Taming the Titans: A Survey of Efficient LLM Inference Serving

Anonymous ACL submission

Abstract

Large Language Models (LLMs) for Generative AI have achieved remarkable progress, evolving into sophisticated and versatile tools widely adopted across various domains and applications. However, the substantial memory overhead caused by their vast number of parameters, combined with the high computational demands of the attention mechanism, poses significant challenges in achieving low latency and high throughput for LLM inference services. This paper presents a comprehensive survey of recent advances in LLM inference optimization, categorized into: (1) fundamental techniques (model placement, request scheduling, decoding prediction, storage management, disaggregation, and multiplexing); (2) specialized architectures (MoE, LoRA, and speculative decoding); and (3) scenario-specific optimizations (long-context problem, RAG, Augmented LLMs, test-time reasoning, and multimodal integration). Finally, we outline potential research directions to further advance the field of LLM inference serving.

1 Introduction

With the rapid evolution of open-source Large Language Models (LLMs), weekly updates to model architectures and capabilities have become the norm in recent years. The surging demand for these models is evident from Huggingface download statistics, which range from hundreds of thousands for models like Mistral-Small-24B-Instruct-2501 (Mistral, 2025), phi-4 (Abdin et al., 2024), and Llama-3.3-70B-Instruct (Grattafiori et al., 2024) to millions for DeepSeek-V3 (DeepSeek-AI et al., 2024) and DeepSeek-R1 (DeepSeek-AI et al., 2025) over recent months. However, when deploying these models, their large-scale parameters and attention mechanisms impose substantial demands on memory and computational resources, presenting significant obstacles to achieving the desired low latency and high throughput in processing requests. These challenges have spurred extensive research across multiple domains of inference serving optimization to meet Service Level Objectives (SLOs).

This paper presents a systematic survey of LLM inference serving methods, organized hierarchically from fundamental techniques to specialized architectures and specific scenarios, as illustrated in Figure 1.

Fundamental Techniques optimization (§3) begins with model placement (§3.1), essential for distributing parameters across devices when single-GPU memory is insufficient. Subsequent request scheduling (§3.2) prioritizes batched processing through decoding length prediction (§3.3), where shorter requests are prioritized to reduce overall latency. Dynamic batch management then governs request insertion/eviction during iterative processing. While KV cache (§3.4) mitigates redundant computation, challenges persist in storage efficiency, reuse strategies, and compression. Due to the distinction between the prefill and decoding phases, the disaggregated architecture (§3.5) was introduced, facilitating the optimization of each phase. And how multiple LLMs efficiently share the same computing resources through multiplexing (§3.6).

Special Architecture (§4) including Mixture of Experts (MoE) (§4.1), which involves challenges such as expert placement, load balancing, and All-to-All communication. It also includes Low-Rank Adaptation (LoRA) (§4.2) and Speculative Decoding (§4.3), all of which require adaptability to address evolving demands.

Specific Scenario (§5) include advanced tasks such as Long Context processing (§5.1), Retrieval-Augmented Generation (RAG) (§5.2), Augmented LLMs (§5.2), Test-Time Reasoning (§5.4), and Multimodal (§5.5). These scenarios demand high adaptability to effectively address evolving requirements.

While previous work (Miao et al., 2023; Yuan et al., 2024; Zhou et al., 2024; Li et al., 2024a)

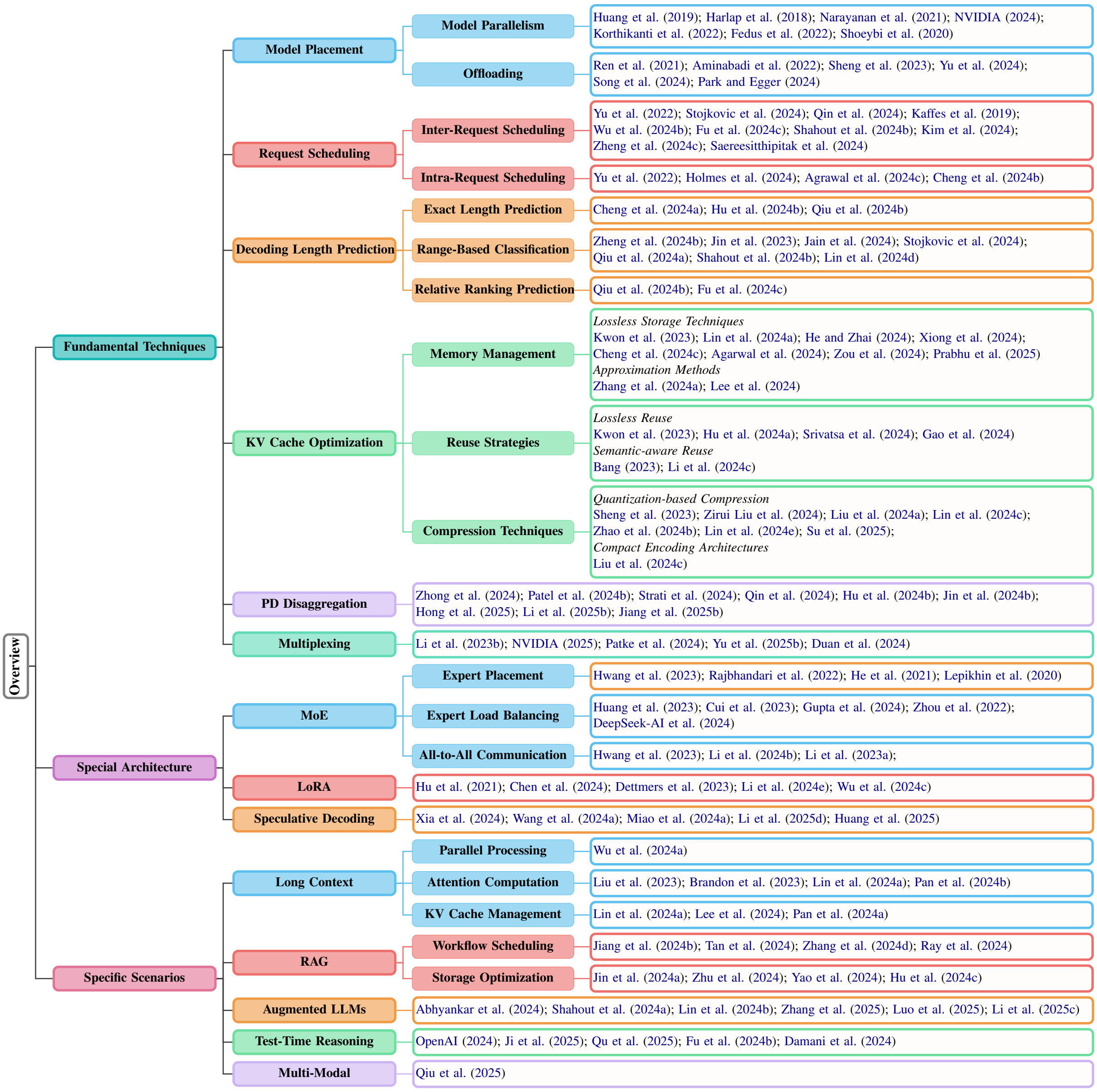

Figure 1: Taxonomy of approaches for LLM inference serving.

has built a strong foundation, the rapid development of the field calls for an urgent summary of emerging methods and scenarios to support deeper research. Our work integrates a wide range of both classical and cutting-edge methods, and adopts a forward-looking perspective and highlights several promising directions for future research. For completeness, we also include cloud-level topics (§A), miscellaneous areas (§B), and some inference framework (§C) in the Appendix.

2 Background

This section provides an overview of LLM fundamentals, aimed at enhancing the understanding of inference serving, along with the relevant evaluation metrics.

2.1 Transformer-based LLM

The LLM is primarily constructed on the foundation of the vanilla Transformer architecture, with a particular emphasis on its decoding component. The architecture is composed of multiple layers, primarily consisting of two key components: Multi-Head Self-Attention (MHA) and Feedforward Network (FFN), complemented by the LayerNorm operation.

The input representation $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ of the model is initially processed by tokenizing the user input and incorporating positional information. Subsequently, it is transformed through three learnable weight matrices, denoted as $\mathbf{W}^Q$, $\mathbf{W}^K$, and $\mathbf{W}^V$, to obtain the corresponding query ($\mathbf{Q}$), key ($\mathbf{K}$), and value ($\mathbf{V}$) vectors which are utilized as

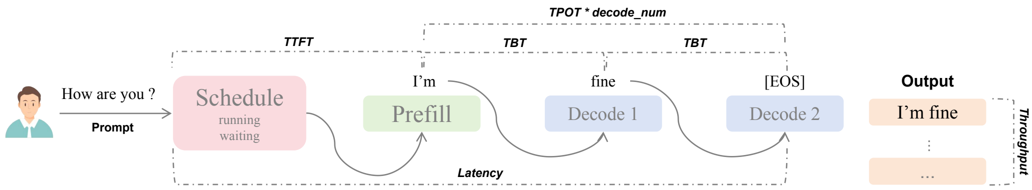

Figure 2: Illustration of some common evaluation metrics.

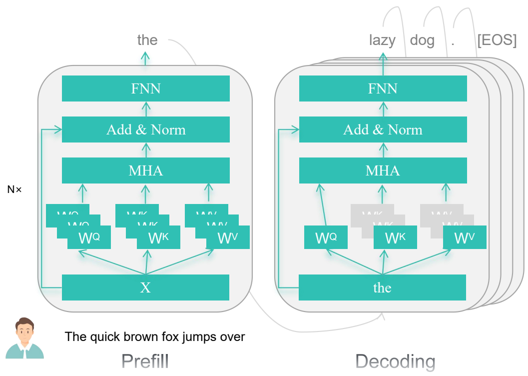

Figure 3: Illustration of the LLM Inference process.

inputs for the subsequent MHA:

$$\mathrm{MHA}(\mathbf{Q}, \mathbf{W}, \mathbf{V}) = \mathrm{Softmax}(\frac{\mathbf{QK}^T}{\sqrt{\mathbf{d}_k}})\mathbf{V} \\ \mathbf{Q} = \mathbf{XW}^Q;\ \mathbf{K} = \mathbf{XW}^K;\ \mathbf{V} = \mathbf{XW}^V \tag{1}$$

where $\mathbf{d}_k$ denotes the dimensionality of each attention head. It is evident that this constitutes the most time-consuming component, with a time complexity of $\mathcal{O}(n^2)$. The model processes $\mathbf{m}$ heads separately and concatenates the results:

$$\mathbf{O} = \mathrm{Concat}(\mathbf{H}_1, \mathbf{H}_2, \ldots, \mathbf{H}_m)\mathbf{W}^O \\ \mathbf{H}_i = \mathrm{MHA}(\mathbf{Q}_i, \mathbf{W}_i, \mathbf{V}_i) \tag{2}$$

The FFN applies two linear transformations to its input, which is first processed by LayerNorm and residual connection:

$$\mathrm{FNN}(\mathbf{x}) = \max(0, \mathbf{x}\mathbf{W}_1 + \mathbf{b}_1)\mathbf{W}_2 + \mathbf{b}_2 \tag{3}$$

2.2 Inference

LLM inference involves two phases: prefill and decoding, as illustrated in Figure 3. In prefill, the model processes the entire input in a compute-bound forward pass to produce the first token, while caching $\mathbf{K}$ and $\mathbf{V}$ (**KV cache**) to avoid recomputation. During decoding, tokens are generated sequentially using KV cache (gray blocks), which reduces the time complexity to $\mathcal{O}(n)$ at the cost of increased memory overhead. For each new token $\mathbf{X}_{new}$, the corresponding $\mathbf{Q}_{new}$, $\mathbf{K}_{new}$, and $\mathbf{V}_{new}$ are computed, ensuring efficient generation and terminate at the [EOS] token.

2.3 Evaluation

These are the conventional metrics (Agarwal et al., 2023; Zhong et al., 2024; Qin et al., 2024; Yu et al., 2022) listed in Figure 2. In addition, Goodput (Zhong et al., 2024), or "effective throughput", measures the maximum request rate that meets SLOs. Etalon (Agrawal et al., 2024a) is used to evaluate fluency to maintain smooth output during real-time interactions and its maximum output rate while preserving a certain level of fluency.

3 Fundamental Techniques

3.1 Model Placement

Due to the large number of parameters in LLMs, which exceed a single GPU's capacity, distributing them across multiple GPUs or offloading them to CPUs has become a common practice.

Model Parallelism. This part focuses on several parallelism strategies. *Pipeline parallelism* (Huang et al., 2019; Harlap et al., 2018; Narayanan et al., 2021) distributes distinct model layers across multiple devices, enabling concurrent processing of sequential data to accelerate training/inference. *Tensor parallelism* (Shoeybi et al., 2020) splits individual operations or layers into smaller sub-tensors computed in parallel across devices, enhancing computational efficiency and enabling larger model dimensions. *Sequential parallelism* (Korthikanti et al., 2022) partitions LayerNorm and Dropout activations along the sequence dimension for long-context tasks. *Context parallelism* (NVIDIA, 2024) extends this by splitting all layers along the sequence dimension. *Expert parallelism* (Fedus et al., 2022) allocates sparse MoE components across GPUs, optimizing memory usage for sparse LLMs. More details can be seen in §4.1.

