Staircase Streaming for Low-Latency Multi-Agent Inference

Anonymous ACL submission

Abstract

Recent advances in large language models (LLMs) opened up new directions for leveraging the collective expertise of multiple LLMs. These methods, such as Mixture-of-Agents, typically employ additional inference steps to generate intermediate outputs, which are then used to produce the final response. While multi-agent inference can enhance response quality, it can significantly increases the time to first token (TTFT), posing a challenge for latency-sensitive applications and hurts user experience. To address this issue, we propose *staircase streaming* for low-latency multi-agent inference. Instead of waiting for the complete intermediate outputs from previous steps, we begin generating the final response as soon as we receive partial outputs from these steps. Experimental results demonstrate that staircase streaming reduces TTFT by up to 93% while maintaining response quality.

1 Introduction

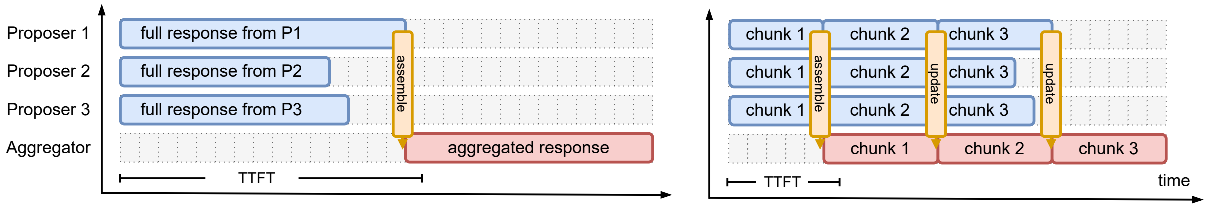
Large language models (LLMs) (Zhang et al., 2022; Chowdhery et al., 2022; Touvron et al., 2023; Team et al., 2023; Brown et al., 2020; OpenAI, 2023) have significantly advanced the field of natural language processing, showing remarkable capabilities across a wide range of tasks and domains. With the proliferation of LLMs and their impressive achievements, a new line of research has emerged: *multi-agent inference* (Du et al., 2023; Liang et al., 2023; Chen et al., 2023b; Wang et al., 2024b). It aims to harness the collective strengths of multiple LLMs to produce more reliable, consistent, and high-quality responses, matching or even surpassing state-of-the-art performance.

While multi-agent inference techniques can significantly enhance response quality by combining the outputs of multiple LLMs, they often incorporate additional inference steps. For example, Multi-Agent Debate (MAD) (Liang et al., 2023) uses several models to generate answers independently and then debate their answers; Mixture-of-Agents (MoA) (Wang et al., 2024b) employs proposers to generate initial responses and an aggregator to synthesize a higher-quality response from them. They both require to wait for all additional generations to finish before producing the final output, leading to an increased time to first token (TTFT). This latency is particularly problematic for real-time applications such as chatbots, where low latency is crucial for user experience. This naturally raises the question: *How can we leverage the strengths of multi-agent inference while minimizing such latency?*

To address this question, we propose *staircase streaming*, a novel approach for low-latency multi-agent inference. Instead of waiting for complete results from all the agents in the previous step, our approach begins streaming tokens once the first chunk of tokens becomes available from the preceding step. By leveraging partial results and overlapping the generation processes, staircase streaming effectively reduces latency while maintaining the benefits of multi-agent inference.

The intuition behind staircase streaming is based on two key observations: (1) *Effective partial prompting*: Jiang et al. 2023c demonstrate that even partial tokens within a prompt can effectively instruct LLM. Similarly, we found partial generation results can provide valuable, diverse viewpoints for other LLMs. (2) *Diminishing returns in subsequent outputs*: The most crucial information in a model's response is often front-loaded. The first portion of generation typically contains the main ideas, while subsequent parts generally add details, examples, or refinements to these core concepts.

