HelixTrain: Enhancing Long-Context LLM Training via 3D Dynamic Parallelism

Shiju Wang
College of AI
Tsinghua University
wangsj25@mails.tsinghua.edu.cn

Yiyang Liu
College of AI
Tsinghua University
yy-liu25@mails.tsinghua.edu.cn

Jiawang Zhao
Institute for Interdisciplinary Information Sciences
Tsinghua University
zhao-jw25@mails.tsinghua.edu.cnn

1 Background

1.1 Distributed LLM Training

Distributed training is widely used for accelerating LLM training, and there exist several parallel training strategies.

Data Parallelism. Data Parallelism (DP) assigns training input of sequences to different devices in a DP group, where the model states (parameters, gradients, and optimizer states) are duplicated and gradients need to be reduced to ensure mathematical consistency. To alleviate the overhead of model states, sharded data parallelism (SDP) techniques (e.g., DeepSpeed-ZeRO Rajbhandari et al. [2020] and PyTorch FSDP Zhao et al. [2023]) further partition model states across devices. Accordingly, gather and scatter communications are introduced to obtain complete parameters required for LLM's execution and reduce gradients, respectively. The gather and scatter communications in SDP can be overlapped with computation.

Tensor Parallelism. Tensor Parallelism (TP) Narayanan et al. [2021b] partitions weight matrices of linear projection along rows or columns in FFN and attention layers. An all-reduce communication is performed to synchronize the results and maintain mathematical consistency. Megatron-SP Korthikanti et al. [2022] replaces the all-reduce communication with a pair of reduce-scatter and all-gather communications, reducing activation memory footprint without increasing communication cost.

Sequence Parallelism. Sequence Parallelism (SP) partitions sequences into multiple slices and distributes them among devices in an SP group, introducing communication for self-attention operations. SP is typically categorized into two variants: Ulysses-style SP and Ring-style SP. DeepSpeed-Ulysses Jacobs et al. [2023] proposes Ulysses-Style SP, which first performs three all-to-all communications on query, key, and value to gather the complete sequence and distribute along the head dimension, after which a head parallel attention is performed. An additional all-to-all communication is finally required to redistribute the attention output along the sequence dimension. In contrast, Ring-Style SP Liu et al. [2023a], Li et al. [2023], Brandon et al. [2023], Li et al. [2024] exemplified by Context Parallelism (CP) performs self-attention through multiple steps, where the keys and values of different slices are exchanged via p2p communication.

Pipeline Parallelism. Pipeline Parallelism (PP) horizontally partitions a model into several parts (stages) that execute sequentially, requiring transmission of activation between two neighboring parts. This transmission introduces negligible communication overhead as it occurs only once. To enhance device occupancy, PP partitions training inputs into micro-batches, which categorizes PP

Preprint.

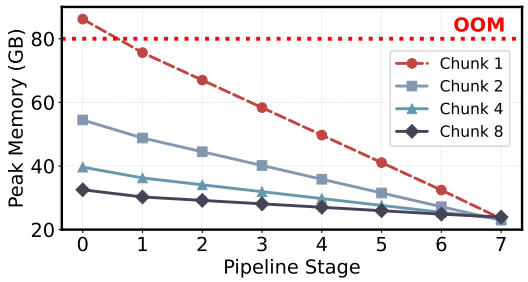

Figure 1: Profiled peak memory footprint per stage of Seq1F1B (DAPPLE when Chunk is 1) to train GPT-7B with a context of 16K tokens on 8 A800-80G GPUs, where "Chunk" refers to number of chunks to split a sequence into. Statistics are simulated for DAPPLE due to the limited memory capacity.(adapted from Wang et al. [2025a])