Offloading. When resources are limited, balancing GPU and CPU usage is essential. Some techniques (Ren et al., 2021; Aminabadi et al., 2022; Sheng et al., 2023) store most of a model's weights in system memory or storage, loading only the necessary portions into GPU memory on demand.

PowerInfer (Song et al., 2024) computes hot neurons on the GPU and cold ones on the CPU, reducing both GPU memory usage and data transfer overhead. TwinPilots (Yu et al., 2024) integrates GPU and CPU with their hierarchical memories in an asymmetric multiprocessing framework. Park and Egger (2024) propose dynamic, fine-tuned workload allocation for efficient resource use.

3.2 Request Scheduling

For each instance, request scheduling directly impacts latency optimization. Here, we review relevant algorithms from both inter-request and intra-request scheduling perspectives.

Inter-Request Scheduling This section explores request batch prioritization under high load, focusing on execution order. Most LLM systems (Yu et al., 2022; Stojkovic et al., 2024; Qin et al., 2024) use First-Come-First-Served (FCFS), which can cause head-of-line blocking (Kaffes et al., 2019), such as a long request delaying a short one. Prioritizing shorter requests can reduce latency and better meet SLOs.

Advances in decoding length prediction (§3.3) have enabled smarter scheduling strategies. FastServe (Wu et al., 2024b) uses a Skip-Join MLFQ to prioritize urgent or long-waiting requests while preempting long ones to speed up shorter tasks. Fu et al. (2024c) approximate Shortest Job First (SJF) by using predicted decoding times. Shahout et al. (2024b) improve Shortest Remaining Time First (SRTF) with dynamic length prediction and a preemption ratio, though frequent model calls introduce overhead. Prophet (Saeresitthipitak et al., 2024) applies SJF in prefill and Skip-Join MLFQ in decoding. Kim et al. (2024) shows that cost-guided preemption lowers GPU usage. BatchLLM (Zheng et al., 2024c), instead, focuses on maximizing global sharing.

Intra-Request Scheduling This section covers scheduling within concurrent request batches to enhance parallel decoding by handling variations in arrival, completion, and output length. Orca (Yu et al., 2022) offers iteration-level scheduling for dynamic request management per iteration, improving flexibility over inter-request methods. Dynamic SplitFuse (Holmes et al., 2024) and chunked-prefills (Agrawal et al., 2024c) break the prefill stage into smaller parts merged with decoding to cut delays from long prompts. SCLS (Cheng et al., 2024b) divides generation into fixed-length slices, controlling service time and memory use precisely.

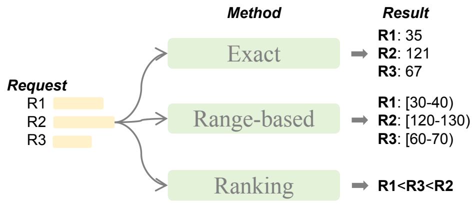

Figure 4: Overview of Length Prediction Methods.

3.3 Decoding Length Prediction

Uncertain generation length complicates scheduling, but recent work (Figure 4) classifies prediction methods into three categories.

Exact Length Prediction predicts token counts using methods like BERT embeddings with random forest regression (Cheng et al., 2024a), small OPT models (Hu et al., 2024b), or simpler regression under constraints (Qiu et al., 2024b).

Range-Based Classification classifies requests into length bins. Some approaches (Zheng et al., 2024b; Jin et al., 2023; Jain et al., 2024; Qiu et al., 2024a; Stojkovic et al., 2024) predict length ranges directly from prompts. Notably, several works (Shahout et al., 2024b) leverage classifiers on token embeddings in real-time.

Relative Ranking Prediction predicts relative relationships between requests. Fu et al. (2024c) predicts relative relationships within the same batch, enhancing robustness and reducing overfitting.

Relative Ranking Prediction simplifies batch processing by only ordering requests, but requires re-ranking for carry-over requests, unlike other methods that avoid this overhead.

Other alternative approaches include SkipPredict (Shahout and Mitzenmacher, 2024), which uses quick classification for short/long tasks before detailed prediction, and BatchLLM (Zheng et al., 2024c) that predicts lengths through prompt analysis and statistical patterns.

3.4 KV Cache Optimization

KV cache cuts inference time from quadratic to linear but poses challenges in memory management, reuse, and compression. Optimizations for specialized storage are in §5.1 and 5.2.

Memory Management. *Lossless Storage Techniques* Kwon et al. (2023) propose PagedAttention and vLLM, using OS-style paging to nearly

Dimension	Lossless	Semantic-Aware
Matching	Exact	Semantic similarity
Requests	Fixed patterns, repeats	Diverse, open-ended
Consistency	✓✓✓	✓✓
Overhead	✓	✓✓

Table 1: Comparison of KV cache reuse strategies.

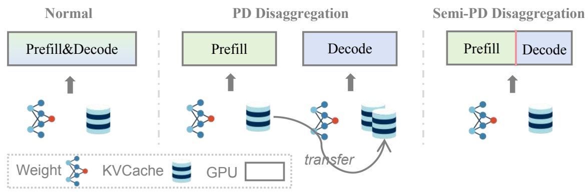

Figure 5: Comparing with normal, PD and Semi-PD Disaggregation.

eliminate memory fragmentation. FastDecode (He and Zhai, 2024) offloads cache to CPU memory through distributed processing, while LayerKV (Xiong et al., 2024) uses hierarchical allocation and offloading with layer-wise. KunServe (Cheng et al., 2024c) frees space for cache by removing model parameters, compensating via a pipeline mechanism from other instances. InstCache (Zou et al., 2024) enhances responsiveness through LLM-driven instruction prediction. PagedAttention's non-contiguous layout adds complexity and overhead, while vAttention (Prabhu et al., 2025) reduces fragmentation and preserves virtual memory contiguity via CUDA-based memory decoupling.

Approximation Methods PQCache (Zhang et al., 2024a) uses Product Quantization to compress embeddings and cut computation. InfiniGen (Lee et al., 2024) improves performance through dynamic KV cache prefetching and reduced data transfer.

Reuse Strategies. *Lossless Reuse* PagedAttention (Kwon et al., 2023) enables multi-request cache sharing through page-level management. Radix tree-based systems (Hu et al., 2024a; Srivatsa et al., 2024) implement global prefix sharing with dynamic node deletion. CachedAttention (Gao et al., 2024) minimizes redundant computation in dialogues through cross-turn cache reuse.

Semantic-aware Reuse GPTCache (Bang, 2023) uses semantic similarity to cache and reuse LLM outputs, while SCALM (Li et al., 2024c) clusters queries to uncover meaningful semantic patterns. Compared with those two matching approaches in Table 1.

Compression Techniques. To reduce inference overhead, tensor quantization and compact representations are used to balance performance and efficiency (Wang et al., 2024b).

Quantization-based Compression This method reduces memory by lowering precision. (Sheng et al., 2023) applies group-wise 4-bit KV cache quantization without I/O overhead. AWQ (Lin et al., 2024c) highlights that quantizing non-salient weights reduces quantization loss. Kivi (Zirui Liu et al., 2024) uses per-channel/token cache quantization. MiniCache (Liu et al., 2024a) exploits interlayer KV similarity for compression. Atom (Zhao et al., 2024b) adopts mixed-precision and fine-grained quantization. QServe (Lin et al., 2024e) co-designs W4A8KV4 quantization to boost GPU efficiency. OTT quantization (Su et al., 2025), which incorporates outlier token tracing, has made significant progress.

Compact Encoding Architectures It is also desirable to use smaller matrix representations instead of the previous heavy matrix. CacheGen (Liu et al., 2024c) employs a custom tensor encoder to compress KV cache into compact bitstreams, saving bandwidth with minimal decoding overhead.

3.5 PD Disaggregation

PD disaggregation separates computation-bound prefill from memory-bound decoding, enabling specialized optimization for each (Figure 5).

DistServe (Zhong et al., 2024) minimizes communication overhead by optimizing resource allocation and bandwidth-aware placement. Splitwise (Patel et al., 2024b) and HEXGEN-2 (Jiang et al., 2025b) explores homogeneous and heterogeneous device designs to optimize cost, throughput, and power. DéjàVu (Strati et al., 2024) eliminates pipeline stalls via microbatch swapping and state replication. Mooncake (Qin et al., 2024) uses a KVCache-centric disaggregated design, leveraging idle CPU, DRAM, and SSD with early rejection under load. TetriInfer (Hu et al., 2024b) applies two-level scheduling with resource prediction to avoid decoding hotspots. P/D-Serve (Jin et al., 2024b) tackles LLM deployment challenges via fine-grained prefill/decode organization, dynamic adjustments, on-demand request allocation, and efficient cache transmission.

KV Cache Transfer Problem. Semi-PD (Hong et al., 2025) separates the two stages on the same GPU by leveraging the streaming multiprocessor (SM) method, eliminating KV cache transfer over-

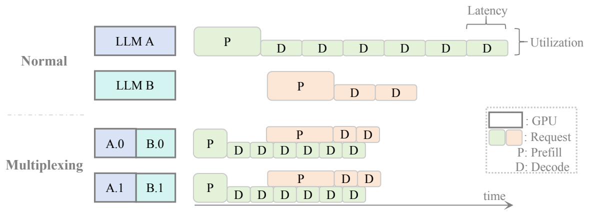

Figure 6: An example of Normal and Multiplexing.

head and enabling weight sharing. FlowKV (Li et al., 2025b) reduces KV cache migration time by minimizing communication, optimizing memory continuity, and using strategies like bidirectional alignment and merged transfers.

3.6 Multiplexing

Running a single LLM across multiple GPUs often leads to idle time due to varying model sizes and usage. To improve utilization, multiplexing multiple LLMs on a single GPU has gained attention (as shown in Figure 6).

AlpaServe (Li et al., 2023b) adopts a more rudimentary approach, resulting in suboptimal GPU utilization. To improve efficiency, MIG (NVIDIA, 2025) introduces hardware-level partitioning by dividing a single physical GPU into multiple isolated virtual instances, allowing parallel execution of multiple models. In contrast, MuxServe (Duan et al., 2024) partitions GPU Streaming Multiprocessors (SMs) via CUDA MPS and fully shares memory, requiring workload-dependent ratio tuning, they both well-suited for stable workloads with strong isolation requirements. Unlike these static methods, QLM (Patke et al., 2024) switches models on the GPU to handle different requests. Prism (Yu et al., 2025b) goes further by dynamically adjusting GPU resource allocation, enabling flexible and efficient multi-model sharing.

4 Special Architecture

4.1 MoE

MoE models, known for parameter sparsity, excel in LLMs (e.g., DeepSeek-V3 (DeepSeek-AI et al., 2024), Mixtral 8x7B (Jiang et al., 2024a)). Key inference latency challenges include expert parallelism, load balancing, and All-to-All communication (Figure 7), with Liu et al. (2024b) offering a comprehensive optimization survey.

Expert Placement. Tutel (Hwang et al., 2023) introduces switchable parallelism and dynamic pipelining without extra overhead, while

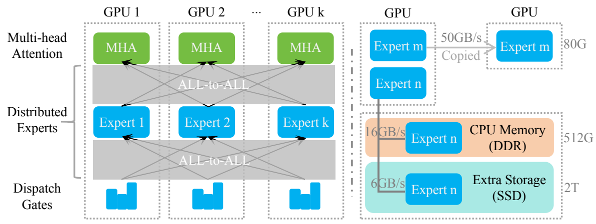

Figure 7: This figure illustrates a MoE architecture, highlighting expert placement, All-to-All communication (left), and load balancing (right). On the right, high-traffic Expert **m** and low-traffic Expert **n** are shown. For example, two strategies are presented: replicating m to a new GPU or offloading **n** to free space for **m**.

DeepSpeed-MoE (Rajbhandari et al., 2022) combines expert parallelism (He et al., 2021; Lepikhin et al., 2020) with expert-slicing. CoEL (Li et al., 2025a) leverages the sparse activation of MoE to enable resource collaboration both within and across devices. fMOE (Yu et al., 2025a) effectively guides expert prefetching, caching, and offloading decisions through its expert graph and semantically enhanced search mechanism.

Expert Load Balancing. Imbalanced token distribution causes device underutilization. Expert Buffering (Huang et al., 2023) allocates active experts to GPUs and others to CPUs, pairing high- and low-load experts using historical data. Brainstorm (Cui et al., 2023) dynamically assigns GPU units based on load, while Lynx (Gupta et al., 2024) adaptively reduces active experts. ExpertChoice (Zhou et al., 2022) selects top-k tokens per expert, rather than the reverse. High-load experts in DeepSeek-V3 (DeepSeek-AI et al., 2024) are identified using deployment statistics and periodically duplicated to optimize performance.

All-to-All Communication. Expert processing involves all-to-all exchanges for token dispatch and output gathering. Tutel (Hwang et al., 2023) uses a 2D hierarchical All-to-All algorithm, Aurora (Li et al., 2024b) optimizes token transmission order during All-to-All exchanges, and Lina (Li et al., 2023a) prioritizes All-to-All operations over concurrent All-Reduce whenever feasible, leveraging tensor partitioning to improve performance.

4.2 LoRA

LoRA (Hu et al., 2021; Chen et al., 2024; Dettmers et al., 2023) adapts LLMs to various tasks with small, trainable adapters. CaraServe (Li et al., 2024e) enables GPU-efficient, cold-start-free, SLO-aware serving via model multiplexing, CPU-GPU coordination, and rank-aware scheduling. dLoRA (Wu et al., 2024c) dynamically merges and unmerges adapters with the base model, and migrates requests and adapters across worker replicas.

