Poster: PipeLLM: Pipeline LLM Inference on Heterogeneous Devices with Sequence Slicing

Ruilong Ma*, Jingyu Wang*, Qi Qi†, Xiang Yang, Haifeng Sun, Zirui Zhuang, Jianxin Liao
{maruilong,wangjingyu,qiqi8266,yangxiang,hfsun,zhuangzirui}@bupt.edu.cn,jxlbupt@gmail.com
Beijing University of Posts and Telecommunications
Beijing, China

ABSTRACT

Large Language Models (LLMs) has fostered the creation of innovative requirements. Locally deployed LLMs for micro-enterprise mitigates potential issues such as privacy infringements and sluggish response. However, they are hampered by the limitations in computing capability and memory space of possessed devices. We introduce PipeLLM, which allocates the model across devices commensurate with their computing capabilities. It enables the parallel execution of layers with slicing input sequence along the token dimension. PipeLLM demonstrates the potential to accelerate LLM inference with heterogeneity devices, offering a solution for LLM deployment in micro-enterprise hardware environment.

CCS CONCEPTS

• **Computing methodologies** → **Parallel algorithms**; **Distributed artificial intelligence**;

KEYWORDS

LLM Inference Acceleration; Pipeline Inference; Model Deployment

ACM Reference Format:
Ruilong Ma[1], Jingyu Wang[1], Qi Qi[2], Xiang Yang, Haifeng Sun, Zirui Zhuang, Jianxin Liao. 2023. Poster: PipeLLM: Pipeline LLM Inference on Heterogeneous Devices with Sequence Slicing. In *ACM SIGCOMM 2023 Conference (ACM SIGCOMM '23), September 10, 2023, New York, NY, USA.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3603269.3610856

1 INTRODUCTION

The achievements of LLMs are credited to the extensive empirical knowledge that digs from vast data, as represented by trained weights. However, proficiently harnessing the knowledge requires substantial computational resources and memory storage. Certain micro-enterprises face the necessity of submitting context analysis requests to remote LLMs, managed by AI corporations, which respond slowly and present privacy risks for data-intensive operations. Meanwhile, the limitations in capabilities of existing devices and financial constraints in acquiring high-performance devices impede the deployment of private, local LLMs.

A viable approach could be to exploit possessed devices for parallel inference. However, implementing parallelism in inference presents some challenges:

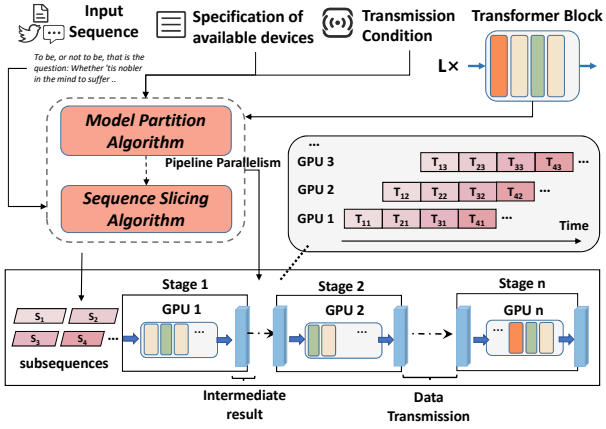


Figure 1: Workflow of the PipeLLM architecture

Parallelism for Sparse Requests: Previous methods [1, 3, 6] typically partitions the aggregated batch data into micro-batches for parallel pipelining. However, it is ineffective for inference in micro-enterprises that often deal with sparse requests, potentially consisting of a single data instance with an extended sequence.

Limited Transmissions: Communication is a major performance bottleneck in LLM training and inference [2]. The utilization of operator-level parallelism [4, 7] is hindered by the communication overhead required for synchronization. Deploying high-velocity transmissions like NV-Link as a remedy remains financially unattainable for micro-enterprises.

Proper Distribution for Heterogeneous Devices: To employ all available computational resources, the integration of heterogeneous devices becomes a necessity. The workload schedule should take the heterogeneous capacities and disparate transmission condition into decision, rather than distributing evenly across devices.