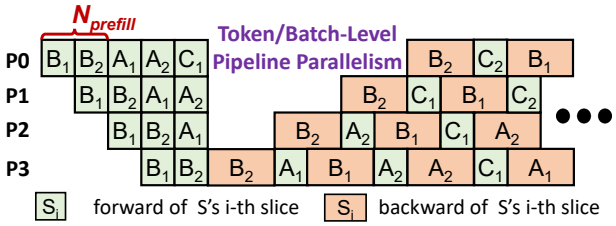

Figure 2: Illustration of Token/Batch-Level Pipeline Parallelism. For Batch-Level PP, $N_{prefill}$ is equal to number of slices that a sequence is split into (1 for batch-level PP). (adapted from Wang et al. [2025a])

into two variants: 1) *batch-level* PP, like DAPPLE Fan et al. [2021] that divides input samples, and 2) *token-level* PP exemplified by Seq1F1B Sun et al. [2024a] and TeraPipe Li et al. [2021] that further splits a sequence into slices. For the sake of training stability, we focus on synchronous PP with periodic pipeline flushes. Various batch-level pipeline schedules Fan et al. [2021], Li and Hoefler [2021], pyt [2021], Narayanan et al. [2021a], Liu et al. [2023b] have been proposed. DAPPLE and Seq1F1B's schedule consisting of three distinct stages: *warmup*, *steady*, and *cooldown*. For the token-level PP's schedule, it's worth noting that *for each slice, the forward pass must be scheduled after its preceding slices, while the backward pass must be scheduled after its subsequent slices*, as illustrated in Fig. 2. This is because the query of a token accesses only preceding tokens' keys and values in the forward pass, thus gradients of key and value of a token rely on those of subsequent tokens in the backward pass. Token-level PP employs a finer-granularity micro-batch of slices, exhibiting a lower memory footprint compared to batch-level PP, as shown in Fig. 1.

1.2 Sequence Packing

Techniques such as padding or packing are used to deal with sequences of different lengths. As shown in Fig. 3, padding pads or truncates sequences to the same length, introducing unnecessary

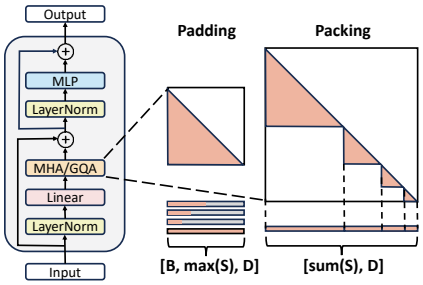

Figure 3: Illustration of sequence packing and padding's difference in attention mask and activation arrangement.(adapted from Wang et al. [2025a])

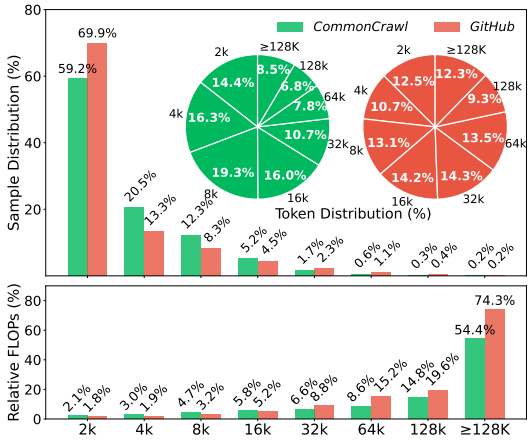

Figure 4: Statistics of sequences grouped by length intervals. The upper subgraph presents the sample and token distribution, while the bottom denotes the computation FLOPS distribution.(adapted from Wang et al. [2025a])

computation overhead. In contrast, sequence packing Krell et al. [2021], which concatenates multiple input sequences into a single sequence and adjusts attention masks to prevent cross-attention between unrelated tokens from different sequences, emerges as a more modern and efficient alternative.

1.3 Gradient Checkpointing

Gradient checkpointing is a widely adopted technique in LLM training that trades computation for activation memory footprint reduction. Specifically, intermediate activations are freed after the forward pass, but recomputed in the backward pass for gradient computation if checkpointing is applied.

1.4 Gradient Accumulation

To train LLM at a large batch size with limited memory capacity, gradient accumulation is proposed, which updates parameters once using the reduced gradients accumulated from multiple micro-batches and yields the same optimization trajectory.

