ACCELERATION MULTIPLE HEADS DECODING FOR LLM VIA DYNAMIC TREE ATTENTION

Zhendong Zhang
Vivo Mobile Communication Co. Ltd
zhd.zhang.ai@gmail.com

ABSTRACT

Multiple heads decoding accelerates the inference of Large Language Models (LLMs) by predicting next several tokens simultaneously. It generates and verifies multiple candidate sequences in parallel via tree attention with a fixed structure. In this paper, we replace the fixed tree attention with dynamic tree attention on multiple head decoding, specifically in the context of MEDUSA. We propose a simple and low complexity strategy to generate candidates and construct the dynamic tree structure. Preliminary experiments show that the proposed method improves the decoding efficiency of multiple head decoding for LLMs while maintaining the generation quality. This result demonstrates the potential for improvement of multiple head decoding in candidate generation.

1 INTRODUCTION

The scale of Large Language Models (LLMs) has been growing rapidly in recent years (Radford et al., 2019; Brown et al., 2020; Achiam et al., 2023). However, this growth leads to an increase in inference latency. From a system perspective, the main latency bottleneck of LLM inference is memory bandwidth rather than arithmetic computations (Shazeer, 2019). This bottleneck is inherent to the sequential nature of auto-regressive decoding, which generates only a single token at a time, underutilizes the arithmetic computation potential of modern accelerators (Cai et al., 2024).

Researchers have explored generating multiple tokens simultaneously, which follows a guess-verify approach. Depending on the methods used for initial token guessing and subsequent verification, recent techniques can be classified into three categories: speculative decoding, Jacobi decoding, and multiple head decoding. Speculative decoding uses a smaller draft model to generate a token sequence, which is subsequently verified by the original model (Leviathan et al., 2022; Chen et al., 2023). Jacobi decoding typically initiates a new sequence with $[PAD]$ tokens, then iteratively verifies and updates the sequence by solving Jacobi equations until a fixed point is reached (Song et al., 2020; Santilli et al., 2023). Multiple heads decoding predicts multiple next tokens by extra output heads. It then constructs multiple candidate sequences by combining these heads, and verifies them in parallel by the original model (Stern et al., 2018; Cai et al., 2024).

This paper focus on multiple head decoding, particularly MEDUSA proposed in (Cai et al., 2024). Given the original model's last hidden states $\mathbf{h}_t$ at last input position t, MEDUSA introduces K additional decoding heads to h_t. The k-th head is designed to predict the token in the $(t+k+1)$-th position, while the original head predicts the $(t+1)$-th position. Denote $\mathbf{p}^{(k)}$ as the predicted vocabulary distribution of k-th head. The top predictions from $\mathbf{p}^{(k)}$ are used to generate candidate sequences, with each candidate being a combination of the top predictions from different heads. MEDUSA uses a fixed set of combining patterns. By merging their common parts, these patterns are represented as a tree. This tree structure is constructed by estimating the accuracy via a calibration dataset. After generating candidates using the fixed tree structure, MEDUSA verifies them in parallel through tree attention (Miao et al., 2023; Cai et al., 2024), which involves incorporating the tree structure into the attention mask.

Although the fixed tree structure captures certain inherent biases of MEDUSA heads, it may not fully account for context-dependent variations. We believe that a dynamic tree structure can handle context dependency better and improve the decoding efficiency of LLMs. In this paper, we propose

Table 1: MT-Bench results with Vicuna-7B model

Method	Speed up	Generation Quality
MEDUSA-1	2.50	5.21
MEDUSA-1 Dynamic	2.66	5.17
MEDUSA-2	3.32	5.24
MEDUSA-2 Dynamic	3.51	5.21

a simple and efficient strategy to dynamically construct the tree structure: selecting top-n candidates from all possible combinations (we will show how to efficiently do this). Experiments demonstrate that dynamic tree improves the decoding efficiency in terms of tokens per inference. Our code is available at `https://github.com/zzd1992/MEDUSA-Plus`.