4.3 Speculative Decoding

Speculative decoding (Xia et al., 2024; Wang et al., 2024a) speeds up inference by generating draft tokens with smaller LLMs and verifying them in parallel with target LLM, reducing latency and costs without quality loss. SpecInfer (Miao et al., 2024a) uses tree-based speculative inference for faster distributed and single-GPU offloading inference. AdaServe (Li et al., 2025d) and SpecServe (Huang et al., 2025) are tailored for customized SLOs, employing different fine-grained speculative decoding methods for LLMs with varying SLO requirements.

5 Specific Scenarios

5.1 Long Context

As LLMs evolve, context lengths have expanded significantly, reaching hundreds of thousands or even millions of tokens (moonshot, 2023). This growth presents both opportunities and challenges for distributed deployment, computation, and storage, especially in parallel processing, attention computation, and KV cache management.

Parallel Processing. Loongserve (Wu et al., 2024a) enhances this with elastic sequence parallelism for efficient long-context LLM serving.

Attention Computation. The attention mechanism encounters significant challenges in parallel processing and resource management. RingAttention (Liu et al., 2023) uses blockwise self-attention and FFN computation to distribute long sequences across devices, overlapping KV communication with attention. StripedAttention (Brandon et al., 2023), an extension of RingAttention, addresses imbalances from causal attention's triangular structure. DistAttention (Lin et al., 2024a) subdivides attention across GPUs, avoiding cache transfer during decoding and enabling partitioning for arbitrary sequence lengths with minimal data transfer. InstInfer (Pan et al., 2024b) offloads attention and data to Computational Storage Drives, reducing KV transfer overheads significantly.

KV Cache Management. Efficient storage for growing KV cache is crucial for generating new

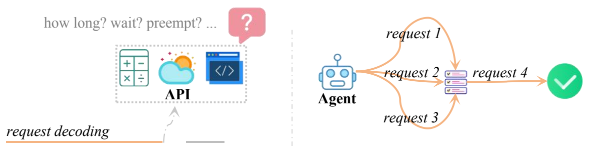

Figure 8: Overview of augmented LLMs, such as LLM with API, and LLM-based Agent.

tokens. Infinite-LLM (Lin et al., 2024a) manages dynamic LLM contexts by scheduling cache at the cluster level, balancing resources, and maximizing throughput. InfiniGen (Lee et al., 2024) optimizes cache management in CPU memory for offloading-based systems. Marconi (Pan et al., 2024a) introduces tailored admission and eviction policies for hybrid models, using experimental and theoretical analysis to show that personalized cache sizing per layer reduces memory usage significantly.

5.2 RAG

RAG enables LLMs to retrieve external knowledge for responses, but the diversity and complexity of processing pose challenges in optimizing latency and KV cache storage for large retrieval contexts.

Workflow Scheduling. Several recent innovations have focused on improving the efficiency, flexibility, and optimization of RAG workflows. PipeRAG (Jiang et al., 2024b) improves efficiency via pipeline parallelism, flexible retrieval intervals, and performance-driven quality adjustment. Teola (Tan et al., 2024) models LLM workflows as data flow nodes (e.g., Embedding, Indexing, Searching) for precise execution control. RaLMSpec (Zhang et al., 2024d) employs speculative retrieval with batched verification to reduce serving overhead. RAGServe (Ray et al., 2024) schedules queries and adjusts RAG configurations (e.g., text chunks, synthesis methods) to balance quality and latency.

Storage Optimization. Efficient storage management is critical for RAG systems, particularly in handling large-scale KV caches. Recent studies include RAGCache (Jin et al., 2024a), which employs knowledge trees and dynamic speculative pipelining to reduce redundancy. SparseRAG (Zhu et al., 2024) manages cache efficiently with prefilling and selective decoding, focusing on relevant tokens. CacheBlend (Yao et al., 2024) reuses cache and selects tokens based on a fixed percentage to recompute KV values for partial updates, enhancing efficiency and reducing latency. In contrast to CacheBlend, EPIC (Hu et al., 2024c)

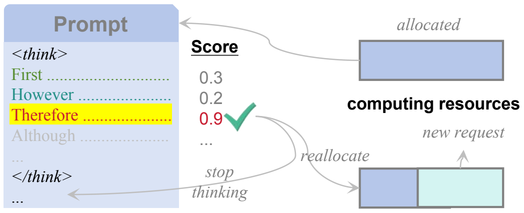

Figure 9: An example of efficient reasoning serving.

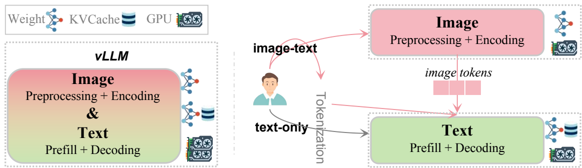

Figure 10: vLLM vs. decouple way in MLLM.

introduces position-independent context caching via static sparse computation, recomputing only a small number of tokens at the beginning of each block.

5.3 Augmented LLMs

LLMs increasingly integrate with external tools like APIs and Agents (as shown in Figure 8). APISERVE (Abhyankar et al., 2024) dynamically manages GPU resources for external APIs, while LAMPS (Shahout et al., 2024a) leverages predicting memory usage. Parrot (Lin et al., 2024b) optimizes scheduling by identifying request dependencies and commonalities, particularly in Agent scenarios, using Semantic Variables to tag each request. While Parrot is a pioneering approach to addressing agent-related challenges, it has significant limitations in practice. Tempo (Zhang et al., 2025) handles collective LLM requests by modeling execution dependencies as a DAG, with nodes as requests and edges as dependencies. Luo et al. (2025) considers the Agent as a program-level problem rather than a request-level one, which offers a valuable insight. Li et al. (2025c) uses a fluid queuing model to analyze Agent workload stability.

5.4 Test-Time Reasoning

Inference-time algorithms (OpenAI, 2024) enhance the reasoning ability (Ji et al., 2025; Qu et al., 2025) of LLMs but achieve this by generating a large number of tokens, which can strain computational resources (as shown in Figure 9). Dynasor (Fu et al., 2024b) introduces the Certaindex metric to dynamically track a model's reasoning progress, adjust computational resources based on task difficulty, and proactively terminate unpromising requests. Damani et al. (2024) optimizes resource allocation (e.g., different LLMs, computing budgets) by using a built-in reward model to assess the marginal benefit of additional computation for each request.

5.5 Multimodal LLM

As Multimodal LLM (MLLM) matures, the resulting surge in traffic presents significant deployment challenges. To address stage heterogeneity, image encoding latency, modality interference, and long-tail workload issues in multi-modal inference, ModServe (Qiu et al., 2025) introduces a decoupled architecture, as shown in Figure 10. It separates key stages such as image preprocessing, encoding, and LLM operations into independently scalable components, enabling fine-grained resource management and workload-aware optimization.

6 Future Works

Given the rapid evolution of LLM inference services, we present several recommendations for future research. *General Agent Scheduling*: Real-world demands for integrating audio, image, video, and text agents add complexity to inference serving. *Intelligent LLM Inference Service*: Utilizing the capabilities of smaller LLMs to optimize the deployment, scheduling, and storage management of larger LLMs. *Safety and Privacy*: As most services rely on cloud computing, it is essential to prevent cache leaks and ensure that any leaked data cannot be used to reconstruct user conversations. We hope that these suggestions will provide valuable insights for advancing future research.

7 Conclusion

This paper presents a comprehensive review of existing methods, covering general strategies, specialized architectures, and emerging scenarios. Specifically, the general approaches include model placement, scheduling, memory, PD disaggregation, and multiplexing optimization. We also discuss specialized architectures such as MoE, LoRA, and speculative decoding, as well as emerging scenarios like long-context problem, RAG, augmented LLM, test-time reasoning, and multimodal. We hope this work provides valuable insights for ongoing research in this critical area.

8 Limitations

This paper provides a summary and categorization of methods for LLM inference services, with the following limitations:

- While we have not conducted detailed experiments or comparative analyses of all approaches, we have presented critical insights on some key points.
- In the foundational sections, there may be some overlap with previous surveys, primarily involving a few classic papers. However, since the most recent survey (Zhou et al., 2024) only covers literature up to July 2024, many of the references we include were published after that date.
- The scope of this survey is time-sensitive, with coverage concluding at the end of April 2025.

References

Marah Abdin, Jyoti Aneja, Harkirat Behl, Sébastien Bubeck, Ronen Eldan, Suriya Gunasekar, Michael Harrison, Russell J. Hewett, Mojan Javaheripi, Piero Kauffmann, James R. Lee, Yin Tat Lee, Yuanzhi Li, Weishung Liu, Caio C. T. Mendes, Anh Nguyen, Eric Price, Gustavo de Rosa, Olli Saarikivi, Adil Salim, Shital Shah, Xin Wang, Rachel Ward, Yue Wu, Dingli Yu, Cyril Zhang, and Yi Zhang. 2024. Phi-4 technical report. *Preprint*, arXiv:2412.08905.

Reyna Abhyankar, Zijian He, Vikranth Srivatsa, Hao Zhang, and Yiying Zhang. 2024. Infercept: Efficient intercept support for augmented large language model inference. *Preprint*, arXiv:2402.01869.

Megha Agarwal, Asfandyar Qureshi, Nikhil Sardana, Linden Li, Julian Quevedo, and Daya Khudia. 2023. Llm inference performance engineering: Best practices.

Saurabh Agarwal, Anyong Mao, Aditya Akella, and Shivaram Venkataraman. 2024. Symphony: Improving memory management for llm inference workloads. *Preprint*, arXiv:2412.16434.

Amey Agrawal, Anmol Agarwal, Nitin Kedia, Jayashree Mohan, Souvik Kundu, Nipun Kwatra, Ramachandran Ramjee, and Alexey Tumanov. 2024a. Etalon: Holistic performance evaluation framework for llm inference systems. *Preprint*, arXiv:2407.07000.

Amey Agrawal, Nitin Kedia, Jayashree Mohan, Ashish Panwar, Nipun Kwatra, Bhargav Gulavani, Ramachandran Ramjee, and Alexey Tumanov. 2024b. Vidur: A large-scale simulation framework for llm inference. *Preprint*, arXiv:2405.05465.

Amey Agrawal, Nitin Kedia, Ashish Panwar, Jayashree Mohan, Nipun Kwatra, Bhargav Gulavani, Alexey Tumanov, and Ramachandran Ramjee. 2024c. Taming Throughput-Latency tradeoff in LLM inference with Sarathi-Serve. In *18th USENIX Symposium on Operating Systems Design and Implementation (OSDI 24)*, pages 117–134, Santa Clara, CA. USENIX Association.

Reza Yazdani Aminabadi, Samyam Rajbhandari, Minjia Zhang, Ammar Ahmad Awan, Cheng Li, Du Li, Elton Zheng, Jeff Rasley, Shaden Smith, Olatunji Ruwase, and Yuxiong He. 2022. Deepspeed inference: Enabling efficient inference of transformer models at unprecedented scale. *Preprint*, arXiv:2207.00032.

Abhimanyu Bambhaniya, Ritik Raj, Geonhwa Jeong, Souvik Kundu, Sudarshan Srinivasan, Midhilesh Elavazhagan, Madhu Kumar, and Tushar Krishna. 2024. Demystifying platform requirements for diverse llm inference use cases. *Preprint*, arXiv:2406.01698.

Fu Bang. 2023. GPTCache: An open-source semantic cache for LLM applications enabling faster answers and cost savings. In *Proceedings of the 3rd Workshop for Natural Language Processing Open Source Software (NLP-OSS 2023)*, pages 212–218, Singapore. Association for Computational Linguistics.

Alexander Borzunov, Max Ryabinin, Artem Chumachenko, Dmitry Baranchuk, Tim Dettmers, Younes Belkada, Pavel Samygin, and Colin Raffel. 2023. Distributed inference and fine-tuning of large language models over the internet. *Preprint*, arXiv:2312.08361.

William Brandon, Aniruddha Nrusimha, Kevin Qian, Zachary Ankner, Tian Jin, Zhiye Song, and Jonathan Ragan-Kelley. 2023. Striped attention: Faster ring attention for causal transformers. *Preprint*, arXiv:2311.09431.

Yukang Chen, Shengju Qian, Haotian Tang, Xin Lai, Zhijian Liu, Song Han, and Jiaya Jia. 2024. Longlora: Efficient fine-tuning of long-context large language models. *Preprint*, arXiv:2309.12307.

Ke Cheng, Wen Hu, Zhi Wang, Peng Du, Jianguo Li, and Sheng Zhang. 2024a. Enabling efficient batch serving for lmaas via generation length prediction. *arXiv preprint arXiv:2406.04785*.

Ke Cheng, Wen Hu, Zhi Wang, Hongen Peng, Jianguo Li, and Sheng Zhang. 2024b. Slice-level scheduling for high throughput and load balanced llm serving. *Preprint*, arXiv:2406.13511.

Rongxin Cheng, Yifan Peng, Yuxin Lai, Xingda Wei, Rong Chen, and Haibo Chen. 2024c. Kunserve: Elastic and efficient large language model serving with parameter-centric memory management. *Preprint*, arXiv:2412.18169.