1.5 Skewness Distribution of Sequence Length

Sequence length distribution of the real-world dataset exhibits skewness, as shown in Fig. 4. Take *GitHub* for example, 91.5% of the sequences have no more than 8K tokens, with only 0.6% of the sequences whose lengths exceed 64K or more. However, these 0.6% sequences contribute to 21.6% of the total tokens and a substantial amount of computation FLOPs. Moreover, recently released LLMs Yang et al. [2025], Dubey et al. [2024], Li et al. [2025] adopt a mixture of long and short sequences for context extension. Llama3 Dubey et al. [2024] indicates that mixing 0.1% of long-context data with short-context data optimizes performance across both short-context and long-context benchmarks. *The workload heterogeneity reveals an optimization opportunity for workload-aware dynamic pipeline schedule*. Previous works Jiang et al. [2024], Ge et al. [2025], Zhao et al. [2025] only study dynamic pipeline schedules for batch-level PP, restricting the applicability in long-context training scenarios with limited resources.

2 Problem Definition

Long context training is crucial for LLM's context extension. Its substantial memory footprint and computational overhead necessitates distributed training to accelerate the process, where it's of significant importance to reduce the communication overhead and memory footprint. As mentioned in § 1.5, workload variety of sequence length is introduced, resulting in suboptimal performance of static parallel training strategy. Moreover, we prefer to apply optimal checkpointing configuration

that minimizes computational overhead while adheres to hardware memory capacity. To this end, four questions emerge for efficient long-context training on varied-length corpus:

- What's the best parallel strategy given a training cluster and input training data?
- If the optimal parallel strategy is determined, what's the best checkpointing configuration?
- It's there any interference between checkpointing and parallelism?
- Due to the dynamic nature of workload, how to efficiently switch parallelism strategies?

3 Proposed Method

Motivation. The limitations of existing works FlexSP Wang et al. [2025b] and InfiniPipe Wang et al. [2025a] motivate us to employ the strengths of each work and tackle the problems encountered when combining them together. Specifically, FlexSP adopts *Flexible Sequence Parallelism* (FSP) for efficient communication of varied-length sequences. However, it ignores the optimization opportunity of pipeline parallelism, harming the scalability to larger-scale training clusters. On the other hand, InfiniPipe proposes *Elastic Pipeline Parallelism* (EPP), where dynamic pipeline schedule and gradient checkpointing configuration for workload variety is co-optimized. However, it considers only SP and PP, resulting in suboptimal performance deployed on larger-scale cluster. To be concrete, the enlarged SP degree introduces costly inter-node communication while the enlarged PP degree requires more micro-batches to reduce pipeline bubbles. To this end, we aim to further improve the training scalability by adopting the strengths of FlexSP's *Flexible Sequence Parallelism* and InfiniPipe's *Elastic Pipeline Parallelism*.

Solution: *3D Dynamic Parallelism*. We partition the parallelism strategy horizontally and vertically, where EPP is adopted for vertical parallelism and FSP is chosen for horizontal parallelism. Generally speaking, three parallelisms: *data parallelism*, *sequence parallelism* and *pipeline parallelism* are considered. Moreover, a 3D hierarchical processing of training input is employed. Besides InfiniPipe's approach that splits long sequences into short slices and packs short sequences together, HelixTrain further designates different micro-batches to sequence parallel groups in a data parallel manner. The workload balance of these micro-batches should be ensured to reduce idle time of devices. To our best knowledge, *3D Dynamic Parallelism* exhibits the most flexible and data-centric parallelism pattern.

4 Related Work

Long Context Training

Many sequence parallelism patterns for long context training have been proposed Liu et al. [2023a], Brandon et al. [2023], Li et al. [2023]. Other works Wang et al. [2025b], Ge et al. [2024] observe the skewness distribution of sequence length and aim to address workload heterogeneity. FlexSP Wang et al. [2025b] observes the skewness distribution of sequence length and proposes flexible sequence parallelism to improve communication efficiency.

Pipeline Parallelism Optimization

Recent works like AdaPipe Sun et al. [2024b], Mario Liu et al. [2025], and SPPO Chen et al. [2025] have explored checkpointing and offloading optimizations with PP. However, these works assume homogeneous workloads, but we focus on heterogeneous workloads with varied-length input. ChunkFlow Yuan et al. [2025] introduces hierarchical processing for varied-length sequences. ByteScale Ge et al. [2025] and WLB-LLM Wang et al. [2025c] optimize workload balance of batch-level PP for heterogeneous workload. InfiniPipe Wang et al. [2025a] co-optimizes dynamic pipeline schedule and gradient checkpointing policy, effectively employing pipeline parallelism in long-context training scenario.

