StarTrail: Concentric Ring Sequence Parallelism for Efficient Near-Infinite-Context Transformer Model Training

Ziming Liu*
National University of Singapore
liuziming@comp.nus.edu.sg

Shaoyu Wang*
University of Southern California
wangshao@usc.edu

Shenggan Cheng
National University of Singapore
shenggan@comp.nus.edu.sg

Zhongkai Zhao
National University of Singapore
zhongkai.zhao@u.nus.edu

Kai Wang
National University of Singapore
kai.wang@comp.nus.edu.sg

Xuanlei Zhao
National University of Singapore
xuanlei@comp.nus.edu.sg

James Demmel
University of California, Berkeley
demmel@berkeley.edu

Yang You
National University of Singapore
youy@comp.nus.edu.sg

Abstract

Training Transformer models on long sequences in a distributed setting poses significant challenges in terms of efficiency and scalability. Current methods are either constrained by the number of attention heads or excessive communication overheads. To address this problem, we propose **StarTrail**, a multi-dimensional concentric distributed training system for long sequences, fostering an efficient communication paradigm and providing additional tuning flexibility for communication arrangements. Specifically, StarTrail introduces an extra parallel dimension and divides the peer-to-peer communication into sub-rings to substantially reduce communication volume and avoid bandwidth bottlenecks. Through comprehensive experiments across diverse hardware environments and on both Natural Language Processing (NLP) and Computer Vision (CV) tasks, we demonstrate that our approach significantly surpasses state-of-the-art methods that support Long sequence lengths, achieving performance improvements of up to 77.12% on GPT-style models and up to 114.33% on DiT (Diffusion Transformer) models without affecting the computation results.

1 Introduction

Over the past decade, Transformer[38] models have made remarkable strides in diverse fields, including computer vision (CV) and natural language processing (NLP). As the technology has evolved, the ability to efficiently process long sequences with Transformer has emerged as a pivotal challenge. For instance, in text summarization, the ability to handle extensive sequences is vital, as the content to be summarized can range from lengthy chapters to entire books [17, 3]. Similarly, chat-based applications, such as ChatGPT [1], require the capacity to process extensive dialogue

*Equal Contribution.

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

histories to ensure conversational consistency. There are also applications in other fields like video generation[5, 30] and protein structure prediction[15, 7].

The long context in the above scenarios has introduced several challenges for model training and inference: 1) **Efficiency and Adaptability**. The challenge of efficiency predominantly lies in handling long sequences that require quadratic computations during attention, and in addressing the large amount of communication during distributed processing. 2) **Memory**. Besides the major obstacle of storing the model weight and optimizer states, the activation has also exceeded the capacity of a single GPU and risen as a new memory challenge due to the extreme sequence length. 3) **Scalability**. Current Transformer models usually require thousands of GPUs for pre-training, even with datasets of regular lengths. For longer sequences, ensuring an acceptable scaling speedup rate with both the sequence length and the number of GPUs increasing is even more critical to reducing time and economic costs.

Traditional parallelisms such as Data Parallelism[12, 37, 22, 41], Tensor Parallelism[37, 39, 40], and Pipeline Parallelism[13, 10, 23, 25] distribute the model, input batch, and the optimizer states, but can not directly address the large memory requirement of extremely long sequences as the sequence length dimension remains unchanged. To break through this obstacle, Sequence Parallelism has been introduced, splitting the input on the sequence length dimension. Mainstream Sequence Parallelism schemes can generally be classified into two categories, and are usually combined [11] to complement each other's drawbacks. Methods like DeepSpeed Ulysses[14], which are based on all-to-all communication, offer efficiency but require the splitting of attention heads. Consequently, these methods are limited in scalability and can not be scaled to more devices than the number of attention heads. On the other hand, peer-to-peer communication methods[24, 20], such as Ring Attention[24], do allow for near-infinite context lengths; however, they require the transmission of complete keys and values across all GPUs, leading to significantly high communication loads. In summary, there remains a deficiency in communication-efficient methods that are capable of supporting near-infinite context lengths. In this paper, we focus on solving the communication inefficiency of ring-style sequence parallelism.

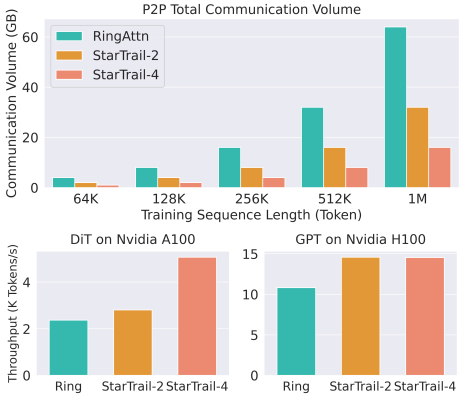


Figure 1: StarTrail-2 and StarTrail-4 theoretically save around 50% and 75% of total P2P communication volume for various sequence length and achieves up to 2x speedup in end-to-end training

.

