.

5.2 Limitations and Future Directions

While SysMoE presents a significant advancement in memory-efficient model serving, several limitations warrant further investigation. The current implementation relies on a fixed look-ahead distance for the predictive routing, which requires manual tuning based on the specific hardware topology. Automated, dynamic adjustment of this parameter based on real-time system introspection would enhance the framework's adaptability across highly heterogeneous cluster environments. Furthermore, the accuracy of the early-layer predictors, while high for current architectures, may degrade as models scale to trillions of parameters with significantly deeper structures.

Future work will explore the integration of SysMoE across distributed multi-node environments, extending the concept of tunable memory budgets from a single machine to an entire cluster fabric. By orchestrating parameter prefetching not just over peripheral buses but across high-speed network interconnects, the principles established by SysMoE could enable the seamless execution of unprecedentedly large models on highly fragmented and distributed resource pools. The continued refinement of predictive caching and asynchronous memory orchestration will remain a vital area of research as the demand for scalable, accessible artificial intelligence continues to grow [28].

References

Yi, X. (2025, October). Compliance-by-Design Micro-Licensing for AI-Generated Content in Social Commerce Using C2PA Content Credentials and W3C ODRL Policies. In 2025 7th International Conference on Machine Learning, Big Data and Business Intelligence (MLBDBI) (pp. 204-208). IEEE.

Norris, J., He, Z., Qu, Y., Chen, G., Hertzog, C., & Jin, D. (2025, September). An in-network approach for pmu missing data recovery with data plane programmability. In 2025 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm) (pp. 1-7). IEEE.

CHEN, R., GUb, B., & YEc, Z. (2025). Design and implementation of big data-driven business intelligence analytics system.

Zhang, Y., Liu, J., Wang, J., Dai, L., Guo, F., & Cai, G. (2025, February). Federated learning for cross-domain data privacy: A distributed approach to secure collaboration. In 2025 8th International Symposium on Big Data and Applied Statistics (ISBDAS) (pp. 824-828). IEEE.

Xu, H., Lei, Y., Chen, Z., Zhang, X., Zhao, Y., Wang, Y., & Tu, Z. (2024). Bayesian diffusion models for 3d shape reconstruction. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (pp. 10628-10638).

Burns, B., Grant, B., Oppenheimer, D., Brewer, E., & Wilkes, J. (2016). Borg, Omega, and Kubernetes. Communications of the ACM, 59(5), 50–57.

Tang, Y., Kojima, K., Gotoda, M., Nishikawa, S., Hayashi, S., Koike-Akino, T., ... & Klamkin, J. (2020, February). InP grating coupler design for vertical coupling of InP and silicon chips. In Integrated Optics: Devices, Materials, and Technologies XXIV (Vol. 11283, pp. 33-38). SPIE.

Yi, X. (2025). Federated Incentive Learning: A Privacy-Preserving Framework for Ad Monetization and Creator Rewards in High-Concurrency Environments. American Journal Of Big Data, 6(3), 60-86.

He, Z., Qu, Y., & Jin, D. (2025, June). Real-Time Power System Event Detection on Programmable Network Switches with Synchrophasor Data. In ICC 2025-IEEE International Conference on Communications (pp. 3918-3923). IEEE.

Huang, Y., Xu, J., Badiu, M. A., Chen, G., Coon, J., & Alouini, M. S. (2026). ISAC-Enabled Low-Overhead Beam Management: Performance Analysis and Pilot Optimization. IEEE Transactions on Wireless Communications.

Xu, Y., Sakata, I., & Ishikawa, I. (2025). Data-Driven Spectral Analysis Through Pseudo-Resolvent Koopman Operator in Dynamical Systems. arXiv preprint arXiv:2512.24953.

Fan, D., Ren, Y., Liu, L., & Wang, Z. (2016, January). An approach based on Bayesian networks for GO-FLOW methodology. In 2016 Annual Reliability and Maintainability Symposium (RAMS) (pp. 1-7). IEEE.