References

Pytorch gpipe. `https://pytorch.org/docs/stable/pipeline.html`, 2021.

William Brandon, Aniruddha Nrusimha, Kevin Qian, Zachary Ankner, Tian Jin, Zhiye Song, and Jonathan Ragan-Kelley. Striped attention: Faster ring attention for causal transformers. *CoRR*, abs/2311.09431, 2023. doi: 10.48550/ARXIV.2311.09431. URL https://doi.org/10.48550/arXiv.2311.09431.

Qiaoling Chen, Shenggui Li, Wei Gao, Peng Sun, Yonggang Wen, and Tianwei Zhang. Sppo: Efficient long-sequence llm training via adaptive sequence pipeline parallel offloading. *arXiv preprint arXiv:2503.10377*, 2025.

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

Shiqing Fan, Yi Rong, Chen Meng, et al. DAPPLE: a pipelined data parallel approach for training large models. In *PPoPP*, pages 431–445. ACM, 2021.

Hao Ge, Fangcheng Fu, Haoyang Li, Xuanyu Wang, Sheng Lin, Yujie Wang, Xiaonan Nie, Hailin Zhang, Xupeng Miao, and Bin Cui. Enabling parallelism hot switching for efficient training of large language models. In *Proceedings of the ACM SIGOPS 30th Symposium on Operating Systems Principles*, pages 178–194, 2024.

Hao Ge, Junda Feng, Qi Huang, Fangcheng Fu, Xiaonan Nie, Lei Zuo, Haibin Lin, Bin Cui, and Xin Liu. Bytescale: Efficient scaling of llm training with a 2048k context length on more than 12,000 gpus. *arXiv preprint arXiv:2502.21231*, 2025.

Sam Ade Jacobs, Masahiro Tanaka, Chengming Zhang, Minjia Zhang, Shuaiwen Leon Song, Samyam Rajbhandari, and Yuxiong He. Deepspeed ulysses: System optimizations for enabling training of extreme long sequence transformer models. *CoRR*, abs/2309.14509, 2023. doi: 10.48550/ARXIV.2309.14509. URL https://doi.org/10.48550/arXiv.2309.14509.

Chenyu Jiang, Zhen Jia, Shuai Zheng, Yida Wang, and Chuan Wu. Dynapipe: Optimizing multi-task training through dynamic pipelines. In *Proceedings of the Nineteenth European Conference on Computer Systems*, pages 542–559, 2024.

Vijay Korthikanti, Jared Casper, Sangkug Lym, Lawrence McAfee, Michael Andersch, Mohammad Shoeybi, and Bryan Catanzaro. Reducing activation recomputation in large transformer models. *CoRR*, abs/2205.05198, 2022. doi: 10.48550/ARXIV.2205.05198. URL https://doi.org/10.48550/arXiv.2205.05198.

Mario Michael Krell, Matej Kosec, Sergio P Perez, and Andrew Fitzgibbon. Efficient sequence packing without cross-contamination: Accelerating large language models without impacting performance. *arXiv preprint arXiv:2107.02027*, 2021.

Aonian Li, Bangwei Gong, Bo Yang, Boji Shan, Chang Liu, Cheng Zhu, Chunhao Zhang, Congchao Guo, Da Chen, Dong Li, et al. Minimax-01: Scaling foundation models with lightning attention. *arXiv preprint arXiv:2501.08313*, 2025.

Dacheng Li, Rulin Shao, Anze Xie, Eric P. Xing, Joseph E. Gonzalez, Ion Stoica, Xuezhe Ma, and Hao Zhang. Lightseq: Sequence level parallelism for distributed training of long context transformers. *CoRR*, abs/2310.03294, 2023. doi: 10.48550/ARXIV.2310.03294. URL https://doi.org/10.48550/arXiv.2310.03294.

Dacheng Li, Rulin Shao, Anze Xie, Eric P Xing, Xuezhe Ma, Ion Stoica, Joseph E Gonzalez, and Hao Zhang. Distflashattn: Distributed memory-efficient attention for long-context llms training. In *First Conference on Language Modeling*, 2024.

Shigang Li and Torsten Hoefler. Chimera: efficiently training large-scale neural networks with bidirectional pipelines. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–14, 2021.