Figure 1 illustrates an example of staircase streaming applied to MoA. In the normal case (a), the multi-agent sytem is bottlenecked by the proposer with the longest generation time, resulting in significant latency. With staircase streaming (b),



(a) Normal Streaming (b) Staircase Streaming

Figure 1: Comparison of normal streaming and staircase streaming using an MoA with 3 proposers and 1 aggregator. (a) In normal streaming, each LLM generates a full response before proceeding to the next step, leading to a longer TTFT. (b) Staircase streaming reduces TTFT by initiating the next step once the first chunks of proposed responses are available, enabling parallel processing between the proposers and the aggregator.

the aggregator begins generating output as soon as a chunk of tokens is available from each proposer. Once the aggregator generates a chunk of tokens, it waits for the next chunk from the proposers, updates its prompt with these newly received tokens, and continues generating.

The evaluation results on the Arena-Hard and AlpacaEval benchmarks (Li et al., 2024; Dubois et al., 2024) demonstrate that our staircase streaming can be effectively adapted to multi-agent inference methods, reducing latency up to 93% while maintaining the response quality.

2 Methodology

In this section, we first briefly overview token streaming. We then introduce staircase streaming, our approach that minimizes latency while leveraging multi-agent inference strengths. Additionally, we present a prefix caching optimized variant to further enhance efficiency.

2.1 Token Streaming

Token streaming allows autoregressive LMs to generate and return text incrementally, providing users with an early indication of content quality and enhancing the user experience. TTFT, measuring the latency between initiating a request and generating the first token, is a crucial metric for it.

However, multi-agent inference approaches often struggle with token streaming due to dependencies between multiple models. The first token becomes available after all preceding steps complete and the last step's prefill finishes. Depending on the generation length, this process can incur substantial latency, sometimes spanning dozens of seconds.

Algorithm 1 Staircase Streaming for MoA

Input: User prompt Q; Proposers $P_1, \ldots, P_N$; Aggregator A; Chunk sizes $C_{\text{P},j}$ and $C_{\text{A},j}$ ($j = 1, 2, \ldots$)
(1) THREAD PROPOSER ($i = 1, \ldots, N$):
  Initialize $j \leftarrow 0$
  while not end of response
    $j \leftarrow j + 1$
    $R_{i,j} \leftarrow P_i(Q + \cup_{k=1}^{j-1} R_{i,k})$ ▷ Generate $C_{\text{P},j}$ tokens
    Send chunk $R_{i,j}$ to aggregator A
  end while
(2) THREAD AGGREGATOR:
  Initialize $j \leftarrow 0$
  while not end of response
    $j \leftarrow j + 1$
    Receive chunks $R_{1,j}, \ldots, R_{N,j}$ from all proposers or total number of proposers minus redundancy if redundancy is not 0.
    Update prompt $Q_{\text{A},j}$ for the aggregator with proposed responses so far $[\cup_{k=1}^{j} R_{1,j}, \ldots, \cup_{k=1}^{j} R_{N,j}]$ with template in Table 2.
    $S_j \leftarrow A(Q_{\text{A},j} + \cup_{k=1}^{j-1} S_k)$ ▷ Generate $C_{\text{A},j}$ tokens
    (Streaming S_j to the user)
  end while
Output: $\cup_{k=1}^{j} S_k$

2.2 Staircase Streaming

Staircase streaming is proposed to enable low-latency multi-agent inference by streaming tokens between models as soon as partial results are available, rather than waiting for complete outputs. The key idea is to break the strict sequential dependencies between models, allowing a pipelined execution pattern where models process partial results immediately.

MoA is a typical multi-agent inference system where multiple proposer models generate initial responses based on a user prompt. These responses are then synthesized by an aggregator model to produce the final output. Algorithm 1 illustrates an adaptation of staircase streaming for MoA. The proposers and aggregator run in separate threads. Each proposer generates a chunk of tokens based on the user prompt and previously generated chunks, and then immediately sends the new chunk to the aggregator. The aggregator, after receiving chunks from proposers, updates its prompt by concatenating the new chunks. And it then generates the next chunk of the final output based on the updated prompt, which is streamed to the user. Meanwhile, the proposers process the next chunk of text in parallel. For multi-layer MoA systems, the same staircase streaming approach can be applied to each intermediate LLM, creating a nested staircase streaming pipeline.