2 METHODOLOGY

The proposed method first dynamically generates candidates, then prepares the buffers of dynamic tree attention based on those candidates. Ideally, candidates should be sampled according to their joint distribution. However, it is not directly accessible. As an alternative, we approximate the joint distribution using the Cartesian product of marginal distributions, which is provided by MEDUSA heads. Let $p_i^{(k)}$ be the i-th top prediction of k-th MEDUSA head. Then the probability of sequence $(i_1, i_2, \dots i_k)$ is

$$P(i_1, i_2, \dots i_k) = \prod_{j=1}^{k} p_{i_j}^{(j)} \tag{1}$$

By emulating the Cartesian Product of marginal distributions, we generate all possible candidates. We only consider the top-m predictions of each marginal distribution, i.e. there are $\sum_{k=1}^{K} m^k$ possible candidates. Then we select top-n candidates with the highest probability. This can be done efficiently by a priority queue, as shown in algorithm 1. The computational complexity is $O(Knm \log n)$. Following (Cai et al., 2024), we set $K = 4, n = 64$. We set $m = 32$. Thus, the actual complexity for candidates generating is quite small. The selected candidates form the structure of a tree due to the Cartesian product of marginal distributions.

$$P(i_1, i_2, \dots i_k) = P(i_1, i_2, \dots i_{k-1})P(i_k) \leq P(i_1, i_2, \dots i_{k-1}) \tag{2}$$

If candidate $(i_1, i_2, \dots i_k)$ is selected, then it's parent $(i_1, i_2, \dots i_{k-1})$ is also selected. Once the candidates are generated, we prepare the buffers of dynamic tree attention, specifically the position embedding and attention mask. This is achieved by emulating the generated candidates with a computational complexity of $O(Kn)$. Thus, the overall computational complexity is quite small.

3 EXPERIMENTS

We evaluate the proposed method in terms of tokens per inference (i.e. speed up) and generation quality. The evaluation is carried out using MT-Bench (Zheng et al., 2023), a multi-turn, conversational-format benchmark. We use the same model of MEDUSA-1 and MEDUSA-2 which are trained with fixed backbone and trainable backbone respectively. Currently, our evaluation is limited to Vicuna-7B model (Chiang et al., 2023). Generation quality is measured using the single judgment method on MT-Bench. The results are presented in Table 1. The proposed method improves the decoding efficiency of MEDUSA-1 and MEDUSA-2 while maintaining the generation quality. We provide a visualization of tree attention mask in Figure B. The dynamic tree structure shares common parts with the fixed tree structure, but can adapt to context dependencies. In terms of tokens per second, our method is approximately 10% slower than MEDUSA. However, our implementation is not yet optimized. We are confident that the overhead associated with dynamic tree construction can be substantially reduced through further engineering efforts (see appendix C).

4 DISCUSSION

In this paper, we replace the fixed tree attention with dynamic tree attention on multiple head decoding, specifically in the context of MEDUSA. We propose a simple and low complexity strategy to generate candidates and construct the dynamic tree structure. Preliminary experiments show that the proposed method improves the decoding efficiency of multiple head decoding for LLMs while maintaining the generation quality. This result demonstrates the potential for improvement of multiple head decoding in candidate generation. For future work, we plan to improve the proposed method in the following ways:

- **Optimize the overhead**: optimize candidate generation process to make our method more competitive in terms of wall time.
- **Improve joint distribution approximation**: currently, the joint distribution is approximated by the Cartesian product of marginal distributions. We will explore better strategies to approximate it.