To solve these challenges, we introduce StarTrail, a novel near-infinite-context Transformer training system with concentric multi-ring sequence parallelism that incorporates an additional parallel dimension into the existing ring-style communication. Specifically, instead of including all GPUs in a single parallel group as done in Ring Attention [24], StarTrail groups the GPUs into teams and divides the peer-to-peer communication within these teams. This approach fosters an efficient communication paradigm and provides extra tuning flexibility for communication arrangements. With very little additional memory cost, StarTrail parallelism significantly reduces the peer-to-peer communication volume, as shown in Figure 1. Compared to previous works, StarTrail is not limited in supported sequence length by attention heads like DeepSpeed Ulysses[14] and Megatron Sequence Parallelism[18], and also shows better communication efficiency and scalability than Ring Attention[24]. We perform experiments on mainstream Transformer models, including GPT-style[33] and DiT-style[31], conducting performance and scaling tests across various computing clusters. Experiment results indicate that our StarTrail system outperforms Ring Attention by up to 77.12% on the GPT model and up to 114.33% on the DiT model, showcasing its efficiency and scalability.

2 Background and Related Works

2.1 Long Sequence Training and Sequence Parallelism

The key mechanism behind these Transformer-based models is attention[38], which captures the text feature by calculating the attention score between every two single tokens. However, the sequence length can reach hundreds of thousands, when dealing with multi-round chatting, or high-resolution long video generation. It then becomes necessary to distribute the sequence across multiple GPUs. This distribution helps to reduce both the memory and computation demands on any single device. This strategy is also known as sequence parallelism. Presently, Sequence parallelism can be divided into two main categories: attention-head-sharding-based and peer-to-peer-communication-based. The former involves distributing the attention heads of multi-head attention across multiple GPUs, whereas the latter resembles a distributed version of FlashAttention, relying on peer-to-peer communication to transfer keys, values, and intermediate statistics.

2.1.1 Ring-peer-to-peer-communication-based

The primary method in peer-to-peer-communication-based strategies is Ring Attention[24], which is also the main baseline of this work. Introduced in 2023, Ring Attention[24] innovatively partitions the sequence dimension and utilizes a ring-style peer-to-peer (P2P) communication pattern to transfer Keys and Values across all GPUs. Each GPU receives the key and value matrices from the preceding rank, updates the local attention score, and then forwards them to the next rank, as is shown in Figure 2. This method employs an online-softmax and updates attention scores incrementally, allowing the computation of attention scores without retaining the full sequence length. Thus, it potentially supports infinite context, provided sufficient computing resources are available. However, the requirement for the same number of rounds of P2P communication as the number of GPUs renders this approach less efficient in environments with high-latency communication.

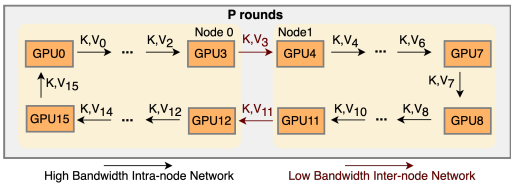
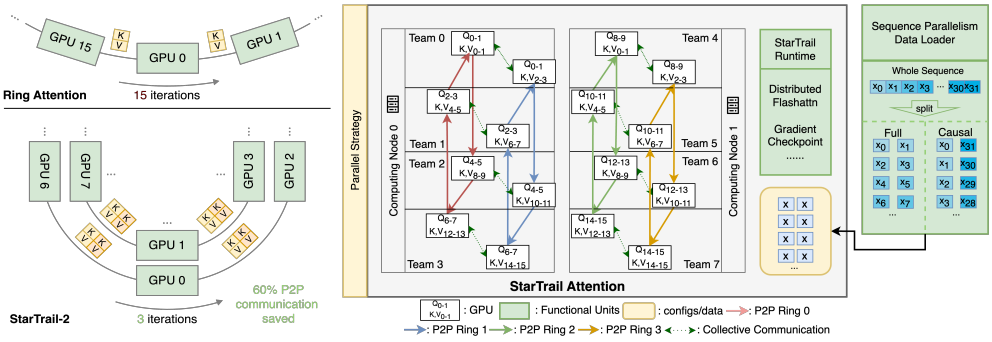


Figure 2: An example of Ring Attention Computation on 16 GPUs in two nodes. The Communication is largely limited by the inter-node bottleneck.