The chunk sizes $C_{\mathrm{P},j}$ and $C_{\mathrm{A},j}$ control the granularity of the streaming process and can be adjusted to balance latency and computational efficiency. Smaller chunk sizes lead to more frequent updates but may require more prefill operations, while larger chunk sizes can increase TTFT. To achieve a better trade-off, we use smaller chunk sizes at the beginning and gradually increase them as the generation continues.

Following the similar principle, staircase streaming can be applied to general multi-agent systems by having each agent generate and stream chunks of tokens in parallel. These chunks are passed to the next agent or aggregator, which updates its prompt and continues the process.

Reducing TTFT In staircase streaming, TTFT is reduced since the aggregator starts generation with the first chunk $R_{i,1}$ from each proposer rather than waiting for the complete responses R_i. TTFT of MoA and staircase version can be represented as:

$$\mathrm{TTFT}_{\mathrm{normal}} = \max_{1\leq i\leq N}\left(\sum_{j=1}^{\mathrm{eos}} T_{R_{i,j}}\right) + T_{\mathrm{prefill}} \quad (1)$$

$$\mathrm{TTFT}_{\mathrm{staircase}} = \max_{1\leq i\leq N}\left(T_{R_{i,1}}\right) + T_{\mathrm{prefill}} \quad (2)$$

where $T_{R_{i,j}}$ is the time taken by proposer i to generate the j-th chunk and T_{prefill} is the time taken by the aggregator to process the initial chunks.

2.3 Prefix-Caching Optimized Staircase Streaming

Prefix-caching enhances the efficiency of transformer models by reusing previously computed key-value pairs for identical prompt prefixes (Zheng et al., 2023). In staircase streaming, we can leverage this technique by appending new token chunks to the end of the prompt, keeping the prefix unchanged. This approach, whose prompt template is presented in Table 3, allows seamless updating of the staircase streaming prompt without requiring a full prefill computation. While this method may fragment responses from the same proposer, it can save more compute.

	ArenaHard Win Rate	AlpacaEval LC Win	TTFT second	TPS tokens/s
Gemma-2-9B-IT	40.6	48.5	$0.06_{\pm 0.01}$	$69.4_{\pm 0.4}$
MAD	50.8	55.8	$6.70_{\pm 2.3}$	$35.9_{\pm 4.4}$
+ staircase	45.9	55.2	$0.45_{\pm 0.0}$	$44.7_{\pm 6.6}$
MoA	47.5	56.8	$10.6_{\pm 3.4}$	$28.3_{\pm 8.9}$
+ staircase	48.3	55.1	$0.47_{\pm 0.1}$	$43.4_{\pm 6.5}$
+ staircase & prefix-cache	46.9	53.0	$0.45_{\pm 0.1}$	$45.0_{\pm 7.0}$

Table 1: Results of different inference methods.

3 Evaluation

3.1 Setup

Benchmarks We evaluate on Arena-Hard (Li et al., 2024) and AlpacaEval 2.0 (Dubois et al., 2024). For efficiency metrics, we assess the time to first token (TTFT) and the end-to-end tokens per second (TPS)[1] on a subset of Arena-Hard and AlpacaEval with 64 samples.

Methods We incorporate staircase streaming to Mixture-of-Agents (MoA) and Multi-Agent Debate (MAD). We use Gemma-2-9B-IT (Team et al., 2024), LLaMA-3.1-8B-Instruct (Dubey et al., 2024), Mistral-7B-Instruct-v0.3 (Jiang et al., 2023a), and Qwen-1.5-7B-Chat (Bai et al., 2023); and use Gemma-2-9B-IT as the aggregator. We set the chunk size to 8 for the first chunk, increasing to 128 for the second chunk, and capping at 256 for proposers and 128 for the aggregator for later chunks. Benchmarks were conducted using the Together Inference API.[2] To minimize the effect of queuing times associated with the serverless API, we use redundancy of 2 – we skip the slowest 2 models – for the first chunk. To reduce the variance caused by network latency and server load, we run the aggregator model with vLLM on 1 H100 GPU for TTFT and TPS metrics. Detailed setup can be found in Appendix A.