REFERENCES

OpenAI Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haim ing Bao, Mo Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Made laine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, Derek Chen, Sully Chen, Ruby Chen, Jason Chen, Mark Chen, Benjamin Chess, Chester Cho, Casey Chu, Hyung Won Chung, Dave Cummings, Jeremiah Currier, Yunxing Dai, Cory Decareaux, Thomas Degry, Noah Deutsch, Damien Deville, Arka Dhar, David Dohan, Steve Dowling, Sheila Dunning, Adrien Ecoffet, Atty Eleti, Tyna Eloundou, David Farhi, Liam Fedus, Niko Felix, Sim'on Posada Fishman, Juston Forte, Is abella Fulford, Leo Gao, Elie Georges, Christian Gibson, Vik Goel, Tarun Gogineni, Gabriel Goh, Raphael Gontijo-Lopes, Jonathan Gordon, Morgan Grafstein, Scott Gray, Ryan Greene, Joshua Gross, Shixiang Shane Gu, Yufei Guo, Chris Hallacy, Jesse Han, Jeff Harris, Yuchen He, Mike Heaton, Jo hannes Heidecke, Chris Hesse, Alan Hickey, Wade Hickey, Peter Hoeschele, Brandon Houghton, Kenny Hsu, Shengli Hu, Xin Hu, Joost Huizinga, Shantanu Jain, Shawn Jain, Joanne Jang, Angela Jiang, Roger Jiang, Haozhun Jin, Denny Jin, Shino Jomoto, Billie Jonn, Heewoo Jun, Tomer Kaftan, Lukasz Kaiser, Ali Kamali, Ingmar Kanitscheider, Nitish Shirish Keskar, Tabarak Khan, Logan Kilpatrick, Jong Wook Kim, Christina Kim, Yongjik Kim, Hendrik Kirchner, Jamie Ryan Kiros, Matthew Knight, Daniel Kokotajlo, Lukasz Kondraciuk, Andrew Kondrich, Aris Konstantinidis, Kyle Kosic, Gretchen Krueger, Vishal Kuo, Michael Lampe, Ikai Lan, Teddy Lee, Jan Leike, Jade Leung, Daniel Levy, Chak Ming Li, Rachel Lim, Molly Lin, Stephanie Lin, Ma teusz Litwin, Theresa Lopez, Ryan Lowe, Patricia Lue, Anna Makanju, Kim Malfacini, Sam Manning, Todor Markov, Yaniv Markovski, Bianca Martin, Katie Mayer, Andrew Mayne, Bob McGrew, Scott Mayer McKinney, Christine McLeavey, Paul McMillan, Jake McNeil, David Medina, Aalok Mehta, Jacob Menick, Luke Metz, Andrey Mishchenko, Pamela Mishkin, Vinnie Monaco, Evan Morikawa, Daniel P. Mossing, Tong Mu, Mira Murati, Oleg Murk, David M'ely, Ashvin Nair, Reiichiro Nakano, Rajeev Nayak, Arvind Neelakantan, Richard Ngo, Hyeonwoo Noh, Ouyang Long, Cullen O'Keefe, Jakub W. Pachocki, Alex Paino, Joe Palermo, Ashley Pantuliano, Giambattista Parascandolo, Joel Parish, Emy Parparita, Alexandre Passos, Mikhail Pavlov, Andrew Peng, Adam Perelman, Filipe de Avila Belbute Peres, Michael Petrov, Henrique Pondé de Oliveira Pinto, Michael Pokorny, Michelle Pokrass, Vitchyr H. Pong, Tolly Powell, Alethea Power, Boris Power, Elizabeth Proehl, Raul Puri, Alec Radford, Jack W. Rae, Aditya Ramesh, Cameron Raymond, Francis Real, Kendra Rimbach, Carl Ross, Bob Rotsted, Henri Roussez, Nick Ryder, Mario D. Saltarelli, Ted Sanders, Shibani Santurkar, Girish Sastry, Heather Schmidt, David Schnurr, John Schulman, Daniel Selsam, Kyla Sheppard, Toki Sherbakov, Jessica Shieh, Sarah Shoker, Pranav Shyam, Szymon Sidor, Eric Sigler, Maddie Simens, Jordan Sitkin, Katarina Slama, Ian Sohl, Benjamin Sokolowsky, Yang Song, Natalie Staudacher, Felipe Petroski Such, Natalie Summers, Ilya Sutskever, Jie Tang, Nikolas A. Tezak, Madeleine Thompson, Phil Tillet, Amin Tootoonchian, Elizabeth Tseng, Pre-