Weihao Cui, Zhenhua Han, Lingji Ouyang, Yichuan Wang, Ningxin Zheng, Lingxiao Ma, Yuqing Yang, Fan Yang, Jilong Xue, Lili Qiu, Lidong Zhou, Quan Chen, Haisheng Tan, and Minyi Guo. 2023. Optimizing dynamic neural networks with brainstorm. In *17th USENIX Symposium on Operating Systems Design and Implementation (OSDI 23)*, pages 797–815, Boston, MA. USENIX Association.

Mehul Damani, Idan Shenfeld, Andi Peng, Andreea Bobu, and Jacob Andreas. 2024. Learning how hard to think: Input-adaptive allocation of lm computation. *Preprint*, arXiv:2410.04707.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Shengfeng Ye, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wanjia Zhao, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanhong Xu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *Preprint*, arXiv:2501.12948.

DeepSeek-AI, Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Haowei Zhang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Li, Hui Qu, J. L. Cai, Jian Liang, Jianzhong Guo, Jiaqi Ni, Jiashi Li, Jiawei Wang, Jin Chen, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, Junxiao Song, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Lei Xu, Leyi Xia, Liang Zhao, Litong Wang, Liyue Zhang, Meng Li, Miaojun Wang, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingming Li, Ning Tian, Panpan Huang, Peiyi Wang, Peng Zhang, Qiancheng Wang, Qihao Zhu, Qinyu Chen, Qiushi Du, R. J. Chen, R. L. Jin, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, Runxin Xu, Ruoyu Zhang, Ruyi Chen, S. S. Li, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shaoqing Wu, Shengfeng Ye, Shengfeng Ye, Shirong Ma, Shiyu Wang, Shuang Zhou, Shuiping Yu, Shunfeng Zhou, Shuting Pan, T. Wang, Tao Yun, Tian Pei, Tianyu Sun, W. L. Xiao, Wangding Zeng, Wanjia Zhao, Wei An, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, X. Q. Li, Xiangyue Jin, Xianzu Wang, Xiao Bi, Xiaodong Liu, Xiaohan Wang, Xiaojin Shen, Xiaokang Chen, Xiaokang Zhang, Xiaosha Chen, Xiaotao Nie, Xiaowen Sun, Xiaoxiang Wang, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xingkai Yu, Xinnan Song, Xinxia Shan, Xinyi Zhou, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, Y. K. Li, Y. Q. Wang, Y. X. Wei, Y. X. Zhu, Yang Zhang, Yanhong Xu, Yanhong Xu, Yanping Huang, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Li, Yaohui Wang, Yi Yu, Yi Zheng, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Ying Tang, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yu Wu, Yuan Ou, Yuchen Zhu, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yukun Zha, Yunfan Xiong, Yunxian Ma, Yuting Yan, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Z. F. Wu, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhen Huang, Zhen Zhang, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhibin Gou, Zhicheng Ma, Zhigang Yan, Zhihong Shao, Zhipeng Xu, Zhiyu Wu, Zhongyu Zhang, Zhuoshu Li, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Ziyi Gao, and Zizheng Pan. 2024. Deepseek-v3 technical report. *Preprint*, arXiv:2412.19437.

Deepspeed. 2023. Deepspeed-mii.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. Qlora: Efficient finetuning of quantized llms. *Preprint*, arXiv:2305.14314.

Jiangfei Duan, Runyu Lu, Haojie Duanmu, Xiuhong Li, Xingcheng Zhang, Dahua Lin, Ion Stoica, and Hao Zhang. 2024. Muxserve: Flexible spatial-temporal multiplexing for multiple llm serving. *Preprint*, arXiv:2404.02015.

William Fedus, Barret Zoph, and Noam Shazeer. 2022. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Preprint*, arXiv:2101.03961.

Yao Fu, Leyang Xue, Yeqi Huang, Andrei-Octavian Brabete, Dmitrii Ustiugov, Yuvraj Patel, and Luo Mai. 2024a. Serverlessllm: Low-latency serverless inference for large language models. *Preprint*, arXiv:2401.14351.

Yichao Fu, Junda Chen, Siqi Zhu, Zheyu Fu, Zhongdongming Dai, Aurick Qiao, and Hao Zhang. 2024b. Efficiently serving llm reasoning programs with certaindex. *Preprint*, arXiv:2412.20993.

Yichao Fu, Siqi Zhu, Runlong Su, Aurick Qiao, Ion Stoica, and Hao Zhang. 2024c. Efficient llm scheduling by learning to rank. *arXiv preprint arXiv:2408.15792*.

Bin Gao, Zhuomin He, Puru Sharma, Qingxuan Kang, Djordje Jevdjic, Junbo Deng, Xingkun Yang, Zhou Yu, and Pengfei Zuo. 2024. {Cost-Efficient} large language model serving for multi-turn conversations with {CachedAttention}. In *2024 USENIX Annual Technical Conference (USENIX ATC 24)*, pages 111–126.

ggml org. 2022. llamacpp.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurelien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Roziere, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, Danny Wyatt, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Francisco Guzmán, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Govind Thattai, Graeme Nail, Gregoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel Kloumann, Ishan Misra, Ivan Evtimov, Jack Zhang, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Karthik Prasad, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, Khalid El-Arini, Krithika Iyer, Kshitiz Malik, Kuenley Chiu, Kunal Bhalla, Kushal Lakhotia, Lauren Rantala-Yeary, Laurens van der Maaten, Lawrence Chen, Liang Tan, Liz Jenkins, Louis Martin, Lovish Madaan, Lubo Malo, Lukas Blecher, Lukas Landzaat, Luke de Oliveira, Madeline Muzzi, Mahesh Pasupuleti, Mannat Singh, Manohar Paluri, Marcin Kardas, Maria Tsimpoukelli, Mathew Oldham, Mathieu Rita, Maya Pavlova, Melanie Kambadur, Mike Lewis, Min Si, Mitesh Kumar Singh, Mona Hassan, Naman Goyal, Narjes Torabi, Nikolay Bashlykov, Nikolay Bogoychev, Niladri Chatterji, Ning Zhang, Olivier Duchenne, Onur Çelebi, Patrick Alrassy, Pengchuan Zhang, Pengwei Li, Petar Vasic, Peter Weng, Prajjwal Bhargava, Pratik Dubal, Praveen Krishnan, Punit Singh Koura, Puxin Xu, Qing He, Qingxiao Dong, Ragavan Srinivasan, Raj Ganapathy, Ramon Calderer, Ricardo Silveira Cabral, Robert Stojnic, Roberta Raileanu, Rohan Maheswari, Rohit Girdhar, Rohit Patel, Romain Sauvestre, Ronnie Polidoro, Roshan Sumbaly, Ross Taylor, Ruan Silva, Rui Hou, Rui Wang, Saghar Hosseini, Sahana Chennabasappa, Sanjay Singh, Sean Bell, Seohyun Sonia Kim, Sergey Edunov, Shaoliang Nie, Sharan Narang, Sharath Raparthy, Sheng Shen, Shengye Wan, Shruti Bhosale, Shun Zhang, Simon Vandenhende, Soumya Batra, Spencer Whitman, Sten Sootla, Stephane Collot, Suchin Gururangan, Sydney Borodinsky, Tamar Herman, Tara Fowler, Tarek Sheasha, Thomas Georgiou, Thomas Scialom, Tobias Speckbacher, Todor Mihaylov, Tong Xiao, Ujjwal Karn, Vedanuj Goswami, Vibhor Gupta, Vignesh Ramanathan, Viktor Kerkez, Vincent Gonguet, Virginie Do, Vish Vogeti, Vítor Albiero, Vladan Petrovic, Weiwei Chu, Wenhan Xiong, Wenyin Fu, Whitney Meers, Xavier Martinet, Xiaodong Wang, Xiaofang Wang, Xiaoqing Ellen Tan, Xide Xia, Xinfeng Xie, Xuchao Jia, Xuewei Wang, Yaelle Goldschlag, Yashesh Gaur, Yasmine Babaei, Yi Wen, Yiwen Song, Yuchen Zhang, Yue Li, Yuning Mao, Zacharie Delpierre Coudert, Zheng Yan, Zhengxing Chen, Zoe Papakipos, Aaditya Singh, Aayushi Srivastava, Abha Jain, Adam Kelsey, Adam Shajnfeld, Adithya Gangidi, Adolfo Victoria, Ahuva Goldstand, Ajay Menon, Ajay Sharma, Alex Boesenberg, Alexei Baevski, Allie Feinstein, Amanda Kallet, Amit Sangani, Amos Teo, Anam Yunus, Andrei Lupu, Andres Alvarado, Andrew Caples, Andrew Gu, Andrew Ho, Andrew Poulton, Andrew Ryan, Ankit Ramchandani, Annie Dong, Annie Franco, Anuj Goyal, Aparajita Saraf, Arkabandhu Chowdhury, Ashley Gabriel, Ashwin Bharambe, Assaf Eisenman, Azadeh Yazdan, Beau James, Ben Maurer, Benjamin Leonhardi, Bernie Huang, Beth Loyd, Beto De Paola, Bhargavi Paranjape, Bing Liu, Bo Wu, Boyu Ni, Braden Hancock, Bram Wasti, Brandon Spence, Brani Stojkovic, Brian Gamido, Britt Montalvo, Carl Parker, Carly Burton, Catalina Mejia, Ce Liu, Changhan Wang, Changkyu Kim, Chao Zhou, Chester Hu, Ching-Hsiang Chu, Chris Cai, Chris Tindal, Christoph Feichtenhofer, Cynthia Gao, Damon Civin, Dana Beaty, Daniel Kreymer, Daniel Li, David Adkins, David Xu, Davide Testuggine, Delia David, Devi Parikh, Diana Liskovich, Didem Foss, Dingkang Wang, Duc Le, Dustin Holland, Edward Dowling, Eissa Jamil, Elaine Montgomery, Eleonora Presani, Emily Hahn, Emily Wood, Eric-Tuan Le, Erik Brinkman, Esteban Arcaute, Evan Dunbar, Evan Smothers, Fei Sun, Felix Kreuk, Feng Tian, Filippos Kokkinos, Firat Ozgenel, Francesco Caggioni, Frank Kanayet, Frank Seide, Gabriela Medina Florez, Gabriella Schwarz, Gada Badeer, Georgia Swee, Gil Halpern, Grant Herman, Grigory Sizov, Guangyi, Zhang, Guna Lakshminarayanan, Hakan Inan, Hamid Shojanazeri, Han Zou, Hannah Wang, Hanwen Zha, Haroun Habeeb, Harrison Rudolph, Helen Suk, Henry Aspegren, Hunter Goldman, Hongyuan Zhan, Ibrahim Damlaj, Igor Molybog, Igor Tufanov, Ilias Leontiadis, Irina-Elena Veliche, Itai Gat, Jake Weissman, James Geboski, James Kohli, Janice Lam, Japhet Asher, Jean-Baptiste Gaya, Jeff Marcus, Jeff Tang, Jennifer Chan, Jenny Zhen, Jeremy Reizenstein, Jeremy Teboul, Jessica Zhong, Jian Jin, Jingyi Yang, Joe Cummings, Jon Carvill, Jon Shepard, Jonathan McPhie, Jonathan Torres, Josh Ginsburg, Junjie Wang, Kai Wu, Kam Hou U, Karan Saxena, Kartikay Khandelwal, Katayoun Zand, Kathy Matosich, Kaushik Veeraraghavan, Kelly Michelena, Keqian Li, Kiran Jagadeesh, Kun Huang, Kunal Chawla, Kyle Huang, Lailin Chen, Lakshya Garg, Lavender A, Leandro Silva, Lee Bell, Lei Zhang, Liangpeng Guo, Licheng Yu, Liron Moshkovich, Luca Wehrstedt, Madian Khabsa, Manav Avalani, Manish Bhatt, Martynas Mankus, Matan Hasson, Matthew Lennie, Matthias Reso, Maxim Groshev, Maxim Naumov, Maya Lathi, Meghan Keneally, Miao Liu, Michael L. Seltzer, Michal Valko, Michelle Restrepo, Mihir Patel, Mik Vyatskov, Mikayel Samvelyan, Mike Clark, Mike Macey, Mike Wang, Miquel Jubert Hermoso, Mo Metanat, Mohammad Rastegari, Munish Bansal, Nandhini Santhanam, Natascha Parks, Natasha White, Navyata Bawa, Nayan Singhal, Nick Egebo, Nicolas Usunier, Nikhil Mehta, Nikolay Pavlovich Laptev, Ning Dong, Norman Cheng, Oleg Chernoguz, Olivia Hart, Omkar Salpekar, Ozlem Kalinli, Parkin Kent, Parth Parekh, Paul Saab, Pavan Balaji, Pedro Rittner, Philip Bontrager, Pierre Roux, Piotr Dollar, Polina Zvyagina, Prashant Ratanchandani, Pritish Yuvraj, Qian Liang, Rachad Alao, Rachel Rodriguez, Rafi Ayub, Raghotham Murthy, Raghu Nayani, Rahul Mitra, Rangaprabhu Parthasarathy, Raymond Li, Rebekkah Hogan, Robin Battey, Rocky Wang, Russ Howes, Ruty Rinott, Sachin Mehta, Sachin Siby, Sai Jayesh Bondu, Samyak Datta, Sara Chugh, Sara Hunt, Sargun Dhillon, Sasha Sidorov, Satadru Pan, Saurabh Mahajan, Saurabh Verma, Seiji Yamamoto, Sharadh Ramaswamy, Shaun Lindsay, Shaun Lindsay, Sheng Feng, Shenghao Lin, Shengxin Cindy Zha, Shishir Patil, Shiva Shankar, Shuqiang Zhang, Shuqiang Zhang, Sinong Wang, Sneha Agarwal, Soji Sajuyigbe, Soumith Chintala, Stephanie Max, Stephen Chen, Steve Kehoe, Steve Satterfield, Sudarshan Govindaprasad, Sumit Gupta, Summer Deng, Sungmin Cho, Sunny Virk, Suraj Subramanian, Sy Choudhury, Sydney Goldman, Tal Remez, Tamar Glaser, Tamara Best, Thilo Koehler, Thomas Robinson, Tianhe Li, Tianjun Zhang, Tim Matthews, Timothy Chou, Tzook Shaked, Varun Vontimitta, Victoria Ajayi, Victoria Montanez, Vijai Mohan, Vinay Satish Kumar, Vishal Mangla, Vlad Ionescu, Vlad Poenaru, Vlad Tiberiu Mihailescu, Vladimir Ivanov, Wei Li, Wenchen Wang, Wenwen Jiang, Wes Bouaziz, Will Constable, Xiaocheng Tang, Xiaojian Wu, Xiaolan Wang, Xilun Wu, Xinbo Gao, Yaniv Kleinman, Yanjun Chen, Ye Hu, Ye Jia, Ye Qi, Yenda Li, Yilin Zhang, Ying Zhang, Yossi Adi, Youngjin Nam, Yu, Wang, Yu Zhao, Yuchen Hao, Yundi Qian, Yunlu Li, Yuzi He, Zach Rait, Zachary DeVito, Zef Rosnbrick, Zhaoduo Wen, Zhenyu Yang, Zhiwei Zhao, and Zhiyu Ma. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Tyler Griggs, Xiaoxuan Liu, Jiaxiang Yu, Doyoung Kim, Wei-Lin Chiang, Alvin Cheung, and Ion Stoica. 2024. Mélange: Cost efficient large language model serving by exploiting gpu heterogeneity. *Preprint*, arXiv:2404.14527.