3.2 Results

Table 1 presents the evaluation results. Our approach significantly reduces TTFT by up to 93% and increases TPS by up to 1.6x, enhancing the responsiveness while preserving the response quality of multi-agent inference. Staircase streaming is effective across MoA and MAD, showing the ver-

[1]TPS = The number tokens generated by the aggregator / the time elapsed between sending the request and receiving the last token.

[2]https://api.together.ai/playground/chat

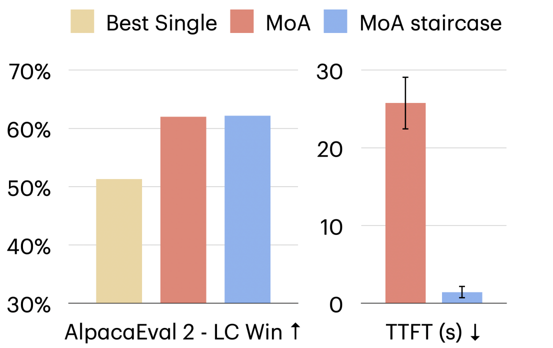


Figure 2: Results on Larger LLMs. The 'Best Single' model is WizardLM 8x22B. For MoA, the proposers include Qwen1.5-72B-Chat, Qwen1.5-110B-Chat, Wizard 8x22B, Mixtral-8x22B-Instruct-v0.1, and Llama-3-70B-Instruct. The aggregator is Qwen1.5-110B-Chat. TTFT was evaluated using the Together serverless endpoint, so the results may vary depending on server load.

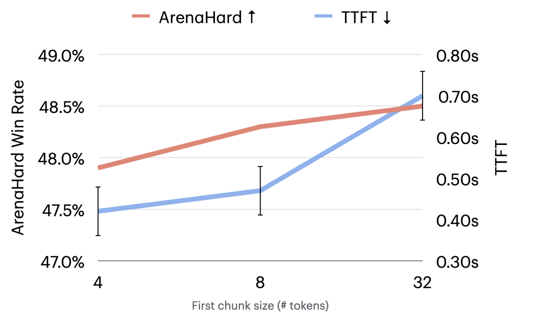


Figure 3: The impact of first chunk size on ArenaHard win rate and TTFT.

satility. While the prefix-cache optimized variant slightly reduces the win rate, it further improves the efficiency,

Scaling up Model Size Figure 2 shows AlpacaEval results for larger models. The 'Best Single' model achieved a 51% LC win rate. In contrast, MoA with 6 proposers achieved 62%, showing the benefits of leveraging the expertise from multiple LLMs. The staircase version achieved a similar LC win rate while maintaining reasonable TTFT. This shows staircase streaming effectively scales to larger models, achieving good performance with manageable latency.

Effect of Chunk Sizes The first chunk size is a crucial hyperparameter for optimizing TTFT, as indicated in eq. (2). Figure 3 presents the impact of varying the first chunk size on response quality and TTFT. As the chunk size increases from 4 to 32 tokens, the ArenaHard win rate improves, while TTFT initially decreases but then significantly increases for larger chunk sizes. This presents the trade-off between response quality and latency in staircase streaming, with a chunk size of 8 tokens as the sweet point.

4 Related Work

4.1 Multi-Agent Inference

To mitigate the computational costs associated with multi-LLM inference, previous studies have explored training a *router* that predicts the best-performing model from a fixed set of LLMs for a given input (Wang et al., 2024a; Shnitzer et al., 2024; Lu et al., 2023; Chen et al., 2023c). Another line of work focuses on collaborative inference between multiple LLMs (Chan et al., 2023; Du et al., 2023; Liang et al., 2023; Chen et al., 2023b; Wang et al., 2024b), where models work together to generate more reliable, consistent, and high-quality responses.

