Towards Efficient Compound Large Language Model System Serving in the Wild

Yifei Zhu*, Botao Zhu*, Chen Chen†, Xiaoyi Fan‡
*UM-SJTU Joint Institute, Shanghai Jiao Tong University
†John Hopcroft Center, Shanghai Jiao Tong University
‡ Jiangxing Intelligence Inc.
Email: yifei.zhu@sjtu.edu.cn, zhubotao@sjtu.edu.cn, chen-chen@sjtu.edu.cn, xiaoyifan@jiangxingai.com

***Abstract*—Utilizing compound Large Language Model (LLM) systems, instead of a monolithic LLM model, is gradually becoming a practical solution to realize a diverse range of industry applications. In compound LLM systems, an LLM collaborates with other external tools, APIs, or LLMs to offer intelligent services. In this poster, we identify the unique challenges, namely temporal and topological uncertainty, brought about by compound LLM systems in system serving. We then propose a priority-based scheduling policy to schedule different stages in DAG-represented compound LLM systems. The preliminary results show promising performance of uncertainty-aware scheduling policies.**

***Index Terms*—Large language model, job scheduling, uncertainty**

I. Introduction

While Large Language Models (LLMs) have showcased promising potential and capabilities in performing various natural language tasks, such as question answering and translation, they encounter limitations when providing high-quality solutions to domain-specific queries in practical settings. Consequently, there is a noticeable trend towards transitioning from monolithic models to compound LLM systems. In these systems, LLMs collaborate with other external models, tools, APIs, retrievals, or even other LLMs to meet the demands of practical intelligent applications.

Researchers have explored leveraging LLM either as task planners, actual executors, or both, in collaboration with other components. For instance, TaskMatrix [1] employs an LLM to comprehend goals and context through analysis of users' natural language input. It then generates executable code to select and invoke appropriate APIs for accomplishing various tasks. Similarly, OptiGuide [2] utilizes an LLM to translate user queries from natural language into optimization problems that can be tackled by existing solvers. The solver results are then fed back to the LLM to produce user-friendly reports.

In the industry, our industry co-author has also discovered that users generally prefer compound LLM systems over single monolithic models. This preference stems from several practical reasons: (1) Performance issues. Industrial applications often rely on domain-specific proprietary solvers or simulation tools. Fine-tuning an LLM to match the performance of these proprietary tools remains a challenge. (2) Trust and management issues. Unlike the explicit physics mechanisms embedded in existing tools, the generative nature of LLMs and the possibility of hallucination diminish users' trust in these systems. (3) Cost issues. The on-site computing costs and engineering investment, including data cleaning and QA pair preparation, required for training a satisfactory LLM are substantial for users.

This work is supported by the National Natural Science Foundation of China (Grant No. 62302292). Corresponding author: Yifei Zhu.

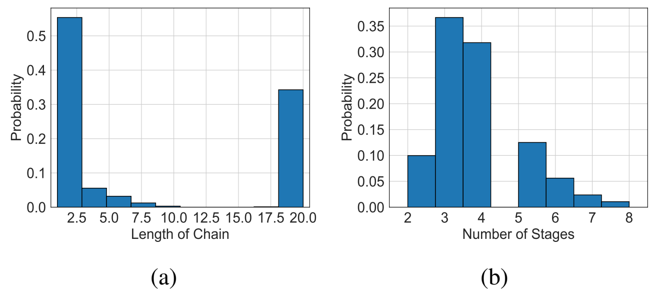

Fig. 1: (a) Length distribution of code generation application (b) Distribution of the number of generated stages of task automation application

To effectively serve diverse compound LLM system running requests, it is common practice to represent each component, including the LLM, and their relationships in a compound LLM system as a directed acyclic graph (DAG) following the conventional cluster serving framework. However, running scheduling algorithms in such systems typically requires information about the DAG topology and task durations, which may be inaccessible in compound LLM systems. As a result, existing algorithms, originally designed for conventional model inference tasks and big data analytics tasks, become inadequate when dealing with the uncertainty in topological structure and execution duration inherent in compound LLM systems.

This poster takes the initiative to address the unique challenge posed to system serving by the emerging compound LLM systems. We propose a simple yet effective priority-based scheduling algorithm to manage a variety of compound LLM requests. We then evaluate the performance of our scheduling policy through testbed implementations and demonstrate promising results in job completion time.

II. Measurement and motivation

We first conduct a preliminary measurement to demonstrate the topological uncertainty in compound LLM systems. Our measurement is conducted on a Ubuntu 20.4 machine with an NVIDIA A800 GPU card. For the LLM model, we employ Llama2-7B [3], recognized as one of the top-performing open-source LLMs during our experimentation. We choose two representative compound LLM applications: (1) Code generation with AutoGen [4] using HumanEval [5] benchmark: This application uses LLM to generate and refine the code iteratively for a given task. The workflow involves sequential stages of code evaluation and LLM inference. (2) Task automation with HuggingGPT [6] using TaskBench [7] benchmark. This application takes a user's request and asks the LLM to generate a workflow consisting of available tools to fulfill the user's request.