Vima Gupta, Kartik Sinha, Ada Gavrilovska, and Anand Padmanabha Iyer. 2024. Lynx: Enabling efficient moe inference through dynamic batch-aware expert selection. *Preprint*, arXiv:2411.08982.

Zixu Hao, Huiqiang Jiang, Shiqi Jiang, Ju Ren, and Ting Cao. 2024. Hybrid slm and llm for edge-cloud collaborative inference. In *Proceedings of the Workshop on Edge and Mobile Foundation Models*, EdgeFM '24, page 36–41, New York, NY, USA. Association for Computing Machinery.

Aaron Harlap, Deepak Narayanan, Amar Phanishayee, Vivek Seshadri, Nikhil Devanur, Greg Ganger, and Phil Gibbons. 2018. Pipedream: Fast and efficient pipeline parallel dnn training. *Preprint*, arXiv:1806.03377.

Jiaao He, Jiezhong Qiu, Aohan Zeng, Zhilin Yang, Jidong Zhai, and Jie Tang. 2021. Fastmoe: A fast mixture-of-expert training system. *Preprint*, arXiv:2103.13262.

Jiaao He and Jidong Zhai. 2024. Fastdecode: High-throughput gpu-efficient llm serving using heterogeneous pipelines. *Preprint*, arXiv:2403.11421.

Ying He, Jingcheng Fang, F. Richard Yu, and Victor C. Leung. 2024. Large language models (llms) inference offloading and resource allocation in cloud-edge computing: An active inference approach. *IEEE Transactions on Mobile Computing*, 23(12):11253–11264.

Soka Hisaharo, Yuki Nishimura, and Aoi Takahashi. 2024. Optimizing llm inference clusters for enhanced performance and energy efficiency. *Authorea Preprints*.

Connor Holmes, Masahiro Tanaka, Michael Wyatt, Ammar Ahmad Awan, Jeff Rasley, Samyam Rajbhandari, Reza Yazdani Aminabadi, Heyang Qin, Arash Bakhtiari, Lev Kurilenko, and Yuxiong He. 2024. Deepspeed-fastgen: High-throughput text generation for llms via mii and deepspeed-inference. *Preprint*, arXiv:2401.08671.

Ke Hong, Lufang Chen, Zhong Wang, Xiuhong Li, Qiuli Mao, Jianping Ma, Chao Xiong, Guanyu Wu, Buhe Han, Guohao Dai, Yun Liang, and Yu Wang. 2025. semi-pd: Towards efficient llm serving via phase-wise disaggregated computation and unified storage. *Preprint*, arXiv:2504.19867.

Cunchen Hu, Heyang Huang, Junhao Hu, Jiang Xu, Xusheng Chen, Tao Xie, Chenxi Wang, Sa Wang, Yungang Bao, Ninghui Sun, and Yizhou Shan. 2024a. Memserve: Context caching for disaggregated llm serving with elastic memory pool. *Preprint*, arXiv:2406.17565.

Cunchen Hu, Heyang Huang, Liangliang Xu, Xusheng Chen, Jiang Xu, Shuang Chen, Hao Feng, Chenxi Wang, Sa Wang, Yungang Bao, Ninghui Sun, and Yizhou Shan. 2024b. Inference without interference: Disaggregate llm inference for mixed downstream workloads. *Preprint*, arXiv:2401.11181.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *Preprint*, arXiv:2106.09685.

Junhao Hu, Wenrui Huang, Haoyi Wang, Weidong Wang, Tiancheng Hu, Qin Zhang, Hao Feng, Xusheng Chen, Yizhou Shan, and Tao Xie. 2024c. Epic: Efficient position-independent context caching for serving large language models. *Preprint*, arXiv:2410.15332.

Haiyang Huang, Newsha Ardalani, Anna Sun, Liu Ke, Hsien-Hsin S. Lee, Anjali Sridhar, Shruti Bhosale, Carole-Jean Wu, and Benjamin Lee. 2023. Towards moe deployment: Mitigating inefficiencies in mixture-of-expert (moe) inference. *Preprint*, arXiv:2303.06182.

Kaiyu Huang, Hao Wu, Zhubo Shi, Han Zou, Minchen Yu, and Qingjiang Shi. 2025. Specserve: Efficient and slo-aware large language model serving with adaptive speculative decoding. *Preprint*, arXiv:2503.05096.

Yanping Huang, Youlong Cheng, Ankur Bapna, Orhan Firat, Mia Xu Chen, Dehao Chen, HyoukJoong Lee, Jiquan Ngiam, Quoc V. Le, Yonghui Wu, and Zhifeng Chen. 2019. Gpipe: Efficient training of giant neural networks using pipeline parallelism. *Preprint*, arXiv:1811.06965.

HuggingFace. 2023. Tgi (text generation inference).

Changho Hwang, Wei Cui, Yifan Xiong, Ziyue Yang, Ze Liu, Han Hu, Zilong Wang, Rafael Salas, Jithin Jose, Prabhat Ram, Joe Chau, Peng Cheng, Fan Yang, Mao Yang, and Yongqiang Xiong. 2023. Tutel: Adaptive mixture-of-experts at scale. *Preprint*, arXiv:2206.03382.

Saki Imai, Rina Nakazawa, Marcelo Amaral, Sunyanan Choochotkaew, and Tatsuhiro Chiba. 2024. Predicting llm inference latency: A roofline-driven ml method.

Kunal Jain, Anjaly Parayil, Ankur Mallick, Esha Choukse, Xiaoting Qin, Jue Zhang, Íñigo Goiri, Rujia Wang, Chetan Bansal, Victor Rühle, Anoop Kulkarni, Steve Kofsky, and Saravan Rajmohan. 2024. Intelligent router for llm workloads: Improving performance through workload-aware scheduling. *Preprint*, arXiv:2408.13510.

Suhas Jayaram Subramanya, Daiyaan Arfeen, Shouxu Lin, Aurick Qiao, Zhihao Jia, and Gregory R Ganger. 2023. Sia: Heterogeneity-aware, goodput-optimized ml-cluster scheduling. In *Proceedings of the 29th Symposium on Operating Systems Principles*, pages 642–657.

Yixin Ji, Juntao Li, Hai Ye, Kaixin Wu, Kai Yao, Jia Xu, Linjian Mo, and Min Zhang. 2025. Test-time compute: from system-1 thinking to system-2 thinking. *Preprint*, arXiv:2501.02497.

Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, Gianna Lengyel, Guillaume Bour, Guillaume Lample, Lélio Renard Lavaud, Lucile Saulnier, Marie-Anne Lachaux, Pierre Stock, Sandeep Subramanian, Sophia Yang, Szymon Antoniak, Teven Le Scao, Théophile Gervet, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2024a. Mixtral of experts. *Preprint*, arXiv:2401.04088.

Wenqi Jiang, Shuai Zhang, Boran Han, Jie Wang, Bernie Wang, and Tim Kraska. 2024b. Piperag: Fast retrieval-augmented generation via algorithm-system co-design. *Preprint*, arXiv:2403.05676.

Youhe Jiang, Fangcheng Fu, Xiaozhe Yao, Guoliang He, Xupeng Miao, Ana Klimovic, Bin Cui, Binhang Yuan, and Eiko Yoneki. 2025a. Demystifying cost-efficiency in llm serving over heterogeneous gpus. *Preprint*, arXiv:2502.00722.

Youhe Jiang, Ran Yan, Xiaozhe Yao, Yang Zhou, Beidi Chen, and Binhang Yuan. 2024c. Hexgen: Generative inference of large language model over heterogeneous environment. *Preprint*, arXiv:2311.11514.

Youhe Jiang, Ran Yan, and Binhang Yuan. 2025b. Hexgen-2: Disaggregated generative inference of LLMs in heterogeneous environment.

Chao Jin, Zili Zhang, Xuanlin Jiang, Fangyue Liu, Xin Liu, Xuanzhe Liu, and Xin Jin. 2024a. Ragcache: Efficient knowledge caching for retrieval-augmented generation. *Preprint*, arXiv:2404.12457.

Yibo Jin, Tao Wang, Huimin Lin, Mingyang Song, Peiyang Li, Yipeng Ma, Yicheng Shan, Zhengfan Yuan, Cailong Li, Yajing Sun, Tiandeng Wu, Xing Chu, Ruizhi Huan, Li Ma, Xiao You, Wenting Zhou, Yunpeng Ye, Wen Liu, Xiangkun Xu, Yongsheng Zhang, Tiantian Dong, Jiawei Zhu, Zhe Wang, Xijian Ju, Jianxun Song, Haoliang Cheng, Xiaojing Li, Jiandong Ding, Hefei Guo, and Zhengyong Zhang.

2024b. P/d-serve: Serving disaggregated large language model at scale. *Preprint*, arXiv:2408.08147.

Yunho Jin, Chun-Feng Wu, David Brooks, and Gu-Yeon Wei. 2023. s^3: Increasing gpu utilization during generative inference for higher throughput. *Advances in Neural Information Processing Systems*, 36:18015–18027.

Kostis Kaffes, Timothy Chong, Jack Tigar Humphries, Adam Belay, David Mazières, and Christos Kozyrakis. 2019. Shinjuku: Preemptive scheduling for μsecond-scale tail latency. In *16th USENIX Symposium on Networked Systems Design and Implementation (NSDI 19)*, pages 345–360, Boston, MA. USENIX Association.

Kyoungmin Kim, Kijae Hong, Caglar Gulcehre, and Anastasia Ailamaki. 2024. The effect of scheduling and preemption on the efficiency of llm inference serving. *Preprint*, arXiv:2411.07447.

Vijay Korthikanti, Jared Casper, Sangkug Lym, Lawrence McAfee, Michael Andersch, Mohammad Shoeybi, and Bryan Catanzaro. 2022. Reducing activation recomputation in large transformer models. *Preprint*, arXiv:2205.05198.

Ferdi Kossmann, Bruce Fontaine, Daya Khudia, Michael Cafarella, and Samuel Madden. 2024. Is the gpu half-empty or half-full? practical scheduling techniques for llms. *Preprint*, arXiv:2410.17840.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. *Preprint*, arXiv:2309.06180.

Wonbeom Lee, Jungi Lee, Junghwan Seo, and Jaewoong Sim. 2024. Infinigen: Efficient generative inference of large language models with dynamic kv cache management. *Preprint*, arXiv:2406.19707.

Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. 2020. Gshard: Scaling giant models with conditional computation and automatic sharding. *Preprint*, arXiv:2006.16668.

Baolin Li, Yankai Jiang, Vijay Gadepally, and Devesh Tiwari. 2024a. Llm inference serving: Survey of recent advances and opportunities. *arXiv preprint arXiv:2407.12391*.

Jialong Li, Shreyansh Tripathi, Lakshay Rastogi, Yiming Lei, Rui Pan, and Yiting Xia. 2024b. Optimizing mixture-of-experts inference time combining model deployment and communication scheduling. *Preprint*, arXiv:2410.17043.

Jiamin Li, Yimin Jiang, Yibo Zhu, Cong Wang, and Hong Xu. 2023a. Accelerating distributed MoE training and inference with lina. In *2023 USENIX Annual Technical Conference (USENIX ATC 23)*, pages 945–959, Boston, MA. USENIX Association.