4.2 Efficient Inference

Recent LLM inference optimizations enhance efficiency and resource use. PagedAttention (Kwon et al., 2023) is proposed to manage non-contiguous memory blocks efficiently. To further address the memory consumption of key-value caches, recent research proposes using quantization and sparsification techniques (Hooper et al., 2024; Zhang et al., 2023; Xiao et al., 2023). Speculative decoding (Leviathan et al., 2022; Chen et al., 2023a) speeds up inference by using extra small models to generate multiple token guesses and validate them in parallel. To the best of our knowledge, there has been limited work for the inference efficiency of multi-agent systems.

5 Conclusion

In this paper, we introduced staircase streaming, a novel approach to reduce TTFT in multi-agent systems by streaming tokens incrementally between models. The key innovation is to break the strict sequential dependencies inherent in existing multi-agent systems, enabling a pipelined execution pattern that leverages partial results to minimize idle waiting time. Our experimental results demonstrate that this approach not only maintains response quality but also significantly reduces TTFT by up to 93%.

6 Limitation

Our method requires modification to the original multi-agent algorithm. While these changes are generally minimal, they may impact performance. So necessary evaluation is needed before adapting staircase streaming to other multi-agent methods to ensure the quality does not change significantly.

The improved TTFT and streaming experience comes at a cost of increased prompt token consumption. To mitigate this, we proposed a prefix-caching optimized version that can save a significant number of the tokens if the API services or the local inference engine supports it. Currently, more and more inference engines (Zheng et al., 2023; Kwon et al., 2023) and API services roll out the support prefix-caching.

7 Ethical Considerations

Although multi-agent inference generally improves the harmlessness of model outputs, as shown in the Mixture-of-Agent paper (Wang et al., 2024b), there are still potential risks that the final answer may retain biases or harmful content from the intermediate outputs. This underscores the importance of diligent oversight and ethical guidelines in the development and deployment of such systems.

References

Reza Yazdani Aminabadi, Samyam Rajbhandari, Ammar Ahmad Awan, Cheng Li, Du Li, Elton Zheng, Olatunji Ruwase, Shaden Smith, Minjia Zhang, Jeff Rasley, and Yuxiong He. 2022. Deepspeed-inference: enabling efficient inference of transformer models at unprecedented scale. In *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, SC '22. IEEE Press.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, Binyuan Hui, Luo Ji, Mei Li, Junyang Lin, Runji Lin, Dayiheng Liu, Gao Liu, Chengqiang Lu, Keming Lu, Jianxin Ma, Rui Men, Xingzhang Ren, Xuancheng Ren, Chuanqi Tan, Sinan Tan, Jianhong Tu, Peng Wang, Shijie Wang, Wei Wang, Shengguang Wu, Benfeng Xu, Jin Xu, An Yang, Hao Yang, Jian Yang, Shusheng Yang, Yang Yao, Bowen Yu, Hongyi Yuan, Zheng Yuan, Jianwei Zhang, Xingxuan Zhang, Yichang Zhang, Zhenru Zhang, Chang Zhou, Jingren Zhou, Xiaohuan Zhou, and Tianhang Zhu. 2023. Qwen technical report. *arXiv preprint arXiv: 2309.16609*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Chi-Min Chan, Weize Chen, Yusheng Su, Jianxuan Yu, Wei Xue, Shanghang Zhang, Jie Fu, and Zhiyuan Liu. 2023. Chateval: Towards better llm-based evaluators through multi-agent debate. *arXiv preprint arXiv:2308.07201*.

Charlie Chen, Sebastian Borgeaud, Geoffrey Irving, Jean-Baptiste Lespiau, L. Sifre, and John M. Jumper. 2023a. Accelerating large language model decoding with speculative sampling. *ArXiv*, abs/2302.01318.

Justin Chih-Yao Chen, Swarnadeep Saha, and Mohit Bansal. 2023b. Reconcile: Round-table conference improves reasoning via consensus among diverse llms. *arXiv preprint arXiv:2309.13007*.

Lingjiao Chen, Matei Zaharia, and James Zou. 2023c. Frugalgpt: How to use large language models while reducing cost and improving performance. *arXiv preprint arXiv:2305.05176*.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. 2022. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*.

Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher R'e. 2022. Flashattention: Fast and memory-efficient exact attention with io-awareness. *ArXiv*, abs/2205.14135.