ston Tuggle, Nick Turley, Jerry Tworek, Juan Felipe Cer'on Uribe, Andrea Vallone, Arun Vijayvergiya, Chelsea Voss, Carroll L. Wainwright, Justin Jay Wang, Alvin Wang, Ben Wang, Jonathan Ward, Jason Wei, CJ Weinmann, Akila Welihinda, Peter Welinder, Jiayi Weng, Lilian Weng, Matt Wiethoff, Dave Willner, Clemens Winter, Samuel Wolrich, Hannah Wong, Lauren Workman, Sherwin Wu, Jeff Wu, Michael Wu, Kai Xiao, Tao Xu, Sarah Yoo, Kevin Yu, Qim ing Yuan, Wojciech Zaremba, Rowan Zellers, Chong Zhang, Marvin Zhang, Shengjia Zhao, Tianhao Zheng, Juntang Zhuang, William Zhuk, and Barret Zoph. Gpt-4 technical report. 2023. URL https://api.semanticscholar.org/CorpusID:257532815.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeff Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Ma teusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. *ArXiv*, abs/2005.14165, 2020. URL https://api.semanticscholar.org/CorpusID:218971783.

Tianle Cai, Yuhong Li, Zhengyang Geng, Hongwu Peng, Jason D. Lee, De huai Chen, and Tri Dao. Medusa: Simple llm inference acceleration framework with multiple decoding heads. *ArXiv*, abs/2401.10774, 2024. URL https://api.semanticscholar.org/CorpusID:267061277.

Charlie Chen, Sebastian Borgeaud, Geoffrey Irving, Jean-Baptiste Lespiau, L. Sifre, and John M. Jumper. Accelerating large language model decoding with speculative sampling. *ArXiv*, abs/2302.01318, 2023. URL https://api.semanticscholar.org/CorpusID:256503945.

Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality, March 2023. URL https://lmsys.org/blog/2023-03-30-vicuna/.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. In *International Conference on Machine Learning*, 2022. URL https://api.semanticscholar.org/CorpusID:254096365.

Xupeng Miao, Gabriele Oliaro, Zhihao Zhang, Xinhao Cheng, Zeyu Wang, Rae Ying Yee Wong, Zhuoming Chen, Daiyaan Arfeen, Reyna Abhyankar, and Zhihao Jia. Specinfer: Accelerating generative llm serving with speculative inference and token tree verification. *ArXiv*, abs/2305.09781, 2023. URL https://api.semanticscholar.org/CorpusID:258740799.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019. URL https://api.semanticscholar.org/CorpusID:160025533.

Andrea Santilli, Silvio Severino, Emilian Postolache, Valentino Maiorca, Michele Mancusi, Riccardo Marin, and Emanuele Rodolà. Accelerating transformer inference for translation via parallel decoding. In *Annual Meeting of the Association for Computational Linguistics*, 2023. URL https://api.semanticscholar.org/CorpusID:258741236.

Noam M. Shazeer. Fast transformer decoding: One write-head is all you need. *ArXiv*, abs/1911.02150, 2019. URL https://api.semanticscholar.org/CorpusID:207880429.

Yang Song, Chenlin Meng, Renjie Liao, and Stefano Ermon. Accelerating feedforward computation via parallel nonlinear equation solving. In *International Conference on Machine Learning*, 2020. URL https://api.semanticscholar.org/CorpusID:235422735.

Mitchell Stern, Noam M. Shazeer, and Jakob Uszkoreit. Blockwise parallel decoding for deep autoregressive models. In *Neural Information Processing Systems*, 2018. URL https://api.semanticscholar.org/CorpusID:53208380.