Jiaxing Li, Chi Xu, Feng Wang, Isaac M von Riedemann, Cong Zhang, and Jiangchuan Liu. 2024c. Scalm: Towards semantic caching for automated chat services with large language models. *Preprint*, arXiv:2406.00025.

Luchang Li, Sheng Qian, Jie Lu, Lunxi Yuan, Rui Wang, and Qin Xie. 2024d. Transformer-lite: High-efficiency deployment of large language models on mobile phone gpus. *Preprint*, arXiv:2403.20041.

Ning Li, Song Guo, Tuo Zhang, Muqing Li, Zicong Hong, Qihua Zhou, Xin Yuan, and Haijun Zhang. 2025a. The moe-empowered edge llms deployment: Architecture, challenges, and opportunities. *Preprint*, arXiv:2502.08381.

Suyi Li, Hanfeng Lu, Tianyuan Wu, Minchen Yu, Qizhen Weng, Xusheng Chen, Yizhou Shan, Binhang Yuan, and Wei Wang. 2024e. Caraserve: Cpu-assisted and rank-aware lora serving for generative llm inference. *Preprint*, arXiv:2401.11240.

Weiqing Li, Guochao Jiang, Xiangyong Ding, Zhangcheng Tao, Chuzhan Hao, Chenfeng Xu, Yuewei Zhang, and Hao Wang. 2025b. Flowkv: A disaggregated inference framework with low-latency kv cache transfer and load-aware scheduling. *Preprint*, arXiv:2504.03775.

Yueying Li, Jim Dai, and Tianyi Peng. 2025c. Throughput-optimal scheduling algorithms for llm inference and ai agents. *Preprint*, arXiv:2504.07347.

Zhuohan Li, Lianmin Zheng, Yinmin Zhong, Vincent Liu, Ying Sheng, Xin Jin, Yanping Huang, Zhifeng Chen, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023b. Alpaserve: Statistical multiplexing with model parallelism for deep learning serving. *Preprint*, arXiv:2302.11665.

Zikun Li, Zhuofu Chen, Remi Delacourt, Gabriele Oliaro, Zeyu Wang, Qinghan Chen, Shuhuai Lin, April Yang, Zhihao Zhang, Zhuoming Chen, Sean Lai, Xupeng Miao, and Zhihao Jia. 2025d. Adaserve: Slo-customized llm serving with fine-grained speculative decoding. *Preprint*, arXiv:2501.12162.

Bin Lin, Chen Zhang, Tao Peng, Hanyu Zhao, Wencong Xiao, Minmin Sun, Anmin Liu, Zhipeng Zhang, Lanbo Li, Xiafei Qiu, Shen Li, Zhigang Ji, Tao Xie, Yong Li, and Wei Lin. 2024a. Infinite-llm: Efficient llm service for long context with distattention and distributed kvcache. *Preprint*, arXiv:2401.02669.

Chaofan Lin, Zhenhua Han, Chengruidong Zhang, Yuqing Yang, Fan Yang, Chen Chen, and Lili Qiu. 2024b. Parrot: Efficient serving of llm-based applications with semantic variable. *Preprint*, arXiv:2405.19888.

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. 2024c. Awq: Activation-aware weight quantization for llm compression and acceleration. In *MLSys*.

Xue Lin, Zhibo Zhang, Peining Yue, Haoran Li, Jin Zhang, Baoyu Fan, Huayou Su, and Xiaoli Gong. 2024d. Syncintellects: Orchestrating llm inference with progressive prediction and qos-friendly control. In *2024 IEEE/ACM 32nd International Symposium on Quality of Service (IWQoS)*, pages 1–10.

Yujun Lin, Haotian Tang, Shang Yang, Zhekai Zhang, Guangxuan Xiao, Chuang Gan, and Song Han. 2024e. Qserve: W4a8kv4 quantization and system co-design for efficient llm serving. *Preprint*, arXiv:2405.04532.

Akide Liu, Jing Liu, Zizheng Pan, Yefei He, Gholamreza Haffari, and Bohan Zhuang. 2024a. Minicache: Kv cache compression in depth dimension for large language models. *Preprint*, arXiv:2405.14366.

Hao Liu, Matei Zaharia, and Pieter Abbeel. 2023. Ring attention with blockwise transformers for near-infinite context. *Preprint*, arXiv:2310.01889.

Jiacheng Liu, Peng Tang, Wenfeng Wang, Yuhang Ren, Xiaofeng Hou, Pheng-Ann Heng, Minyi Guo, and Chao Li. 2024b. A survey on inference optimization techniques for mixture of experts models. *Preprint*, arXiv:2412.14219.

Yuhan Liu, Hanchen Li, Yihua Cheng, Siddhant Ray, Yuyang Huang, Qizheng Zhang, Kuntai Du, Jiayi Yao, Shan Lu, Ganesh Ananthanarayanan, Michael Maire, Henry Hoffmann, Ari Holtzman, and Junchen Jiang. 2024c. Cachegen: Kv cache compression and streaming for fast large language model serving. *Preprint*, arXiv:2310.07240.

Michael Luo, Xiaoxiang Shi, Colin Cai, Tianjun Zhang, Justin Wong, Yichuan Wang, Chi Wang, Yanping Huang, Zhifeng Chen, Joseph E. Gonzalez, and Ion Stoica. 2025. Autellix: An efficient serving engine for llm agents as general programs. *Preprint*, arXiv:2502.13965.

Yixuan Mei, Yonghao Zhuang, Xupeng Miao, Juncheng Yang, Zhihao Jia, and Rashmi Vinayak. 2024. Helix: Distributed serving of large language models via max-flow on heterogeneous gpus. *Preprint*, arXiv:2406.01566.

Xupeng Miao, Gabriele Oliaro, Zhihao Zhang, Xinhao Cheng, Hongyi Jin, Tianqi Chen, and Zhihao Jia. 2023. Towards efficient generative large language model serving: A survey from algorithms to systems. *arXiv preprint arXiv:2312.15234*.

Xupeng Miao, Gabriele Oliaro, Zhihao Zhang, Xinhao Cheng, Zeyu Wang, Zhengxin Zhang, Rae Ying Yee Wong, Alan Zhu, Lijie Yang, Xiaoxiang Shi, Chunan Shi, Zhuoming Chen, Daiyaan Arfeen, Reyna Abhyankar, and Zhihao Jia. 2024a. Specinfer: Accelerating large language model serving with tree-based speculative inference and verification. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3*, ASPLOS '24, page 932–949. ACM.

Xupeng Miao, Chunan Shi, Jiangfei Duan, Xiaoli Xi, Dahua Lin, Bin Cui, and Zhihao Jia. 2024b. Spotserve: Serving generative large language models on preemptible instances. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*, pages 1112–1127.

Microsoft. 2022. Deepspeed-mii.

Mistral. 2025. Mistral-small-24b-instruct-2501.

moonshot. 2023. kimi.

Deepak Narayanan, Mohammad Shoeybi, Jared Casper, Patrick LeGresley, Mostofa Patwary, Vijay Anand Korthikanti, Dmitri Vainbrand, Prethvi Kashinkunti, Julie Bernauer, Bryan Catanzaro, Amar Phanishayee, and Matei Zaharia. 2021. Efficient large-scale language model training on gpu clusters using megatron-lm. *Preprint*, arXiv:2104.04473.

Sophia Nguyen, Beihao Zhou, and YD Liu. 2024. S. towards sustainable large language model serving. In *ACM HotCarbon Workshop on Sustainable Computer Systems*.

NVIDIA. 2023. Tensorrt-llm.

NVIDIA. 2024. Context parallelism overview.

NVIDIA. 2025. Multi-instance gpu.

OpenAI. 2024. Introducing openai o1-preview.

Rui Pan, Zhuang Wang, Zhen Jia, Can Karakus, Luca Zancato, Tri Dao, Yida Wang, and Ravi Netravali. 2024a. Marconi: Prefix caching for the era of hybrid llms. *Preprint*, arXiv:2411.19379.

Xiurui Pan, Endian Li, Qiao Li, Shengwen Liang, Yizhou Shan, Ke Zhou, Yingwei Luo, Xiaolin Wang, and Jie Zhang. 2024b. Instinfer: In-storage attention offloading for cost-effective long-context llm inference. *Preprint*, arXiv:2409.04992.

Daon Park and Bernhard Egger. 2024. Improving throughput-oriented llm inference with cpu computations. In *Proceedings of the 2024 International Conference on Parallel Architectures and Compilation Techniques*, PACT '24, page 233–245, New York, NY, USA. Association for Computing Machinery.

Pratyush Patel, Esha Choukse, Chaojie Zhang, Íñigo Goiri, Brijesh Warrier, Nithish Mahalingam, and Ricardo Bianchini. 2024a. Characterizing power management opportunities for llms in the cloud. In

Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3, ASPLOS '24, page 207–222, New York, NY, USA. Association for Computing Machinery.

Pratyush Patel, Esha Choukse, Chaojie Zhang, Aashaka Shah, Íñigo Goiri, Saeed Maleki, and Ricardo Bianchini. 2024b. Splitwise: Efficient generative llm inference using phase splitting. In *2024 ACM/IEEE 51st Annual International Symposium on Computer Architecture (ISCA)*, pages 118–132. IEEE.

Archit Patke, Dhemath Reddy, Saurabh Jha, Haoran Qiu, Christian Pinto, Chandra Narayanaswami, Zbigniew Kalbarczyk, and Ravishankar Iyer. 2024. Queue management for slo-oriented large language model serving. In *Proceedings of the 2024 ACM Symposium on Cloud Computing*, SoCC '24, page 18–35, New York, NY, USA. Association for Computing Machinery.

Jie Peng, Zhang Cao, Huaizhi Qu, Zhengyu Zhang, Chang Guo, Yanyong Zhang, Zhichao Cao, and Tianlong Chen. 2024. Harnessing your dram and ssd for sustainable and accessible llm inference with mixed-precision and multi-level caching. *Preprint*, arXiv:2410.14740.

Ramya Prabhu, Ajay Nayak, Jayashree Mohan, Ramachandran Ramjee, and Ashish Panwar. 2025. vattention: Dynamic memory management for serving llms without pagedattention. In *Proceedings of the 30th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 1*, pages 1133–1150.

Ruoyu Qin, Zheming Li, Weiran He, Mingxing Zhang, Yongwei Wu, Weimin Zheng, and Xinran Xu. 2024. Mooncake: A kvcache-centric disaggregated architecture for llm serving. *Preprint*, arXiv:2407.00079.

Haoran Qiu, Anish Biswas, Zihan Zhao, Jayashree Mohan, Alind Khare, Esha Choukse, Íñigo Goiri, Zeyu Zhang, Haiying Shen, Chetan Bansal, Ramachandran Ramjee, and Rodrigo Fonseca. 2025. Modserve: Scalable and resource-efficient large multimodal model serving. *Preprint*, arXiv:2502.00937.

Haoran Qiu, Weichao Mao, Archit Patke, Shengkun Cui, Saurabh Jha, Chen Wang, Hubertus Franke, Zbigniew Kalbarczyk, Tamer Başar, and Ravishankar K Iyer. 2024a. Power-aware deep learning model serving with $\{\mu$-Serve$\}$. In *2024 USENIX Annual Technical Conference (USENIX ATC 24)*, pages 75–93.

Haoran Qiu, Weichao Mao, Archit Patke, Shengkun Cui, Saurabh Jha, Chen Wang, Hubertus Franke, Zbigniew T Kalbarczyk, Tamer Başar, and Ravishankar K Iyer. 2024b. Efficient interactive llm serving with proxy model-based sequence length prediction. *arXiv preprint arXiv:2404.08509*.

Xiaoye Qu, Yafu Li, Zhaochen Su, Weigao Sun, Jianhao Yan, Dongrui Liu, Ganqu Cui, Daizong Liu, Shuxian Liang, Junxian He, Peng Li, Wei Wei, Jing Shao, Chaochao Lu, Yue Zhang, Xian-Sheng Hua, Bowen Zhou, and Yu Cheng. 2025. A survey of efficient reasoning for large reasoning models: Language, multimodality, and beyond. *Preprint*, arXiv:2503.21614.

Bozidar Radunovic and Jean-Yves Le Boudec. 2007. A unified framework for max-min and min-max fairness with applications. *IEEE/ACM Transactions on Networking*, 15(5):1073–1083.

Samyam Rajbhandari, Conglong Li, Zhewei Yao, Minjia Zhang, Reza Yazdani Aminabadi, Ammar Ahmad Awan, Jeff Rasley, and Yuxiong He. 2022. Deepspeed-moe: Advancing mixture-of-experts inference and training to power next-generation ai scale. *Preprint*, arXiv:2201.05596.

Deevashwer Rathee, Dacheng Li, Ion Stoica, Hao Zhang, and Raluca Popa. 2024. Mpc-minimized secure llm inference. *Preprint*, arXiv:2408.03561.

Siddhant Ray, Rui Pan, Zhuohan Gu, Kuntai Du, Ganesh Ananthanarayanan, Ravi Netravali, and Junchen Jiang. 2024. Ragserve: Fast quality-aware rag systems with configuration adaptation. *Preprint*, arXiv:2412.10543.