(a) StarTrail-2 reduces 60% P2P communication volume on 16 GPUs compared with ring attention.

(b) StarTrail divides one ring into four concentric sub-rings, with every two connected with collective communication.

Figure 3: An overview of the StarTrail Training System

2.1.2 Attention-Head-Sharding-Based

There are two representative methods, DeepSpeed-Ulysses and Megatron Sequence Parallelism in this category. DeepSpeed-Ulysses[14] transitions from sequence parallelism to a method akin to tensor parallelism with two all-to-all communication. It divides the query, key, and value matrices across the attention heads, thereby preserving the original attention computation structure. Megatron Sequence Parallelism[18] focuses on minimizing memory usage and reducing the necessity for

activation recomputation rather than efficiency. The two methods both rely on the number of attention heads to split the sequence, thus limited in scalability, especially when employing techniques like grouped-query attention (GQA) [2] or multi-query attention (MQA) [36]. As these two sequence parallelism methods are **orthogonal** to the ring-based method, they are usually combined with ring attention to enable longer sequences.

In this paper, we focus on optimizing ring-style sequence parallelism, noting that integrating it with DeepSpeed-Ulysses Parallelism does not interfere with the underlying ring process.

3 StarTrail Training System

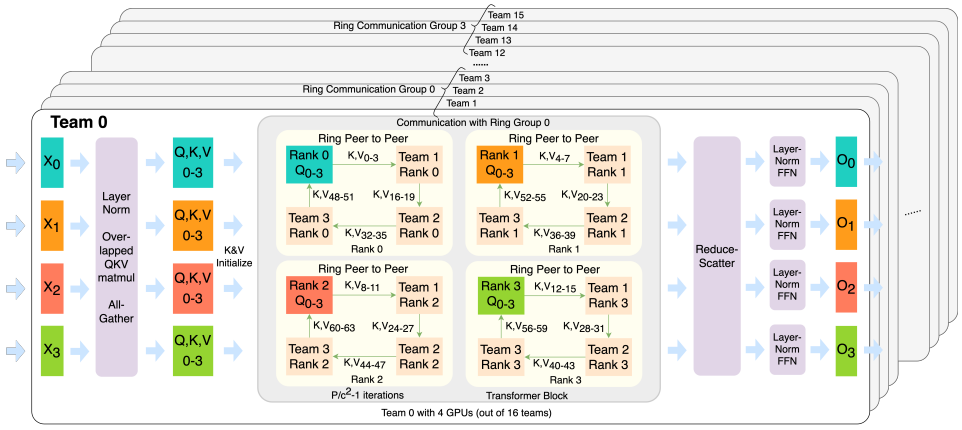


Figure 4: An example of StarTrail Attention on 64 GPUs. Each team member forms a sub-ring with members with the same local rank from other teams in the same ring communication group, reducing the communication volume of each team member by 75%.

3.1 Motivation

Through observation, we identify two main drawbacks of Ring Attention. First, the communication overhead is exceedingly high because every GPU in the system must send and receive keys and values for nearly the entire sequence length before completing the attention computation. Second, variations in bandwidth between and within computing nodes can cause communication bottlenecks. As illustrated in Figure 2, the bandwidths between GPUs 3 and 4 and between GPUs 11 and 12 are lower than those between other GPUs in the ring. Despite this, the system is forced to operate in a complete circle, which can result in unnecessary idle times for other GPUs. To solve these drawbacks, we develop the StarTrail training system, which we will detail in the following section.

3.2 StarTrail Attention

As discussed in the previous section, a major limitation of Ring Attention is the extensive amount of peer-to-peer (P2P) communication required, which becomes problematic in environments with weak connections between computing nodes. We enhance the ring sequence parallelism by introducing an additional dimension. The fundamental idea of StarTrail is akin to a divide-and-conquer strategy. During attention, each token must compute its attention score with every other token in the sequence. While Ring Attention passes keys and values along a ring of P GPUs over $P-1$ iterations, our approach introduces the concept of a **team**. In this setting, each team member interacts only with a designated portion of the overall sequence, and the results are later aggregated using collective

Figure 5: Meanings of the symbols that are used in this paper

P	The number of GPUs
C	The parallel size of StarTrail (team size)
H	The hidden dimension size of the Transformer blocks
N	The total number of tokens within the whole sequence
B	The training batch size
W	The communication bandwidth between GPUs
L	The communication latency between GPUs

communication. Thus, StarTrail can be devided into three phases: **preprocessing**, **ring-phase**, and **postprocessing**.