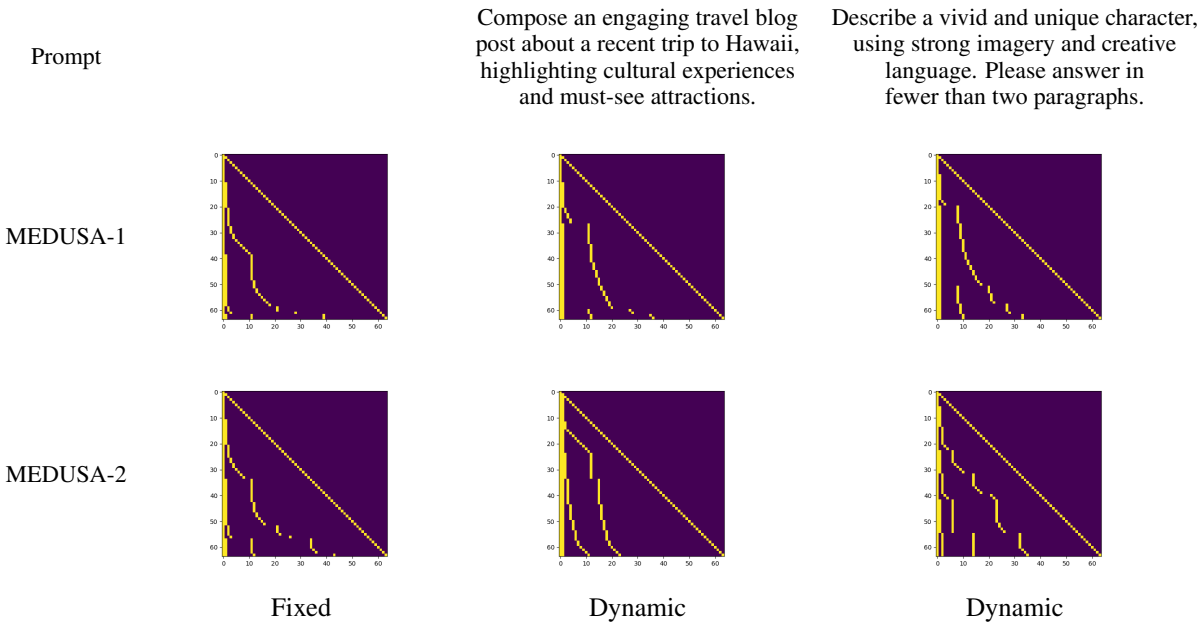
Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, Hao Zhang, Joseph E Gonzalez, and Ion Stoica. Judging llm-as-a-judge with mt-bench and chatbot arena. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 46595–46623. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/91f18a1287b398d378ef22505bf41832-Paper-Datasets_and_Benchmarks.pdf.

Algorithm 1 Candidate Generation

Require: Top-m probability of K MEDUSA heads $\mathbf{P} \in \mathbb{R}^{K \times m}$ and number of candidates n
Ensure: candidate set S
  Initialize candidate set $S \in \emptyset$
  for $i = 1$ to m **do**
    Add $(\mathbf{P}[1, i], i, 1)$ to S
  end for
  for $k = 2$ to K **do**
    Initialize a priority queue Q with maximum size n
    for $(p, idx, depth) \in S$ **do**
      Push $(p, idx, depth)$ to Q
      if $depth = k - 1$ **then**
        for $i = 1$ to m **do**
          Push $(p \cdot \mathbf{P}[k, i], idx \times m + i, k)$ to Q
        end for
      end if
    end for
    $S \leftarrow set(\mathrm{Q})$
  end for
  Return S

A Candidate Generation Algorithm

B Visualization of Tree Attention Mask



C Overhead Analysis

The overhead of MEDUSA comprises two components: candidate generation and posterior evaluation. For the Vicuna-7B, it occupies 20% of the model's pure inference time. MEDUSA-Dynamic, on the other hand, has three components contributing to its overhead: candidate generation, buffer modification, and posterior evaluation. In the case of Vicuna-7B, it accounts for 33% of the model's pure inference time. It is worth noting that the posterior evaluation process is identical for both MEDUSA and MEDUSA-Dynamic.

At present, the candidate generation in MEDUSA-Dynamic is not implemented using Algorithm 1. Instead, we have employed a trivial top-K sorting approach for all possible candidates. The computational complexity associated with Algorithm 1 and buffer modification is low. We are convinced that, with an optimized implementation, the overhead of MEDUSA-Dynamic would be nearly comparable to that of MEDUSA.