Tri Dao, Daniel Haziza, Francisco Massa, and Grigory Sizov. 2023. Flash-decoding for long-context inference.

Yilun Du, Shuang Li, Antonio Torralba, Joshua B Tenenbaum, and Igor Mordatch. 2023. Improving factuality and reasoning in language models through multiagent debate. *arXiv preprint arXiv:2305.14325*.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Yann Dubois, Balázs Galambosi, Percy Liang, and Tatsunori B Hashimoto. 2024. Length-controlled alpacaeval: A simple way to debias automatic evaluators. *arXiv preprint arXiv:2404.04475*.

Coleman Hooper, Sehoon Kim, Hiva Mohammadzadeh, Michael W. Mahoney, Yakun Sophia Shao, Kurt Keutzer, and Amir Gholami. 2024. Kvquant: Towards 10 million context length llm inference with kv cache quantization. *ArXiv*, abs/2401.18079.

Yichong Huang, Xiaocheng Feng, Baohang Li, Yang Xiang, Hui Wang, Bing Qin, and Ting Liu. 2024. Enabling ensemble learning for heterogeneous large language models with deep parallel collaboration. *arXiv preprint arXiv:2404.12715*.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023a. Mistral 7b. *Preprint*, arXiv:2310.06825.

Dongfu Jiang, Xiang Ren, and Bill Yuchen Lin. 2023b. LLM-blender: Ensembling large language models with pairwise ranking and generative fusion. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14165–14178, Toronto, Canada. Association for Computational Linguistics.

Huiqiang Jiang, Qianhui Wu, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. 2023c. Llmlingua: Compressing prompts for accelerated inference of large language models. *arXiv preprint arXiv:2310.05736*.

Sehoon Kim, Karttikeya Mangalam, Suhong Moon, Jitendra Malik, Michael W. Mahoney, Amir Gholami, and Kurt Keutzer. 2023. Speculative decoding with big little decoder. In *Neural Information Processing Systems*.

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. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. 2022. Fast inference from transformers via speculative decoding. In *International Conference on Machine Learning*.

Tianle Li, Wei-Lin Chiang, Evan Frick, Lisa Dunlap, Tianhao Wu, Banghua Zhu, Joseph E Gonzalez, and Ion Stoica. 2024. From crowdsourced data to high-quality benchmarks: Arena-hard and benchbuilder pipeline. *arXiv preprint arXiv:2406.11939*.

Tian Liang, Zhiwei He, Wenxiang Jiao, Xing Wang, Yan Wang, Rui Wang, Yujiu Yang, Zhaopeng Tu, and Shuming Shi. 2023. Encouraging divergent thinking in large language models through multi-agent debate. *arXiv preprint arXiv:2305.19118*.

Keming Lu, Hongyi Yuan, Runji Lin, Junyang Lin, Zheng Yuan, Chang Zhou, and Jingren Zhou. 2023. Routing to the expert: Efficient reward-guided ensemble of large language models. *Preprint*, arXiv:2311.08692.

OpenAI. 2023. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. 2017. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*.

Ying Sheng, Lianmin Zheng, Binhang Yuan, Zhuohan Li, Max Ryabinin, Beidi Chen, Percy Liang, Christopher Ré, Ion Stoica, and Ce Zhang. 2023. Flexgen: high-throughput generative inference of large language models with a single gpu. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org.

Tal Shnitzer, Anthony Ou, Mírian Silva, Kate Soule, Yuekai Sun, Justin Solomon, Neil Thompson, and Mikhail Yurochkin. 2024. Large language model routing with benchmark datasets.

Benjamin Spector and Christal Re. 2023. Accelerating llm inference with staged speculative decoding. *ArXiv*, abs/2308.04623.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Yonghui Wu, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, et al. 2023. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*.

Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, et al. 2024. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Hongyi Wang, Felipe Maia Polo, Yuekai Sun, Souvik Kundu, Eric Xing, and Mikhail Yurochkin. 2024a. Fusing models with complementary expertise. In *The Twelfth International Conference on Learning Representations*.