For **preprocessing**, we duplicate the queries within a team using an all-gather operation. This ensures that when a GPU receives new keys and values, it can compute the attention scores for the entire team's queries. Similarly, gathering the keys and values allows us to reduce the number of P2P communication iterations by transmitting longer sequences per iteration. Following the preprocessing, we enter the **ring-style communication phase**. With the number of GPUs in one team being C, CN/P tokens are exchanged in each iteration, and each GPU is responsible for computing N/C tokens. This leads to a number of iterations of $\frac{N/C}{CN/P} = \frac{P}{C^2}$, within a smaller ring, which we refer to as a **subring**. For convenience, we group $\frac{P}{C^2}$ adjacent teams into a **team group** for subring communication, where GPUs sharing the same local team rank form the subring. An initial P2P communication step is executed to ensure that each team group has access to the complete set of keys and values for the sequence (details are provided in the Appendix). After completing the subring iterations, each GPU holds $1/C$ of the overall computation result for its team. With the help of online softmax, we then apply a simple reduce-scatter to combine these results while eliminating the duplicate tokens, which we refer to as the **postprocessing**. Throughout the attention process, asynchronous communication is employed alongside the early launch of communication kernels to maximize the overlap of computation and communication tasks. Now we will delve into more details in the StarTrail training process.

3.2.1 Configurations of StarTrail Parallelism

In the StarTrail system, GPUs are grouped into *Teams* to coordinate computation and communication tasks more efficiently. StarTrail introduces an additional parameter, C, which determines the replication factor of the input and, consequently, the number of GPUs within each team. The range of C is from 1 to $\sqrt{P}$. When C equals one, the algorithm falls back to Ring Attention. When C equals $\sqrt{P}$, the algorithm becomes a completely collective-communication-based one with no rings. When $1 < C < \sqrt{P}$, it becomes a structure with multiple rings looping concurrently.

Algorithm 1 StarTrail Attention Block (Forward)

Require: Input sequence **x**, Linear Function **query, key,** and **value**, attention parallelism size **c**, global rank **r**, global size **gs**, team process group **pg**, init send/recv target $\mathbf{r}_{\text{send}}$ and $\mathbf{r}_{\text{recv}}$

compute the gathered $\mathbf{q}_{\text{team}}, \mathbf{k}_{\text{team}}, \mathbf{v}_{\text{team}}$ = AllGather_QKVmatmul(**query, key, value, x, pg**)
launch the asynchronous send and receive request $\mathbf{req}_{\text{send}}$ and $\mathbf{req}_{\text{recv}}$, sending $\mathbf{k}_{\text{team}}, \mathbf{v}_{\text{team}}$ to $\mathbf{r}_{\text{send}}$ and receiving $\mathbf{k}_{\text{next}}, \mathbf{v}_{\text{next}}$ from $\mathbf{r}_{\text{recv}}$
get the ring P2P target $\mathbf{r}_{\text{next}}$ and $\mathbf{r}_{\text{last}}$ with get_P2P_ranks(**r**, **gs**, **c**)
initialize attention score **O**, extra statistics **lse** to zero. // lse stands for log-sum-exp
for $1 \leq i \leq$world_size$/c^2$ **do**
    wait for $\mathbf{req}_{\text{send}}$ and $\mathbf{req}_{\text{recv}}$
    $\mathbf{k}_{\text{current}} = \mathbf{k}_{\text{next}}$, $\mathbf{v}_{\text{current}} = \mathbf{v}_{\text{next}}$
    launch $\mathbf{req}_{\text{send}}$ to send $\mathbf{k}_{\text{current}}$ and $\mathbf{v}_{\text{current}}$ to $\mathbf{r}_{\text{next}}$, launch $\mathbf{req}_{\text{recv}}$ to receive $\mathbf{k}_{\text{next}}$ and $\mathbf{v}_{\text{next}}$ from $\mathbf{r}_{\text{last}}$
    calculate **lse, O** = forward_iteration(**lse, O**, $\mathbf{q}_{\text{team}}, \mathbf{k}_{\text{current}}, \mathbf{v}_{\text{current}}$)
end for
compute $\mathbf{O}_{\text{final}}$ =ReduceScatter_combine(**lse, O, pg**)
return $\mathbf{O}_{\text{final}}$

Forward Propagation. In Figure 4, we have an example of one team of four GPUs out of all the 64 GPUs performing StarTrail-style attention. Each training iteration begins with the dataloader splitting the entire input sequence of length N into N/P sub-sequences, which are then loaded onto each GPU. As previously mentioned, the next step involves computing the queries, keys, and values. These are computed separately via matrix multiplication, followed immediately by the launch of the all-gather kernel, which gathers the above QKVs within the team, allowing for the overlap of up to two-thirds of the communication with computation.