Fig. 1a shows the distribution of the lengths of workflows in code generation applications. We can see that the length shows great uncertainty, ranging from 1 to 20. Fig. 1b shows the diverse number of stages in the workflow generated by LLM in the task automation application. The measurement results reveal that compound LLM systems exhibit diverse topological patterns in practice. Moreover, it's noted that certain workflows are not pre-determined but rather iteratively generated by the LLM. This uncertainty, coupled with the variability in LLM execution as discussed in prior studies [8], poses significant challenges to the applicability of existing scheduling policies.

III. Design and evaluation

A. Priority-based scheduling

In addition to our previously implemented applications, we introduce another compound LLM-based sorting application, comprising four LLMs utilizing the Graph of Thought (GoT) framework [9]. We then propose a simple priority-based scheduling policy that incorporates awareness of uncertainty to enhance performance. This priority-based policy, named PS-TCS, predefines a preference order for different applications (Task automation > Code generation > Sorting) and prioritizes stages accordingly during scheduling. The rationale behind this policy lies in the increasing DAG topological uncertainty across applications, progressing from sorting with a fixed predetermined DAG to task automation where the DAG is unknown beforehand and only revealed after running the task planner stage.

B. Preliminary results

To evaluate the performance of different DAG scheduling algorithms, we build a simple simulator based on the measurement results. The simulator simulates the cloud environment for serving heterogeneous compound LLM applications with different numbers of regular executors and LLM executors. Regular executors are dedicated to executing tasks like code generation, while LLM executors handle LLM inference requests. With the simulated environments, we randomly generate 500 compound LLM requests in equal proportion for three applications using their corresponding benchmark. The arrival of each request for applications follows a Poisson distribution. We compare the performance with First Come First Serve (FCFS) and Random Scheduling (RS). FCFS prioritizes the stages of the application that come first. RS randomly selects stages for scheduling regardless of their arrival time or the belonged application. The average job completion time (JCT) is recorded. We change the resource settings, i.e., the number of LLM executors (L) and Regular executors (R) from 10 to 40, and evaluate the performance of these scheduling policies. The results are shown in Table I. We can see that the priority-based method PS-TCS achieves the best performance under different resource settings. It reduces job completion by up to 47% compared to traditional methods like FCFS and RS.

TABLE I: Average JCT of Different Scheduling Polices

Method	R=10, L=10	R=20, L=20	R=30, L=30	R=40, L=40
FCFS	893	347	155	87
RS	812	298	143	87
PS-TCS	**474**	**197**	**119**	**86**

IV. Conclusion

We explore the compound LLM system serving problem, where each request involves various LLMs and external tools. We highlight the topological uncertainty in compound LLM systems and its impact on existing cluster schedulers. Using a simple yet effective priority-based scheduling policy, we demonstrate the potential of integrating uncertainty for performance improvement. These initial findings suggest the need for new job scheduling systems and call for fresh abstraction frameworks tailored to compound LLM systems.

References

[1] Y. Liang, C. Wu, T. Song, W. Wu, Y. Xia, Y. Liu, Y. Ou, S. Lu, L. Ji, S. Mao, Y. Wang, L. Shou, M. Gong, and N. Duan, "Taskmatrix.ai: Completing tasks by connecting foundation models with millions of apis," 2023.

[2] B. Li, K. Mellou, B. Zhang, J. Pathuri, and I. Menache, "Large language models for supply chain optimization," 2023.

[3] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar *et al.*, "Llama: Open and efficient foundation language models," *arXiv preprint arXiv:2302.13971*, 2023.

[4] Q. Wu, G. Bansal, J. Zhang, Y. Wu, S. Zhang, E. Zhu, B. Li, L. Jiang, X. Zhang, and C. Wang, "Autogen: Enabling next-gen llm applications via multi-agent conversation framework," *arXiv preprint arXiv:2308.08155*, 2023.

[5] M. Chen, J. Tworek, H. Jun, Q. Yuan, H. P. d. O. Pinto, J. Kaplan, H. Edwards, Y. Burda, N. Joseph, G. Brockman *et al.*, "Evaluating large language models trained on code," *arXiv preprint arXiv:2107.03374*, 2021.

[6] Y. Shen, K. Song, X. Tan, D. Li, W. Lu, and Y. Zhuang, "Hugginggpt: Solving ai tasks with chatgpt and its friends in hugging face," *Proc. NeurIPS*, vol. 36, 2024.

[7] Y. Shen, K. Song, X. Tan, W. Zhang, K. Ren, S. Yuan, W. Lu, D. Li, and Y. Zhuang, "Taskbench: Benchmarking large language models for task automation," *arXiv preprint arXiv:2311.18760*, 2023.

[8] Z. Zheng, X. Ren, F. Xue, Y. Luo, X. Jiang, and Y. You, "Response length perception and sequence scheduling: An llm-empowered llm inference pipeline," *Proc. NeurIPS*, vol. 36, 2024.

[9] M. Besta, N. Blach, A. Kubicek, R. Gerstenberger, L. Gianinazzi, J. Gajda, T. Lehmann, M. Podstawski, H. Niewiadomski, P. Nyczyk *et al.*, "Graph of thoughts: Solving elaborate problems with large language models," *arXiv preprint arXiv:2308.09687*, 2023.