To address the above challenges, we design *PipeLLM*. Exploiting the autoregressive nature of LLMs, PipeLLM slices and pipelines the input sequence from the token dimension. It incurs minimal communication cost as only transmitting the intermediate results. It also shields hardware discrepancies by distributing LLMs based on computing capabilities, ensuring effective resource utilization and making the underlying hardware transparent to users.

2 METHOD

LLMs are typically composed of multiple stacked transformer blocks. The mask attention mechanism in the transformer block enables the computation for the hidden state of a token t_i in the input sequence S involves only the preceding tokens $t_0, ..., t_{i-1}$. Motivating by this

*These authors contributed equally to this work.
†Corresponding authors.

ACM SIGCOMM '23, September 10, 2023, New York, NY, USA

ACM ISBN 979-8-4007-0236-5/23/09. https://doi.org/10.1145/3603269.3610856

characteristic, we pipeline the inference to accelerate from the token dimension as shown in Fig. 1. LLMs are partitioned considering specifications of available devices and transmission conditions, then distributed across multiple devices. Once a sequence S arrives, it is sliced into subsequences $s_0, ..., s_n$ and executed sequentially across devices. Each device d_j handles a stage of computation. When device d_j completes the computation for s_i, the intermediate result is transmitted to device d_{j+1}. T_{ij} denotes the execution time of subsequence s_i on device d_j. Since the performance of the slowest stage sets the upper limit on the entire pipeline, it is crucial to minimize the maximum execution time among all T_{ij}.

We first optimize the pipeline by distributing the LLM that aligns with their computational capabilities while considering transmission overhead. LLM is divided between arithmetic operation layers. Given a constant sequence of devices, we design a dynamic programming algorithm to iterate the minimum execution time of the first k layers of the LLM across former m devices. Once the optimal partition is ascertained, sequences of the same length will exhibit similar execution times on the different devices.

Then we determine the optimal slicing scheme for each sequence. Terapipe [5] observed that the computational load is higher for tokens located towards the end of a sequence since each token's computation involves self-analysis and the fusion of information from preceding tokens. Therefore, an ideal slicing should include longer slices at the beginning and shorter slices towards the end. Additionally, using a more granular slicing may lead to underutilization of computational power, while adopting a coarser slicing decreasing parallelism. Hence, we design an algorithm to identify the optimal slicing. We enumerate all different slowest execution time of slices. Given the constraints of maximum execution time, we employ a dynamic programming algorithm to restrict other slices to maintain the same, since the optimization of sequence S can derive from $S - s_n$. It yields the slicing scheme that attains the optimal overall pipeline latency.

3 EVALUATION

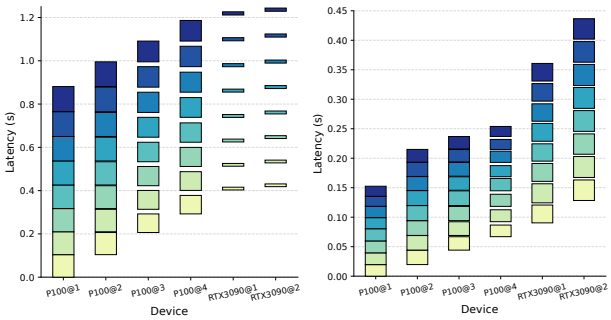
We established a computational cluster with two host machines to mimic a micro-enterprise hardware setup. The first machine contains four NVIDIA Tesla P100 GPUs (12 GB memory each), while the second is fitted with two NVIDIA RTX 3090 GPUs (24 GB memory each). With the 10,496 CUDA cores of RTX 3090 against 3,584 of the P100, a heterogeneous environment is created. Communication between hosts is via a wired network with the bandwidth of 1000 Mbps, and intra-host communication is via PCIE. We use Meta's LLaMA-7b for our experiments, which has 7 billion parameters and necessitates at least 14 GB of GPU memory during execution. The input sequence in experiments contains 2048 tokens.

Table 1 shows the latency and memory usage for local execution and PipeLLM. Given the P100's limited memory, LLaMA-7B is distributed evenly to two P100s. Employing four P100 pipeline significantly improves latency from 3.927s to 1.373s, achieving a speedup to 2.86×. Additionally, incorporating the superior RTX3090 further reduced latency. Concerning memory, LLM is assigned among machines according to their computing capabilities, thereby mitigating the memory burden per machine as the increased devices.