Once this phase is complete, each GPU within the team possesses the same Q, K, and V, each of a length of $\frac{CN}{P}$. To distribute the communication and computation tasks among the team members,

we divide the original workload based on four specific ranks assigned to each GPU. These ranks determine each GPU's partners and position within the P2P ring, as is shown in Figure 6.

Following the setup, the Keys and Values are dispatched to their designated locations within the cluster to establish the initial sub-ring, setting the stage for the multi-ring iteration phase of StarTrail attention. Given that each sub-sequence is $\frac{CN}{P}$ long and each GPU is tasked with computing the attention score for $\frac{1}{C}$ of the whole sequence, it results in $\frac{N/C}{CN/P} - 1 = P/C^2 - 1$ rounds of communication. This implies that there are P/C^2 GPUs in one ring.

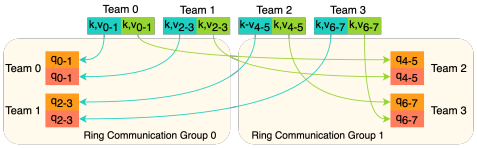


Figure 6: An example of ring initialization process of 8GPUs and 4 sub-rings in StarTrail.

The iteration process involves storing the log-sum-exp (lse) and intermediate output O, which are updated step by step. Queries are retained locally, while Keys and Values circulate through the ring via P2P communication. After completing the iterations, each team member accumulates the attention scores for the entire team's sub-sequence of Queries with $1/C$ of the Keys and Values from the full sequence.

A simple reduce-scatter operation is then employed to amalgamate the intermediate results and distribute them among the team members. Each GPU ultimately contains the final attention score for its portion of the sequence over the entire sequence.

Backward Propagation. The major distinction between backward and forward propagation is the inability to calculate queries independently during the backward phase. Unlike forward propagation, the backward phase requires the complete set of keys and values to calculate the gradient for queries, and vice versa. To manage this, we have structured the gradient calculation into two loops: the key & value outer loop and the query inner loop. In the outer loop, gradients for keys and values are tracked and maintained fixed on the corresponding GPUs within the sub-rings; these gradients do not transfer between GPUs. The inner loop, however, handles the gradients for queries, which start initialized as zero and are circulated along the sub-rings together with the Queries themselves. During each iteration, the approach mirrors the backward computation method used in FlashAttention[8], where the updated gradient of the current query shard is passed to the next GPU in the ring, while the gradients for keys and Values are retained for subsequent query shards.

3.2.2 Theoretical Analysis

During the analysis, we will employ a case study using the StarTrail system with an attention parallel size of C = 4 on a llama-30B model, which consists of 64 layers. For this model, referred to as model M, the batch size = B is set to 1, the sequence length = N to 65536, the hidden dimension = H to 6656, and the number of GPUs = P to 64. Additionally, the computation will utilize bfloat16 precision.

Communication Analysis. Let's analyze the communication overhead within one forward Transformer block on a single GPU. For Ring Attention, the communication is primarily due to the ring P2P loop. As the total number of iterations done is $P - 1$, the total communication overhead can be calculated as:

$$(P-1)(\frac{2BNH}{PW} + L) = \frac{2BNH(P-1)}{WP} + (P-1)L \tag{1}$$

and this overhead can be partially overlapped with the attention computation.

For StarTrail, the communication overhead comes from both collective and P2P. The collective overhead for all-gather and reduce-scatter is:

$$\frac{4BNH(C-1)}{PW} \tag{2}$$

while the P2P communication can be similarly computed as:

$$(\frac{P}{C^2} - 1)(\frac{2CBNH}{PW} + L) = \frac{(P-C^2)2BNH}{CPW} + (\frac{P}{C^2} - 1)L \tag{3}$$

The advantages of StarTrail over Ring Attention during the ring-P2P phase are evident in three main aspects: 1) **Reduced Communication and Latency**: Ring Attention requires C times more communication than StarTrail, significantly increasing the bandwidth requirement across the entire cluster. For the llama 30B model M, the total communication volume of ring P2P communication and collective communication volume for Ring Attention and StarTrail can be computed as 1.625 GB and 0.152 GB(collective) + 0.406GB (P2P) = 0.558GB. Furthermore, while Ring Attention necessitates $P-1$ iterations per attention block, StarTrail only requires $\frac{P}{C^2}-1$, reducing the latency overhead by around C^2. 2) **Localized Communication**: In scenarios like those depicted in Figure 3, StarTrail's ring P2P communication can be confined within the same computing node, where bandwidth is typically much higher than between computing nodes. Conversely, Ring Attention demands inter-node communication during every iteration, which can be less efficient. 3) **Enhanced Overlap of Communication and Computation**: During each iteration, the communication volume of StarTrail is C times higher than that of Ring Attention, while the computational volume during attention is approximately C^2 times greater. This higher computation-to-communication ratio makes it easier for StarTrail to overlap P2P communication with computation, enhancing overall efficiency.