Jie Ren, Samyam Rajbhandari, Reza Yazdani Aminabadi, Olatunji Ruwase, Shuangyan Yang, Minjia Zhang, Dong Li, and Yuxiong He. 2021. Zero-offload: Democratizing billion-scale model training. *Preprint*, arXiv:2101.06840.

Schwinn Saereesitthipitak, Ashish Rao, Cathy Zhou, and William Li. 2024. Prophet: An llm inference engine optimized for head-of-line blocking.

Rana Shahout, Cong Liang, Shiji Xin, Qianru Lao, Yong Cui, Minlan Yu, and Michael Mitzenmacher. 2024a. Fast inference for augmented large language models. *Preprint*, arXiv:2410.18248.

Rana Shahout, Eran Malach, Chunwei Liu, Weifan Jiang, Minlan Yu, and Michael Mitzenmacher. 2024b. Don't stop me now: Embedding based scheduling for llms. *Preprint*, arXiv:2410.01035.

Rana Shahout and Michael Mitzenmacher. 2024. Skippredict: When to invest in predictions for scheduling. *Preprint*, arXiv:2402.03564.

Ying Sheng, Shiyi Cao, Dacheng Li, Banghua Zhu, Zhuohan Li, Danyang Zhuo, Joseph E. Gonzalez, and Ion Stoica. 2024. Fairness in serving large language models. *Preprint*, arXiv:2401.00588.

Ying Sheng, Lianmin Zheng, Binhang Yuan, Zhuohan Li, Max Ryabinin, Daniel Y. Fu, Zhiqiang Xie, Beidi Chen, Clark Barrett, Joseph E. Gonzalez, Percy Liang, Christopher Ré, Ion Stoica, and Ce Zhang. 2023. Flexgen: High-throughput generative inference of large language models with a single gpu. *Preprint*, arXiv:2303.06865.

Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. 2020. Megatron-lm: Training multi-billion parameter language models using model parallelism. *Preprint*, arXiv:1909.08053.

Yixin Song, Zeyu Mi, Haotong Xie, and Haibo Chen. 2024. Powerinfer: Fast large language model serving with a consumer-grade gpu. *Preprint*, arXiv:2312.12456.

Vikranth Srivatsa, Zijian He, Reyna Abhyankar, Dongming Li, and Yiying Zhang. 2024. Preble: Efficient distributed prompt scheduling for llm serving. *Preprint*, arXiv:2407.00023.

Jovan Stojkovic, Chaojie Zhang, Íñigo Goiri, Josep Torrellas, and Esha Choukse. 2024. Dynamollm: Designing llm inference clusters for performance and energy efficiency. *arXiv preprint arXiv:2408.00741*.

Foteini Strati, Sara Mcallister, Amar Phanishayee, Jakub Tarnawski, and Ana Klimovic. 2024. Déjàvu: Kv-cache streaming for fast, fault-tolerant generative llm serving. *Preprint*, arXiv:2403.01876.

Yi Su, Yuechi Zhou, Quantong Qiu, Juntao Li, Qingrong Xia, Ping Li, Xinyu Duan, Zhefeng Wang, and Min Zhang. 2025. Accurate kv cache quantization with outlier tokens tracing. *Preprint*, arXiv:2505.10938.

Biao Sun, Ziming Huang, Hanyu Zhao, Wencong Xiao, Xinyi Zhang, Yong Li, and Wei Lin. 2024. Llumnix: Dynamic scheduling for large language model serving. *Preprint*, arXiv:2406.03243.

Xin Tan, Yimin Jiang, Yitao Yang, and Hong Xu. 2024. Teola: Towards end-to-end optimization of llm-based applications. *Preprint*, arXiv:2407.00326.

Jikai Wang, Yi Su, Juntao Li, Qingrong Xia, Zi Ye, Xinyu Duan, Zhefeng Wang, and Min Zhang. 2024a. Opt-tree: Speculative decoding with adaptive draft tree structure. *Preprint*, arXiv:2406.17276.

Wenxiao Wang, Wei Chen, Yicong Luo, Yongliu Long, Zhengkai Lin, Liye Zhang, Binbin Lin, Deng Cai, and Xiaofei He. 2024b. Model compression and efficient inference for large language models: A survey. *Preprint*, arXiv:2402.09748.

Bingyang Wu, Shengyu Liu, Yinmin Zhong, Peng Sun, Xuanzhe Liu, and Xin Jin. 2024a. Loongserve: Efficiently serving long-context large language models with elastic sequence parallelism. *Preprint*, arXiv:2404.09526.

Bingyang Wu, Yinmin Zhong, Zili Zhang, Shengyu Liu, Fangyue Liu, Yuanhang Sun, Gang Huang, Xuanzhe Liu, and Xin Jin. 2024b. Fast distributed inference serving for large language models. *Preprint*, arXiv:2305.05920.

Bingyang Wu, Ruidong Zhu, Zili Zhang, Peng Sun, Xuanzhe Liu, and Xin Jin. 2024c. dLoRA: Dynamically orchestrating requests and adapters for LoRA LLM serving. In *18th USENIX Symposium on Operating Systems Design and Implementation (OSDI 24)*, pages 911–927, Santa Clara, CA. USENIX Association.

Hui Wu, Yi Gan, Feng Yuan, Jing Ma, Wei Zhu, Yutao Xu, Hong Zhu, Yuhua Zhu, Xiaoli Liu, Jinghui Gu, and Peng Zhao. 2024d. Efficient llm inference solution on intel gpu. *Preprint*, arXiv:2401.05391.

Heming Xia, Zhe Yang, Qingxiu Dong, Peiyi Wang, Yongqi Li, Tao Ge, Tianyu Liu, Wenjie Li, and Zhifang Sui. 2024. Unlocking efficiency in large language model inference: A comprehensive survey of speculative decoding. *Preprint*, arXiv:2401.07851.

Yi Xiong, Hao Wu, Changxu Shao, Ziqing Wang, Rui Zhang, Yuhong Guo, Junping Zhao, Ke Zhang, and Zhenxuan Pan. 2024. Layerkv: Optimizing large language model serving with layer-wise kv cache management. *Preprint*, arXiv:2410.00428.

Daliang Xu, Hao Zhang, Liming Yang, Ruiqi Liu, Gang Huang, Mengwei Xu, and Xuanzhe Liu. 2024. Fast on-device llm inference with npus. *Preprint*, arXiv:2407.05858.

Huan Yang, Deyu Zhang, Yudong Zhao, Yuanchun Li, and Yunxin Liu. 2024a. A first look at efficient and secure on-device llm inference against kv leakage. *Preprint*, arXiv:2409.04040.

Zheming Yang, Yuanhao Yang, Chang Zhao, Qi Guo, Wenkai He, and Wen Ji. 2024b. Perllm: Personalized inference scheduling with edge-cloud collaboration for diverse llm services. *Preprint*, arXiv:2405.14636.

Jiayi Yao, Hanchen Li, Yuhan Liu, Siddhant Ray, Yihua Cheng, Qizheng Zhang, Kuntai Du, Shan Lu, and Junchen Jiang. 2024. Cacheblend: Fast large language model serving for rag with cached knowledge fusion. *Preprint*, arXiv:2405.16444.

Wangsong Yin, Mengwei Xu, Yuanchun Li, and Xuanzhe Liu. 2024. Llm as a system service on mobile devices. *Preprint*, arXiv:2403.11805.

Chengye Yu, Tianyu Wang, Zili Shao, Linjie Zhu, Xu Zhou, and Song Jiang. 2024. Twinpilots: A new computing paradigm for gpu-cpu parallel llm inference. In *Proceedings of the 17th ACM International Systems and Storage Conference*, SYSTOR '24, page 91–103, New York, NY, USA. Association for Computing Machinery.

Gyeong-In Yu, Joo Seong Jeong, Geon-Woo Kim, Soojeong Kim, and Byung-Gon Chun. 2022. Orca: A distributed serving system for Transformer-Based generative models. In *16th USENIX Symposium on Operating Systems Design and Implementation (OSDI 22)*, pages 521–538, Carlsbad, CA. USENIX Association.

Hanfei Yu, Xingqi Cui, Hong Zhang, Hao Wang, and Hao Wang. 2025a. fmoe: Fine-grained expert offloading for large mixture-of-experts serving. *Preprint*, arXiv:2502.05370.

Shan Yu, Jiarong Xing, Yifan Qiao, Mingyuan Ma, Yangmin Li, Yang Wang, Shuo Yang, Zhiqiang Xie, Shiyi Cao, Ke Bao, Ion Stoica, Harry Xu, and Ying Sheng. 2025b. Prism: Unleashing gpu sharing for cost-efficient multi-llm serving. *Preprint*, arXiv:2505.04021.

Zhihang Yuan, Yuzhang Shang, Yang Zhou, Zhen Dong, Zhe Zhou, Chenhao Xue, Bingzhe Wu, Zhikai Li, Qingyi Gu, Yong Jae Lee, et al. 2024. Llm inference unveiled: Survey and roofline model insights. *arXiv preprint arXiv:2402.16363*.

Hailin Zhang, Xiaodong Ji, Yilin Chen, Fangcheng Fu, Xupeng Miao, Xiaonan Nie, Weipeng Chen, and Bin Cui. 2024a. Pqcache: Product quantization-based kvcache for long context llm inference. *Preprint*, arXiv:2407.12820.

Mingjin Zhang, Jiannong Cao, Xiaoming Shen, and Zeyang Cui. 2024b. Edgeshard: Efficient llm inference via collaborative edge computing. *Preprint*, arXiv:2405.14371.

Wei Zhang, Zhiyu Wu, Yi Mu, Banruo Liu, Myungjin Lee, and Fan Lai. 2025. Tempo: Application-aware llm serving with mixed slo requirements. *Preprint*, arXiv:2504.20068.

Xiaojin Zhang, Yulin Fei, Yan Kang, Wei Chen, Lixin Fan, Hai Jin, and Qiang Yang. 2024c. No free lunch theorem for privacy-preserving llm inference. *Preprint*, arXiv:2405.20681.

Zhihao Zhang, Alan Zhu, Lijie Yang, Yihua Xu, Lanting Li, Phitchaya Mangpo Phothilimthana, and Zhihao Jia. 2024d. Accelerating retrieval-augmented language model serving with speculation. *Preprint*, arXiv:2401.14021.

Juntao Zhao, Borui Wan, Yanghua Peng, Haibin Lin, and Chuan Wu. 2024a. Llm-pq: Serving llm on heterogeneous clusters with phase-aware partition and adaptive quantization. *Preprint*, arXiv:2403.01136.

Yilong Zhao, Chien-Yu Lin, Kan Zhu, Zihao Ye, Lequn Chen, Size Zheng, Luis Ceze, Arvind Krishnamurthy, Tianqi Chen, and Baris Kasikci. 2024b. Atom: Low-bit quantization for efficient and accurate llm serving. *Preprint*, arXiv:2310.19102.

Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Sun, Jeff Huang, Cody Hao Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E. Gonzalez, Clark Barrett, and Ying Sheng. 2024a. Sglang: Efficient execution of structured language model programs. *Preprint*, arXiv:2312.07104.

Zangwei Zheng, Xiaozhe Ren, Fuzhao Xue, Yang Luo, Xin Jiang, and Yang You. 2024b. Response length perception and sequence scheduling: An llm-empowered llm inference pipeline. *Advances in Neural Information Processing Systems*, 36.

Zhen Zheng, Xin Ji, Taosong Fang, Fanghao Zhou, Chuanjie Liu, and Gang Peng. 2024c. Batchllm: Optimizing large batched llm inference with global prefix sharing and throughput-oriented token batching. *Preprint*, arXiv:2412.03594.

Yinmin Zhong, Shengyu Liu, Junda Chen, Jianbo Hu, Yibo Zhu, Xuanzhe Liu, Xin Jin, and Hao Zhang. 2024. Distserve: Disaggregating prefill and decoding for goodput-optimized large language model serving. *arXiv preprint arXiv:2401.09670*.

Yanqi Zhou, Tao Lei, Hanxiao Liu, Nan Du, Yanping Huang, Vincent Zhao, Andrew Dai, Zhifeng Chen, Quoc Le, and James Laudon. 2022. Mixture-of-experts with expert choice routing. *Preprint*, arXiv:2202.09368.

Zixuan Zhou, Xuefei Ning, Ke Hong, Tianyu Fu, Jiaming Xu, Shiyao Li, Yuming Lou, Luning Wang, Zhihang Yuan, Xiuhong Li, et al. 2024. A survey on efficient inference for large language models. *arXiv preprint arXiv:2404.14294*.

Yun Zhu, Jia-Chen Gu, Caitlin Sikora, Ho Ko, Yinxiao Liu, Chu-Cheng Lin, Lei Shu, Liangchen Luo, Lei Meng, Bang Liu, and Jindong Chen. 2024. Accelerating inference of retrieval-augmented generation via sparse context selection. *Preprint*, arXiv:2405.16178.

Zirui Liu, Jiayi Yuan, Hongye Jin, Shaochen Zhong, Zhaozhuo Xu, Vladimir Braverman, Beidi Chen, and Xia Hu. 2024. Kivi : Plug-and-play 2bit kv cache quantization with streaming asymmetric quantization.

Longwei Zou, Tingfeng Liu, Kai Chen, Jiangang Kong, and Yangdong Deng. 2024. Instcache: A predictive cache for llm serving. *Preprint*, arXiv:2411.13820.

