Towards Real-Time and Personalized Code Generation

Han Xu
University of Illinois at Urbana-Champaign
Urbana, IL, USA
hanxu8@illinois.edu

Xingyuan Wang
Meta Platforms Inc.
Seattle, WA, USA
xywang0520@gmail.com

Haipeng Chen
William & Mary
Williamsburg, VA, USA
hchen23@wm.edu

Abstract

Large language models (LLMs) have transformed automated code generation. However, their high computational demands often lead to server overload and increased latency in SaaS deployments. To address this, we present SpeCoder, a framework that accelerates server-side code generation using speculative sampling (SpS) and supervised fine-tuning (SFT). SpS allows lower latency in the code generation, whereas SFT enables more personalized code generation tailored to the user's needs.

ACM Reference Format:
Han Xu, Xingyuan Wang, and Haipeng Chen. 2024. Towards Real-Time and Personalized Code Generation. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management (CIKM '24), October 21–25, 2024, Boise, ID, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3627673.3679071

1 Introduction

The advent of large language models (LLMs) such as GPT-4 [9], Llama [11] and Mistral [5] has revolutionized automated code-related tasks, particularly code generation - the production of source code from natural language descriptions. This area has garnered substantial interest from academia, with research-focused models like StarCoder [7] and CodeLlama [10], and industry, as evidenced by services like GitHub Copilot and Amazon CodeWhisperer. These tools leverage groundbreaking LLM models for code generation to facilitate software development.

However, the substantial computational resources required to run LLMs, especially those with over 10 billion parameters, often necessitate LLM deployments as Software as a Service (SaaS). Given the expansive user base of such services (e.g., GitHub Copilot's million-plus subscribers) and the high frequency of code generation requests, server capacity can become overwhelmed. This results in increased latency, hindering the productivity of developers who rely on timely code generation. Additionally, scaling server infrastructure to mitigate this issue is both complex and expensive.

To address this challenge, we present SpeCoder, an ongoing research project that leverages speculative sampling (SpS) to accelerate server-side LLM code generation. SpeCoder employs an online SpS approach to speed up code generation, along with supervised fine-tuning (SFT) to align LLM output with user preferences.

CIKM '24, October 21–25, 2024, Boise, ID, USA

ACM ISBN 979-8-4007-0436-9/24/10
https://doi.org/10.1145/3627673.3679071

2 Background

2.1 Speculative Sampling (SpS)

Recently, SpS has been introduced as a technique to accelerate the inference process of a target language model M_p using an approximate model M_q [1, 6]. This method involves three steps: (1) the more efficient model M_q generates approximations for k tokens, where $k \in \mathbb{Z}^+$; (2) M_p evaluates all the guesses, accepting those that lead to an identical distribution; and (3) if all guesses are accepted, M_p samples an additional token from the adjusted distribution. If any guess is rejected, M_p corrects the rejected token. This procedure ensures the generation of 1 to $k+1$ tokens in one run.

The acceleration achieved by SpS is attributed to the lower computational and memory bandwidth costs associated with generating these guesses, combined with a high likelihood of acceptance. According to the original literature, SpS demonstrates 2x-2.5x improvements on Chinchilla 70B [3] for natural language summarization and code generation. Other studies have verified that SpS accelerates reasoning tasks by 2.18x-2.62x on Llama2-70B models with minimal loss of reasoning capabilities [12].

2.2 Supervised Fine-Tuning (SFT)

Large-scale language models have demonstrated remarkable progress in understanding and generating language. However, their pre-training typically relies on unsupervised learning on massive text datasets, which may not fully equip them for specialized tasks.

SFT addresses this by adapting these pre-trained models to specific downstream tasks using labeled data. This process helps the model learn the unique patterns and nuances within the labeled examples. By adjusting its parameters according to the specific data distribution and task requirements, the model becomes specialized and performs better on the target task. Moreover, SFT can be paired with Parameter-Efficient Fine-Tuning (PEFT) techniques like LoRA [4], QLoRA [2, 8] and LoRETTA [13], which accelerate the fine-tuning process while reducing memory consumption.

3 Method

As illustrated in Figure 1, building on top of SpS, our SpeCoder approach utilizes a draft model M_q that runs locally, as well as a target model M_p that runs in the cloud. Additionally, assuming the cloud model has been extensively trained on diverse datasets, employing SFT enhances the alignment between the local draft model and user preferences.