Memory Analysis. In this section, we estimate the theoretical peak memory requirements necessary to store the model weights, activations, and optimizer states. Our implementation utilizes the Adam Optimizer [16], bfloat16 precision, and Zero-2 optimization [34]. We name the memory cost for the model and optimizer as M_{m+o}. As for the activation, we refer to the size of one single activation of a sub-sequence on one GPU as

$$A = \frac{B \times N \times H}{P} \tag{4}$$

As we use the checkpointing scheme from [20], a model of Y layers needs to save $Y+1$ activations as checkpoints. Now we calculate the approximate peak memory after Q, K, and V are already calculated and before the attention computation at the last layer of the whole model. For Ring Attention and StarTrail, the peak memories are:

$$PM_{Ring} = M_{m+o} + (Y+4)A \tag{5}$$

$$PM_{Star} = M_{m+o} + (Y+3C+1)A \tag{6}$$

, where C is the StarTrail attention dimension. And for the example model M, the peak memory would be $M_{m+o}+68A$ and $M_{m+o}+77A$, and the extra memory cost compared with Ring Attention is a lot less than 13.2%, while the P2P communication volume is reduced by about 75%. In a word, the extra memory cost is acceptable as a tradeoff for the communication reduction.

4 Evaluation

Table 1: Cluster and Model Configurations. All GPUs are connected with NVLink with computing nodes.

GPU	dev. × node	Mem. (GB)	inter-node bandwidth	Model	#Heads	#Layers	Dim.
H100	8×8	80	8*400Gbps InfiniBand	GPT 3B	12	16	4096
A100	16×2	40	100Gbps Ethernet	GPT 7B	32	32	4096
A100	8×4	40	100Gbps Ethernet	DiT 1B	24	24	1536
A100	4×8	40	100Gbps Ethernet				

The computational resources we use in the experiments include a local Nvidia H100 cluster with eight nodes and three Nvidia A100 clusters, as listed in table 1. We utilize two model types of total three settings, as listed in table 1. For the DiT(Diffusion Transformer) model, we use similar configurations as those in Stable Diffusion 3[9]. We utilize the backbone Diffusion Transformer only, without other components like the text and image encoders. During training, both models use bfloat16 precision and a batch size of 1 to accommodate longer input sequences.

In the evaluation section, we aim to answer three major questions: 1) How much improvement in throughput can StarTrail bring? Additionally, how adaptable is StarTrail to clusters with both good and poor inter-node connections? 2) Is the additional memory cost incurred by StarTrail acceptable considering the throughput improvement it offers? 3) How does StarTrail perform in scenarios of weak and strong scaling? Specifically, does it outperform Ring Attention when scaled to handle longer inputs?

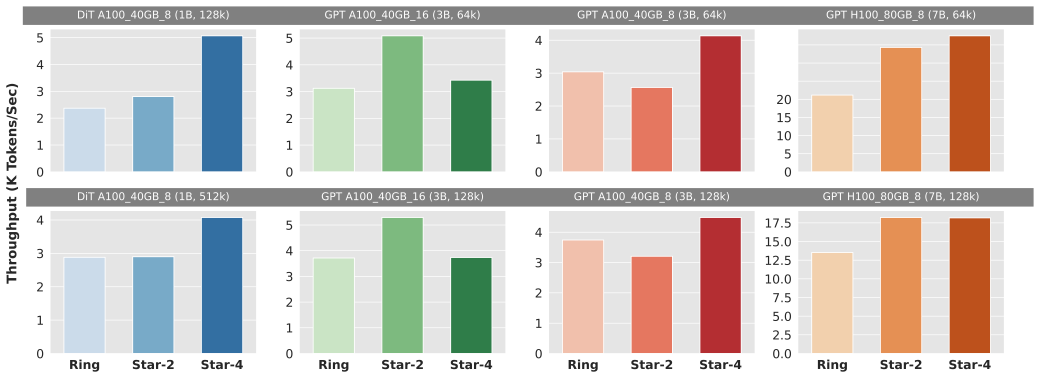


Figure 7: Throughput evaluation of Ring Attention and StarTrail on 32 GPUs from three different clusters. We place the performance of StarTrail with both C=2 and C=4 in the figure. The configurations are marked in the titles of the sub-figures. For instance, A100_40GB_8(1B, 512K) represents that the experiment is on machines with 8 Nvidia A100 40GB GPUs in each node, the model used has one billion parameters, and the sequence length is 512k.