Junlin Wang, Jue Wang, Ben Athiwaratkun, Ce Zhang, and James Zou. 2024b. Mixture-of-agents enhances large language model capabilities. *arXiv preprint arXiv:2406.04692*.

Qineng Wang, Zihao Wang, Ying Su, Hanghang Tong, and Yangqiu Song. 2024c. Rethinking the bounds of llm reasoning: Are multi-agent discussions the key? *arXiv preprint arXiv:2402.18272*.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. 2023. Efficient streaming language models with attention sinks. *ArXiv*, abs/2309.17453.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. 2022. Opt: Open pre-trained transformer language models. *arXiv e-prints*, pages arXiv–2205.

Zhenyu (Allen) Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark W. Barrett, Zhangyang Wang, and Beidi Chen. 2023. H2o: Heavy-hitter oracle for efficient generative inference of large language models. *ArXiv*, abs/2306.14048.

Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Jeff Huang, Chuyue Sun, Cody Hao Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E Gonzalez, et al. 2023. Efficiently programming large language models using sglang. *arXiv preprint arXiv:2312.07104*.

A Detailed Setups

In the code implementation, there are five hyperparameters that can control the behaviors of staircase streaming: *first_chunk_size*, *second_chunk_size*, *chunk_size*, *aggregator_chunk_size*, and *redundancy*. When running staircase streaming, $C_{\mathrm{P},1}$ and $C_{\mathrm{A},1}$ would be set to *first_chunk_size*. $C_{\mathrm{P},2}$ and $C_{\mathrm{A},2}$ are set to *second_chunk_size*. Then for the rest of the chunks, $C_{\mathrm{P},j}$ and $C_{\mathrm{A},j}$ are set to *chunk_size* and *aggregator_chunk_size* respectively for $j > 2$.

Redundancy is introduced to control how many models' responses must be ready before the aggregator can start generating. For example, if there are five proposers and *Redundancy*=2, then that means for the first chunk, it only needs responses from three models before the aggregator can generate the first chunk. This granular setup allows practitioners to have maximum control over the trade-offs between TTFT and performance.

For our main results in Table 1, we use *first_chunk_size*=8, *second_chunk_size*=128, *chunk_size*=256, *aggregator_chunk_size*=128, and *redundancy*=2. These choices is carefully chosen from the ablation study, and generalize across different setups.

B Prompt Templates

We present the prompt templates used in our evaluation. For MoA experiments, we mainly followed the original paper by (Wang et al., 2024b). The staircase streaming templates are shown in Table 2 and Table 3. For MAD experiments, we adapted the templates to suit open-ended chat scenarios, as MAD is originally designed for tasks with short and deterministic answers, e.g. classification. The staircase streaming templates is shown in Table 4.

C Related Work

C.1 Multi-Agent Inference

A straightforward solution to leverage multiple LLMs is ranking outputs from different models. Jiang et al. (2023b) introduced PAIRRANKER, which performs pairwise comparisons on candidate outputs to select the best one. To alleivate the substantial computational costs of multi-LLM inference, other studies have explored training a *router* that predicts the best-performing model from a fixed set of LLMs for a given input (Wang et al., 2024a; Shnitzer et al., 2024; Lu et al., 2023). Additionally, FrugalGPT (Chen et al., 2023c) proposed reducing the cost of using LLMs by employing different models in a cascading manner. To better leverage the responses of multiple models, Jiang et al. (2023b) trained GENFUSER, a model designed to generate an improved response by capitalizing on the strengths of multiple candidates. Huang et al. (2024) proposed fusing the outputs of different models by averaging their output probability distributions.

Another line of work focuses on collaborative collaboration, where multiple LLMs act as agents that collectively discuss and reason through given problems interactively. Du et al. (2023); Liang et al. (2023); Chan et al. (2023) established a mechanism for discussions among LLM-based agents. ReConcile (Chen et al., 2023b) adopt multi-agent discussion with weighted voting. Wang et al. (2024c) systematically compared collaborative approaches and found that a single agent with a strong prompt, including detailed demonstrations, can achieve comparable response quality to collaborative approaches. Shazeer et al. (2017) adopted a layered structure, using different LLMs as proposers to propose answers and another LLM to aggregate the final answer.