Local The local draft model, M_q, is designed as a smaller and more efficient model with a few billion parameters, allowing it to run efficiently on local CPUs/GPUs with low latency. To further improve response times, a local Least Recently Used (LRU) cache of size c stores previous prompts and the corresponding accepted

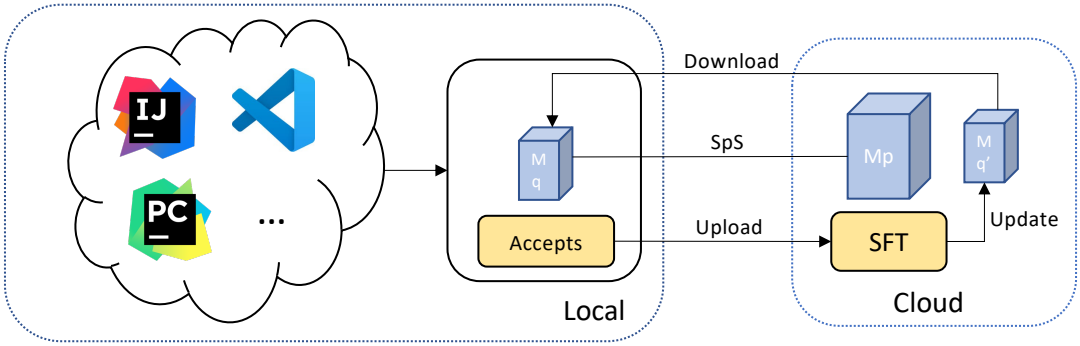

Figure 1: An illustration of the SpeCoder workflow.

responses. When a request falls outside the cache, M_q generates a response in the usual SpS fashion. Additionally, whenever a user accepts a generated response, the local module uploads both the prompt and response to the cloud.

Cloud The cloud hosts the target model M_p, which evaluates the local model M_q's guesses. Once the uploaded preference datasets exceed a threshold t, the cloud leverages a copy of the draft model and employs SFT to fine tune M_q into an updated M'_q. After fine-tuning, the updated M'_q replaces the local M_q model. This iterative process continuously updates the local model to better align with user preferences.

4 Conclusion and Relevance

In conclusion, SpeCoder presents a novel approach to accelerating code generation. By leveraging SpS and SFT, our research strategically distributes the computational load between local and cloud-based models. To the best of our knowledge, this is the first application of SpS to address the challenges of LLM code generation within cloud environments. This work is highly relevant to the CIKM community, as it addresses the critical need for efficient data processing and scalable machine learning systems.

5 Speaker's Bio

Han Xu is a graduate student from the University of Illinois Urbana-Champaign. His expertise spans database systems, natural language processing, and transformer architectures.

Xingyuan Wang is a senior software engineer at Meta, specializing in GPU programming and AI/ML infrastructure systems. She holds a Master's degree in Computer Science from the University of Southern California.

References

[1] Charlie Chen, Sebastian Borgeaud, Geoffrey Irving, Jean-Baptiste Lespiau, Laurent Sifre, and John Jumper. 2023. Accelerating Large Language Model Decoding with Speculative Sampling. arXiv:2302.01318 [cs.CL]

[2] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. QLoRA: Efficient Finetuning of Quantized LLMs. In *Advances in Neural Information Processing Systems*, A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (Eds.), Vol. 36. Curran Associates, Inc., 10088–10115.

[3] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, and et al. 2022. Training Compute-Optimal Large Language Models. arXiv:2203.15556 [cs.CL]

[4] Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-Rank Adaptation of Large Language Models. In *International Conference on Learning Representations*.

[5] Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, and et al. 2023. Mistral 7B. arXiv:2310.06825 [cs.CL]

[6] Yaniv Leviathan, Matan Kalman, and Yossi Matias. 2023. Fast inference from transformers via speculative decoding. In *Proceedings of the 40th International Conference on Machine Learning* (Honolulu, Hawaii, USA) *(ICML'23)*.

[7] Anton Lozhkov, Raymond Li, Loubna Ben Allal, Federico Cassano, Joel Lamy-Poirier, Nouamane Tazi, and et al. 2024. StarCoder 2 and The Stack v2: The Next Generation. arXiv:2402.19173 [cs.SE]

[8] Haowei Ni, Shuchen Meng, Xupeng Chen, Ziqing Zhao, Andi Chen, Panfeng Li, Shiyao Zhang, Qifu Yin, Yuanqing Wang, and Yuxi Chan. 2024. Harnessing Earnings Reports for Stock Predictions: A QLoRA-Enhanced LLM Approach. *arXiv preprint arXiv:2408.06634* (2024).

[9] OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, and et al. 2024. GPT-4 Technical Report. arXiv:2303.08774 [cs.CL]

[10] Baptiste Rozière, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, and et al. 2024. Code Llama: Open Foundation Models for Code. arXiv:2308.12950 [cs.CL]

[11] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, and et al. 2023. Llama 2: Open Foundation and Fine-Tuned Chat Models. arXiv:2307.09288 [cs.CL]

[12] Han Xu, Jingyang Ye, Yutong Li, and Haipeng Chen. 2024. Can Speculative Sampling Accelerate ReAct Without Compromising Reasoning Quality?. In *The Second Tiny Papers Track at ICLR 2024*.

[13] Yifan Yang, Jiajun Zhou, Ngai Wong, and Zheng Zhang. 2024. LoRETTA: Low-Rank Economic Tensor-Train Adaptation for Ultra-Low-Parameter Fine-Tuning of Large Language Models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*. 3161–3176.