Małgorzata Łazuka, Andreea Anghel, and Thomas Parnell. 2024. Llm-pilot: Characterize and optimize performance of your llm inference services. *Preprint*, arXiv:2410.02425.

A LLM Inference Serving in Cluster

This section focuses on cluster-level deployment and scheduling, as well as cloud-based cluster serving, as detailed in Figure 11.

A.1 Cluster Optimization

Internal optimizations for homogeneous devices require more machines as parameter scale increases, while heterogeneous machines are preferred for their flexibility, efficiency, and cost-effectiveness (Mei et al., 2024). External optimizations, like service-oriented cluster scheduling, further enhance internal optimizations.

Architecture and Optimization for Heterogeneous Resources. Jayaram Subramanya et al. (2023) propose a joint optimization framework for adaptive task allocation across GPU types and batch sizes, demonstrating significant throughput improvements over static configurations. Helix (Mei et al., 2024) models the execution of LLM services on heterogeneous GPUs and networks as a maximum flow problem in a directed weighted graph, where nodes represent GPU instances and edges encode GPU and network heterogeneity through capacity constraints. LLM-PQ (Zhao et al., 2024a) advocates an adaptive quantization and phase-aware partition scheme tailored for heterogeneous GPU clusters. HexGen (Jiang et al., 2024c) supports asymmetric parallel execution on GPUs with different computing capabilities. Splitwise (Patel et al., 2024b), DistServe (Zhong et al., 2024) and HEXGEN-2 (Jiang et al., 2025b) optimize computation on heterogeneous disaggregated architectures, with the latter focusing on LLM serving via constraint-based scheduling and graph-based resource optimization. Hisaharo et al. (2024) integrate advanced interconnect technology, high-bandwidth memory, and energy-efficient power management.

Service-Aware Scheduling. DynamoLLM (Stojkovic et al., 2024) optimizes service clusters by adjusting instances, parallelization, and GPU frequencies based on input/output lengths. Splitwise (Patel et al., 2024b) proposes cluster-level scheduling across prefill and decoding on separate devices.

A.2 Load Balancing

Cluster-level load balancing optimizes request distribution to prevent node overload or underutilization, improving throughput and service quality. While most frameworks (Yu et al., 2022; Kwon et al., 2023) rely on traditional methods like Round Robin and Random (Deepspeed, 2023), recent advances in heuristic, dynamic, and predictive scheduling provide more sophisticated solutions.

Heuristic Algorithm. SCLS (Cheng et al., 2024b) employs a max-min algorithm (Radunovic and Le Boudec, 2007) to balance the workloads. It assigns the batch with the longest estimated serving time to the instance with the lowest score, where the score represents the total serve time of all batches in the instance's queue. SAL (Kossmann et al., 2024) quantifies the load on two key factors: (1) the number of queued prefill tokens and (2) the available memory. This ensures that requests are dispatched to the server with the lowest load, addressing scenarios where delays occur due to either a full token batch or insufficient memory.

Dynamic Scheduling. Llumnix (Sun et al., 2024) dynamically reschedules requests across model instances during runtime to handle request heterogeneity and unpredictability. It uses real-time migration to transfer requests and memory states, enabling mid-operation migration to the least loaded instance based on real-time load growth.

Intelligent Predictive Scheduling. Jain et al. (2024) propose a reinforcement learning-based router that models request routing as a Markov Decision Process, aiming to derive an optimal policy for maximizing discounted rewards. It integrates response length prediction, workload impact estimation, and reinforcement learning.

A.3 Cloud-Based LLM Serving

If local LLM deployment lacks resources, cloud services offer a more economical alternative, with recent research focusing on optimizing cloud deployment and edge collaboration for efficiency.

Deployment and Computing Effective. To reduce LLM deployment costs, spot instances are used despite preemption risks. SpotServe (Miao et al., 2024b) mitigates this with dynamic reparallelization, parameter reuse, and stateful inference recovery. ServerlessLLM (Fu et al., 2024a) tackles serverless cold start latency via optimized checkpoints, live migration, and locality-aware scheduling. Mélange (Griggs et al., 2024) optimizes GPU allocation based on request patterns, lowering costs. POLCA (Patel et al., 2024a) boosts efficiency through power management, while Imai

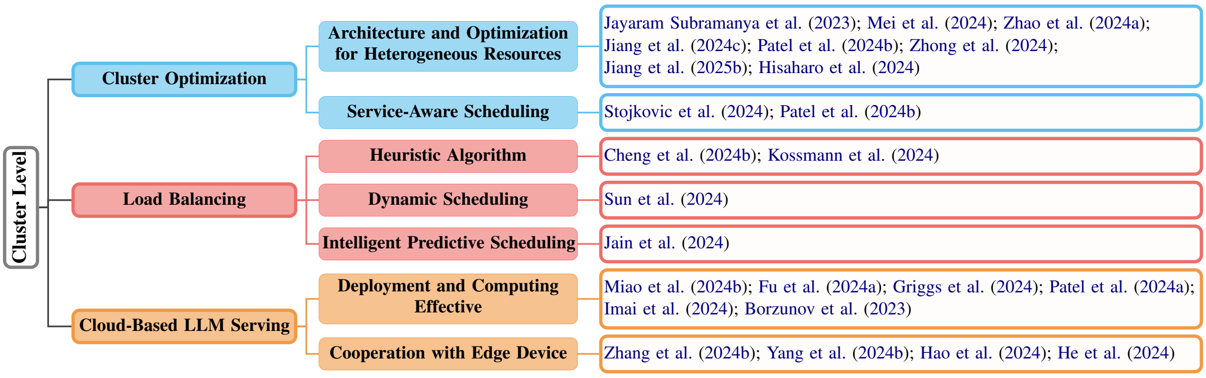

Figure 11: Taxonomy of Cluster-Level strategies for LLM inference serving.

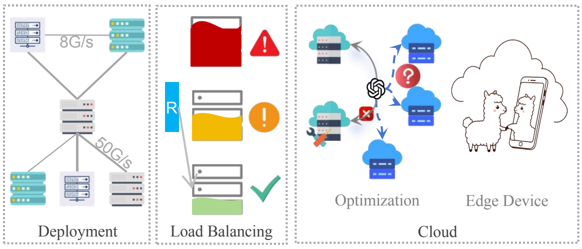

Figure 12: Overview of the deployment, load balancing, and cloud-based in cluster-level optimization. **R** represents a request.

et al. (2024) predict inference latency to enhance cluster management. Borzunov et al. (2023) propose a way to integrate idle resources through geodistributed devices connected via the internet. Jiang et al. (2025a) found that optimizing GPU combinations, deployment setups, and workload distribution boosts LLM cost efficiency on heterogeneous cloud platforms.

Cooperation with Edge Device. To meet SLOs amid cloud latency and bandwidth limits, edge computing offers solutions. EdgeShard (Zhang et al., 2024b) leverages collaboration between distributed edge devices and cloud servers. PreLLM (Yang et al., 2024b) uses a multi-armed bandit framework for personalized scheduling. Hao et al. (2024) integrate small edge models with cloud LLM to address memory constraints, while He et al. (2024) apply deep reinforcement learning for efficient, latency-aware inference offloading.

B Miscellaneous Areas

Other niche but important directions (Figure 13) are also advancing LLM inference toward a more comprehensive and far-reaching future.

B.1 Hardware

Recent advancements in optimizing LLM inference have focused on improving efficiency, speed, and resource utilization in various hardware techniques.

Peng et al. (2024) propose a mixed-precision, multi-level caching system (HBM, DRAM, SSDs) and a model modularization algorithm to enable LLM inference on resource-constrained, outdated hardware. Wu et al. (2024d) explore inference service solutions on Intel GPUs. LLM-Pilot (Łazuka et al., 2024) benchmarks LLM inference across GPUs and recommends the most cost-effective GPU for unseen LLMs. GenZ (Bambhaniya et al., 2024) is an analytical tool for studying the relationship between LLM inference performance and various hardware platform design parameters. Li et al. (2024d) present Transformer-Lite, an innovative inference engine optimized for mobile GPUs, designed to enhance the efficiency and inference speed of LLM deployment on mobile devices. LLMS (Yin et al., 2024) is an innovative system on mobile devices that, under stringent memory constraints, implements fine-grained, chunk-based KV cache compression and a globally optimized swapping mechanism to decouple applications from LLM memory management, thereby minimizing the overhead of context switching. Xu et al. (2024) utilize on-device Neural Processing Unit (NPU) offloading to enhance NPU offloading efficiency and reduce prefill latency.

B.2 Privacy

Protecting user conversation content in LLMs from potential leakage is an important issue. Yang et al. (2024a) adopt weight permutation to shuffle KV pairs, preventing attackers from reconstructing the entire context. Zhang et al. (2024c) quantify the trade-off between privacy protection and utility

Figure 13: Taxonomy of Miscellaneous Areas for LLM inference serving.

loss, pointing out that privacy protection mechanisms (such as randomization) can reduce privacy leakage but will introduce utility loss. MARILL (Rathee et al., 2024) achieves substantial reductions in the costly operations required for secure inference within multi-party computation by optimizing the architecture of LLMs during the fine-tuning phase.

B.3 Simulator

Considering the diversity of computing devices and their associated high costs, a comprehensive simulator is indispensable for conducting trials in virtual environments. Agrawal et al. (2024b) introduce Vidur, a scalable, high-fidelity simulation framework for evaluating LLM performance under various deployment configurations, alongside Vidur-Search, a tool for optimizing deployments to meet performance constraints and reduce costs. The Helix system (Mei et al., 2024), featuring an event-based simulator, enables accurate simulation of LLM inference in heterogeneous GPU clusters by adjusting factors like network conditions, machine heterogeneity, and cluster scale, providing rapid and cost-effective deployment evaluations.

B.4 Fairness

In LLM inference services, request frequency limits are typically imposed on each client (e.g., user or application) to ensure fair resource allocation. These limits prevent excessive requests from monopolizing resources and degrading service quality for others. However, they may also result in underutilized resources. Sheng et al. (2024) propose a novel fairness definition, based on a cost function considering input and output tokens. Additionally, a new scheduling algorithm, the Virtual Token Counter (VTC), introduces fair scheduling through a continuous batching mechanism.

B.5 Energy

Given the substantial power demands of LLM computations, optimizing energy usage is a critical challenge that must be addressed. Nguyen et al. (2024) investigate the carbon emissions of LLMs from operational and embodied perspectives, aiming to promote sustainable LLM services. Researchers analyzed the performance and energy consumption of the LLaMA model across varying parameter scales and batch sizes, incorporating the carbon intensity of different power grid regions. This study provides insights into the environmental impact of LLMs and explores opportunities to optimize sustainable LLM systems.

C Common Inference Framework

We present a comprehensive comparison of vLLM (Kwon et al., 2023), TensorRT-LLM (NVIDIA, 2023), SGLang (Zheng et al., 2024a), TGI (HuggingFace, 2023), DeepSpeed-MII (Microsoft, 2022), and llama.cpp (ggml org, 2022) in terms of hardware support, core optimization focus, and typical use cases (as shown in Table 3).

Metric	Definition	Key Focus
TTFT	Latency from input to first token.	Critical for real-time apps (e.g., chatbots).
TBT	Time interval between consecutive tokens.	Reflects step-by-step responsiveness.
TPOT	Average time per token during decoding.	Measures token generation efficiency.
Throughput	Tokens generated per second across all requests.	Evaluates system capacity under high load.
Capacity	Maximum throughput while meeting SLOs.	Represents system's upper performance limit.
Normalized Latency	Total execution time divided by token count.	Holistic view of system efficiency.
Percentile Metrics	Latency distribution (e.g., P50, P90, P99).	Evaluates stability and performance bounds.

Table 2: LLM Inference Service Evaluation Metrics.

Tool	Hardware Support	Core Optimization Focus	Typical Use Case
vLLM	GPU	PagedAttention, optimized KV cache, high concurrency	Multi-turn chat, OpenChat, FastChat, scalable SaaS
TensorRT-LLM	NVIDIA GPU	Low latency, high throughput	Enterprise-grade GPU server deployment
SGLang	GPU / CPU	Structured prompt execution optimization	Agent-based reasoning, complex logic inference
TGI	GPU	Continuous batching, cloud integration	HuggingFace model-as-a-service (SaaS)
DeepSpeed-MII	GPU	Memory compression, distributed inference	High-throughput API serving
llama.cpp	CPU / Low-end GPU	Quantization, lightweight deployment	Local development / edge computing

Table 3: Comparison of LLM Inference Frameworks.

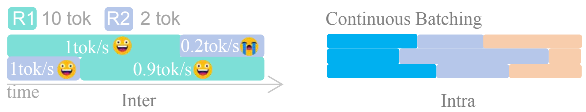

Figure 14: An example of inter- and intra-scheduling. $\mathbf{R}$ represents a request. In Inter-request scheduling, two requests, $\mathbf{R1}$ (10 toks) and $\mathbf{R2}$ (2 toks), arrive simultaneously. Ignoring the prefill process, if $\mathbf{R1}$ is processed first, its generation rate is 1 tok/s, and $\mathbf{R2}$'s rate is 0.2 tok/s. Reversing the order gives $\mathbf{R2}$ a rate of 1 tok/s and $\mathbf{R1}$ 0.9 tok/s. The default decoding speed is 1 token/s.