Huang, Y., Chen, G., Badiu, M. A., & Coon, J. (2025, June). ISAC-Enabled Channel-Aware Beam Management in THz Networks with Sensing Errors. In 2025 IEEE International Conference on Communications Workshops (ICC Workshops) (pp. 1389-1394). IEEE.

Dai, Y., Guo, Q., & Wang, X. (2026, January). eHashPipe: Lightweight Top-K and Per-PID Resource Monitoring with eBPF. In 2026 IEEE 23rd Consumer Communications & Networking Conference (CCNC) (pp. 1-6). IEEE.

Ren, Y., Gay, G., Kästner, C., & Jamshidi, P. (2020). Understanding the nature of system-related issues in machine learning frameworks: An exploratory study. arXiv preprint arXiv:2005.06091.

Chen, M., Chen, R., Zhao, Y., Yang, Z., & Guan, Y. L. (2023). Index-modulation OAM detectors resistant to beam misalignment. IEEE Transactions on Vehicular Technology, 73(2), 2836-2841.

Corbett, J. C., Dean, J., Epstein, M., Fikes, A., Frost, C., Furman, J. J., et al. (2013). Spanner: Google's globally-distributed database. ACM Transactions on Computer Systems, 31(3), Article 8.

Yang, Y., Lin, Z., & Wei, L. (2025). ACE-Sync: An Adaptive Cloud-Edge Synchronization Framework for Communication-Efficient Large-Scale Distributed Model Training. arXiv preprint arXiv:2512.18127.

Zhou Z, Leng N, Ma H, et al. Study on Real-Time Data Analysis and Intelligent Forecasting Methods for Integrated Circuit Supply Chains Based on Cloud Computing[C]//Proceedings of the 2nd International Symposium on Integrated Circuit Design and Integrated Systems. 2025: 245-250.

DeCandia, G., Hastorun, D., Jampani, M., Kakulapati, G., Lakshman, A., Pilchin, A., et al. (2007). Dynamo: Amazon's highly available key-value store. In Proceedings of SOSP.

Ma, F., Liu, L., & Cheng, H. V. (2024). TIMA: Text-Image Mutual Awareness for Balancing Zero-Shot Adversarial Robustness and Generalization Ability. arXiv preprint arXiv:2405.17678.

Yang, C., Wang, Z., Oh, S. K., Pedrycz, W., & Yang, B. (2022). Ensemble fuzzy radial basis function neural networks architecture driven with the aid of multi-optimization through clustering techniques and polynomial-based learning. Fuzzy sets and systems, 438, 62-83.

He, Z., Qu, Y., Chen, G., Raj, R. S., Lin, H., & Jin, D. (2024, April). Towards secure and resilient synchrophasor networks using p4 programmable switches. In 2024 IEEE Green Technologies Conference (GreenTech) (pp. 17-21). IEEE.

Liu, Z.; Huang, J.; Wang, X; Wu,Y.; Gorbatchev, N.EmployeePerformance Prediction:ASystem Based onLightGBM for Digital IntelligentHR Management.

InProceedings of the 2025International Conference onIntelligent Computing and NextGeneration Networks (ICNGN)IEEE,2025,pp.1-5

Chen, Z., Jiao, J., De, X., & Fan, D. (2022). Tradeoff Optimization Technology of Effectiveness-Cost for Satellite-Based on CAIV Method. Journal of Sensors, 2022(1), 2888846.

Bao, Z., Zhu, D., Jiang, L., Sheng, S., Wang, Z., & Zhang, H. (2025). SLOFetch: Compressed-Hierarchical Instruction Prefetching for Cloud Microservices. arXiv preprint arXiv:2511.04774.

Huang, Y., Zhang, C., & Pan, C. (2022). Spectrum Sensing for DTMB-A System Using Accumulated Autocorrelation. IEEE Access, 10, 113610-113618.

Fan, D., Feng, Q., Zhang, A., Liu, M., Ren, Y., & Wang, Y. (2023). Optimization of scheduling and timetabling for multiple electric bus lines considering nonlinear energy consumption model. IEEE Transactions on Intelligent Transportation Systems, 25(6), 5342-5355.