C.2 Efficient Inference

Recent advancements in LLM inference optimization focus on enhancing system efficiency and resource utilization in order to reduce calculation. vLLM (Kwon et al., 2023) addresses fragmentation with PagedAttention, which manages non-contiguous memory blocks efficiently. DeepSpeed-Inference (Aminabadi et al., 2022) combines GPU, CPU, and NVMe memory for high-throughput

Table 2: Prompt template of staircase streaming for MoA.

Role	Content
system	You have been provided with a set of responses from various open-source models to the latest user query. Your task is to synthesize these responses into a single, high-quality response. It is crucial to critically evaluate the information provided in these responses, recognizing that some of it may be biased or incorrect. Your response should not simply replicate the given answers but should offer a refined, accurate, and comprehensive reply to the instruction. Ensure your response is well-structured, coherent, and adheres to the highest standards of accuracy and reliability. Responses from models: 1. $\{\cup_{k=1}^{j} R_{1,k}\}$ 2. $\{\cup_{k=1}^{j} R_{2,k}\}$ ... N. $\{\cup_{k=1}^{j} R_{N,k}\}$
user	{Q}

inference of diverse transformer models. FlexGen (Sheng et al., 2023) increases the throughput with optimized GPU memory usage by integrating memory and computation from GPUs, CPUs, and disks, and quantizes weights to boost inference speed. Flash-Decoding (Dao et al., 2023) based on FlashAttention (Dao et al., 2022) accelerates long-context inference with parallel processing of KV pairs with length considered.

On the algorithm side, KV-Cache optimization is a commonly studied topic. During inference with LLM, it is necessary to store the KV pairs of previously generated tokens in a cache for future token generation. As the length of generated tokens increases, this KV cache expands significantly, resulting in high memory consumption and longer inference times (Hooper et al., 2024; Zhang et al., 2023; Xiao et al., 2023). Another inference speedup technique is speculative decoding (Leviathan et al., 2022; Chen et al., 2023a). It involves smaller models to have multiple educated token guesses in parallel, then validating and pruning these candidates based on their likelihood and consistency with the output, thus reducing computational load and improving performance. Following this idea, Staged Speculative (Spector and Re, 2023) organizes speculative outputs into a tree structure, enhancing batch generation and overall performance. BiLD (Kim et al., 2023) proposes a fallback policy allowing the small model to defer to the target model when uncertain, and a rollback policy for correcting small model errors.

Table 3: Prompt template of prefix-caching optimized staircase streaming for MoA.

Role	Content
system	You have been provided with a set of responses from various open-source models to the latest user query in chunks. Your task is to synthesize these response chunks into a single, high-quality response. It is crucial to critically evaluate the information provided in these responses, recognizing that some of it may be biased or incorrect. As some responses may be incomplete yet, craft your synthesized response to allow for easy updating or expansion as new information becomes available. Your response should not simply replicate the given answers but should offer a refined, accurate, and comprehensive reply to the instruction. Ensure your response is well-structured, coherent, and adheres to the highest standards of accuracy and reliability. Responses from models: Chunk 1: Model 1: $\{R_{1,1}\}$ ... Model N: $\{R_{N,1}\}$ Chunk 2: Model 1: $\{R_{1,2}\}$ ... Model N: $\{R_{N,2}\}$ ...
user	{Q}

Table 4: Prompt template of staircase streaming for MAD. MAD puts the model's own output before the debate prompt, here we assueme it's the 'N'-th model.

Role	Content
user	{Q}
assistant	$\{\{\cup_{k=1}^{j} R_{N,k}\}\}$
user	These are the responses to the query from other agents: One agent solution: $\{\cup_{k=1}^{j} R_{1,k}\}$ One agent solution: $\{\cup_{k=1}^{j} R_{2,k}\}$ ... One agent solution: $\{\cup_{k=1}^{j} R_{N-1,k}\}$ Using the responses from other agents as additional information, can you provide your response to the query? The original query is {Q}