The effectiveness of the PipeLLM in leveraging computational resources and identifying the optimal slicing of sequences has been affirmed. Fig. 2a displays the result of equally distributing the LLM across devices, along with the evenly slicing of sequences. RTX3090 exhibits tiny execution time compared to P100, ascribed to LLM distribution failing to fully harness the device's capabilities, leading to idle time. Moreover, uniform slicing sequences lead to longer execution times for subsequent subsequences, causing a bottleneck in the pipeline's efficiency. Fig. 2b demonstrates that our method schedules the execution of subsequence feasibly, thus it significantly reduces the inference latency.

Table 1: Preliminary evaluation results

Cluster	Method	Latency (s)	Speedup	Memory Usage (MB)
P100	local	-	-	-
2 P100	local	3.927	-	8767
RTX3090	local	0.609	-	16995
4 P100	pipeline	1.373	2.86×	4561
4 P100 + 2 RTX3090	pipeline	0.432	9.09×	1715 | 6769



(a) (b)

Figure 2: The performance of the pipeline inference with or without PipeLLM. Distinct colored blocks represent the execution time of subsequences. Gaps between blocks denote the communication time.

4 CONCLUSION

PipeLLM effectively integrates computing resources, allowing users to accelerate LLM with token-level pipeline on heterogeneous devices without considering hardware differences. The *future work* will focus on maximizing the efficacy of algorithms to identify optimal solutions and developing a versatile framework adaptable to various LLMs.

5 ACKNOWLEDGEMENT

This work was supported in part by the National Key R&D Program of China 2020YFB1807800, in part by the National Natural Science Foundation of China under Grants (62101064, 62201072, 62171057, 62071067, 62001054), in part by the Ministry of Education and China Mobile Joint Fund (MCM20200202), Beijing University of Posts and Telecommunications-China Mobile Research Institute Joint Innovation Center.

REFERENCES

[1] Reza Yazdani Aminabadi, Samyam Rajbhandari, Ammar Ahmad Awan, Cheng Li, Du Li, Elton Zheng, Olatunji Ruwase, Shaden Smith, Minjia Zhang, Jeff Rasley, et al. 2022. DeepSpeed-inference: enabling efficient inference of transformer models

at unprecedented scale. In *SC22: International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE, 1–15.

[2] Jiawei Fei, Chen-Yu Ho, Atal N Sahu, Marco Canini, and Amedeo Sapio. 2021. Efficient sparse collective communication and its application to accelerate distributed deep learning. In *Proceedings of the 2021 ACM SIGCOMM 2021 Conference*. 676–691.

[3] Yanping Huang, Youlong Cheng, Ankur Bapna, Orhan Firat, Dehao Chen, Mia Chen, HyoukJoong Lee, Jiquan Ngiam, Quoc V Le, Yonghui Wu, et al. 2019. Gpipe: Efficient training of giant neural networks using pipeline parallelism. *Advances in neural information processing systems* 32 (2019).

[4] Zhihao Jia, Matei Zaharia, and Alex Aiken. 2019. Beyond Data and Model Parallelism for Deep Neural Networks. *Proceedings of Machine Learning and Systems* 1 (2019), 1–13.

[5] Zhuohan Li, Siyuan Zhuang, Shiyuan Guo, Danyang Zhuo, Hao Zhang, Dawn Song, and Ion Stoica. 2021. Terapipe: Token-level pipeline parallelism for training large-scale language models. In *International Conference on Machine Learning*. PMLR, 6543–6552.

[6] Deepak Narayanan, Mohammad Shoeybi, Jared Casper, Patrick LeGresley, Mostofa Patwary, Vijay Korthikanti, Dmitri Vainbrand, Prethvi Kashinkunti, Julie Bernauer, Bryan Catanzaro, et al. 2021. Efficient large-scale language model training on gpu clusters using megatron-lm. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. 1–15.

[7] Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. 2019. Megatron-lm: Training multi-billion parameter language models using model parallelism. *arXiv preprint arXiv:1909.08053* (2019).