4.1 Throughput and Adaptability

Our first experiment aims to assess the performance of StarTrail and Ring Attention across different clusters with varying environments, testing the adaptability of both methods. There are several factors influencing the efficiency of ring-style attention computation:

Theoretical Computation-Communication Volume Ratio: Primarily determined by the sequence length used during training. Attention computation exhibits a computational complexity of $O(N^2 \cdot H)$, whereas P2P communication complexity is $O(N \cdot H)$. Thus, the model configuration does not impact this ratio; only the sequence length does. A larger N increases the computation-communication ratio, facilitating easier overlap of communication with computation.

Compute Capability and Connectivity of GPUs: The computing overhead, given a specific volume, affects the computation-communication overhead ratio. Higher compute capabilities make overlapping more challenging. We utilize two sets of GPUs in this evaluation: Nvidia A100 40GB and Nvidia H100 80GB, with the latter offering significantly higher theoretical tflops on bf16 computations. Connectivity is considered in two parts: intra-node and inter-node. Our clusters are equipped with NVLink, ensuring robust intra-node communication. For inter-node communication, our H100 nodes leverage InfiniBand with eight adapters per node for superior inter-node bandwidth, whereas the Google Cloud servers use Ethernet. The diversity in node configurations (8-GPU and 16-GPU nodes) allows us to assess adaptability across different topologies. This evaluation not only highlights the inherent differences between the schemes but also tests their flexibility in various hardware settings.

The results of our evaluation are illustrated in Figure 7. We measure throughput in thousands of tokens per second. To better demonstrate how to select the optimal configuration of StarTrail under each condition, we included two configurations, Star-2 and Star-4, in the figure. We omit configurations with lower performance for clarity. As indicated in the figure, in all six settings, at least one configuration of StarTrail achieves higher throughput than Ring Attention, with 2.114x, 1.414x, 1.629x, 1.425x, 1.360x, 1.199x, 1.771x, and 1.346x the throughput of Ring Attention. This advantage is primarily due to the additional parallel dimension that StarTrail introduces. Unlike Ring Attention, which requires inter-node P2P communication in each iteration, StarTrail's P2P communication is mostly confined intra-node, except for initial data transfers. This experiment clearly demonstrates StarTrail's superior performance across various environments. Another observation is that the optimal configuration for StarTrail may vary depending on the environment, reflecting differences in the computation-communication ratio and the trade-offs between collective and P2P communication.

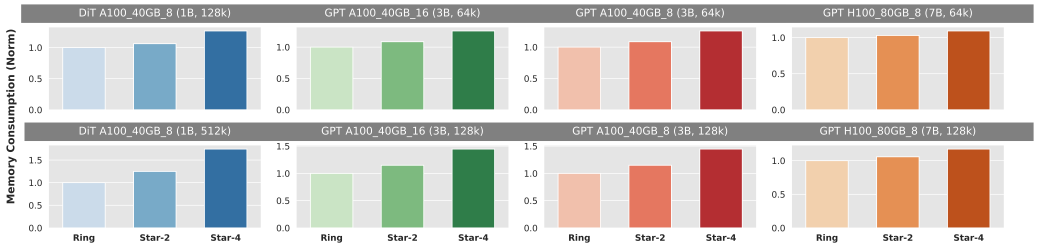
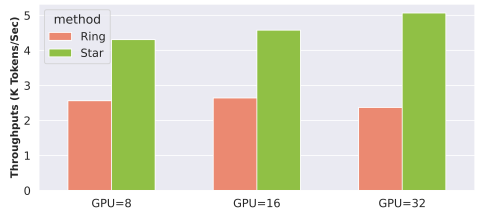


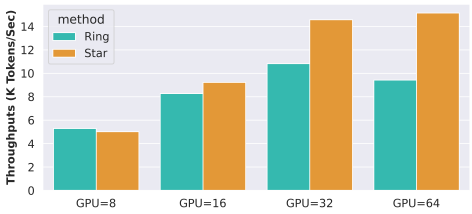
Figure 8: The normalized relative memory cost of different configurations of StarTrail compared with Ring Attention on different clusters.