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

Hao Liu, Matei Zaharia, and Pieter Abbeel. Ring attention with blockwise transformers for near-infinite context. *CoRR*, abs/2310.01889, 2023a. doi: 10.48550/ARXIV.2310.01889. URL https://doi.org/10.48550/arXiv.2310.01889.

Weijian Liu, Mingzhen Li, Guangming Tan, and Weile Jia. Mario: Near zero-cost activation checkpointing in pipeline parallelism. In *Proceedings of the 30th ACM SIGPLAN Annual Symposium on Principles and Practice of Parallel Programming*, pages 197–211, 2025.

Ziming Liu, Shenggan Cheng, Haotian Zhou, and Yang You. Hanayo: Harnessing wave-like pipeline parallelism for enhanced large model training efficiency. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–13, 2023b.

Deepak Narayanan, Amar Phanishayee, Kaiyu Shi, Xie Chen, and Matei Zaharia. Memory-efficient pipeline-parallel dnn training. In *International Conference on Machine Learning*, pages 7937–7947. PMLR, 2021a.

Deepak Narayanan, Mohammad Shoeybi, Jared Casper, et al. Efficient large-scale language model training on GPU clusters using megatron-lm. In *SC*, pages 58:1–58:15. ACM, 2021b.

Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. Zero: memory optimizations toward training trillion parameter models. In *SC*. IEEE/ACM, 2020.

Ao Sun, Weilin Zhao, Xu Han, Cheng Yang, Xinrong Zhang, Zhiyuan Liu, Chuan Shi, and Maosong Sun. Seq1f1b: Efficient sequence-level pipeline parallelism for large language model training. *arXiv preprint arXiv:2406.03488*, 2024a.

Zhenbo Sun, Huanqi Cao, Yuanwei Wang, Guanyu Feng, Shengqi Chen, Haojie Wang, and Wenguang Chen. Adapipe: Optimizing pipeline parallelism with adaptive recomputation and partitioning. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3*, pages 86–100, 2024b.

Shiju Wang, Yujie Wang, Ao Sun, Fangcheng Fu, Zijian Zhu, Bin Cui, Xu Han, and Kaisheng Ma. Data-centric elastic pipeline parallelism for efficient long-context llm training. *arXiv preprint arXiv:2509.21275*, 2025a.

Yujie Wang, Shiju Wang, Shenhan Zhu, Fangcheng Fu, Xinyi Liu, Xuefeng Xiao, Huixia Li, Jiashi Li, Faming Wu, and Bin Cui. Flexsp: Accelerating large language model training via flexible sequence parallelism. In *Proceedings of the 30th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*, pages 421–436, 2025b.

Zheng Wang, Anna Cai, Xinfeng Xie, Zaifeng Pan, Yue Guan, Weiwei Chu, Jie Wang, Shikai Li, Jianyu Huang, Chris Cai, et al. Wlb-llm: Workload-balanced 4d parallelism for large language model training. *arXiv preprint arXiv:2503.17924*, 2025c.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Xiulong Yuan, Hongtao Xu, Wenting Shen, Ang Wang, Xiafei Qiu, Jie Zhang, Yuqiong Liu, Bowen Yu, Junyang Lin, Mingzhen Li, et al. Efficient long context fine-tuning with chunk flow. *arXiv preprint arXiv:2503.02356*, 2025.

Hairui Zhao, Qi Tian, Hongliang Li, and Zizhong Chen. {FlexPipe}: Maximizing training efficiency for transformer-based models with {Variable-Length} inputs. In *2025 USENIX Annual Technical Conference (USENIX ATC 25)*, pages 143–159, 2025.

Yanli Zhao, Andrew Gu, Rohan Varma, Liang Luo, Chien-Chin Huang, Min Xu, Less Wright, Hamid Shojanazeri, Myle Ott, Sam Shleifer, Alban Desmaison, Can Balioglu, Pritam Damania, Bernard Nguyen, Geeta Chauhan, Yuchen Hao, Ajit Mathews, and Shen Li. Pytorch FSDP: experiences on scaling fully sharded data parallel. *Proc. VLDB Endow.*, 16(12):3848–3860, 2023. doi: 10.14778/3611540.3611569. URL https://www.vldb.org/pvldb/vol16/p3848-huang.pdf.