4.2 Memory Consumption

The memory results, displayed in Figure 8, reveal that for the configurations yielding the highest throughput, StarTrail consumes between 7.9% and 30.79% more GPU memory than RingAttention, while achieving 1.199x to 2.114x throughput. Moreover, in scenarios involving larger models, the relative increase in memory consumption due to QKV duplication diminishes. This reduction is explained by equation 6 This phenomenon is further evidenced by the fact that the extra memory ratio for experiments with the 7B model is significantly smaller than that for the 3B and 1B model. Considering the substantial throughput gains provided by StarTrail, this tradeoff between memory usage and efficiency is deemed acceptable.



(a) DiT Strong Scaling on Nvidia A100 40GB GPUs. All configurations include inter-node communication.



(b) GPT Strong Scaling on Nvidia H100 80GB GPUs.

Figure 9: Strong scaling experiments with fixed sequence length of 128K.

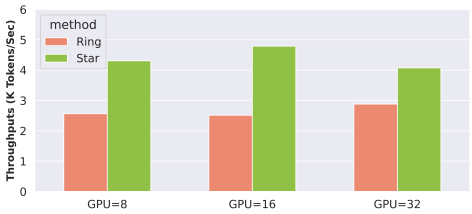
4.3 Strong and Weak Scaling

In the scaling tests we carry out experiments for both strong and weak scaling. For strong scaling, we fix the sequence length to 128K while increasing the number of GPUs from 8 to 64 for the GPT model and from 8 to 32 for the DiT model. For weak scaling, we scale the sequence length from 128k to 512k for the DiT model and from 64k to 512k for the GPT model proportionally increasing the number of GPUs from 8 to 32. As is depicted in Figure 9 and 10, StarTrail shows obvious advantage over Ring Attention as we increase the number of GPUs. The results for strong scaling can also explained by the computation-communication ratio. When scaled to more GPUs, the local sequence length on each GPU becomes smaller, and as explained the previous sections, makes it harder to overlap the P2P communication with attention computation. The overall scaling performance is limited by the nature of ring-style communication, but we still consider our improvement over Ring Attention meaningful due to the necessity of using Ring-style Parallelisms during training.

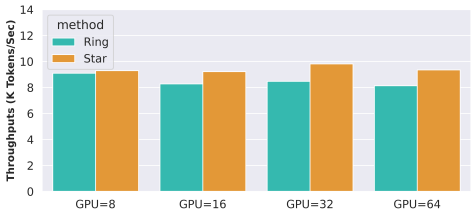
In summary, StarTrail shows better scalability in both strong and weak scaling experiments, making it a better choice for large-scale Transformer model training.

5 Conclusion

StarTrail represents an advanced near-infinite-context Transformer model training system, featuring a communication-optimized concentric ring sequence parallelism scheme. Through experiments, we



(a) DiT Weak scaling on Nvidia A100 40GB GPUs of sequence length from 128K to 512K. All configurations include inter-node communication.



(b) GPT Weak scaling on Nvidia H100 80GB GPUs of sequence length from 64K to 512K.

Figure 10: Weak scaling Experiments

demonstrate that our system not only achieves high efficiency across various training environments but also excels under both strong and weak scaling conditions for both CV and NLP models. Current limitations of StarTrail include that although orthogonal, we can still further improve the co-design of StarTrail and hybrid parallelism in future works. In an era increasingly demanding longer contexts for both NLP and CV, StarTrail is poised to make significant contributions to the industry and inspire innovative research in academia.

6 acknowledgement

Yang You's research group is being sponsored by NUS startup grant (Presidential Young Professorship), Singapore MOE Tier-1 grant, ByteDance grant, NUS ARTIC grant, Apple grant, Alibaba grant, Google Research and Google grant for TPU usage.

References

[1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, et al. Gpt-4 technical report, 2023.

[2] Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebrón, and Sumit Sanghai. Gqa: Training generalized multi-query transformer models from multi-head checkpoints, 2023.

[3] Iz Beltagy, Matthew E. Peters, and Arman Cohan. Longformer: The long-document transformer, 2020.

[4] Zhengda Bian, Hongxin Liu, Boxiang Wang, Haichen Huang, Yongbin Li, Chuanrui Wang, Fan Cui, and Yang You. Colossal-ai: A unified deep learning system for large-scale parallel training. *arXiv preprint arXiv:2110.14883*, 2021.

[5] Tim Brooks, Bill Peebles, Connor Holmes, Will DePue, Yufei Guo, Li Jing, David Schnurr, Joe Taylor, Troy Luhman, Eric Luhman, Clarence Ng, Ricky Wang, and Aditya Ramesh, Feb 2024.

[6] Chang Chen, Xiuhong Li, Qianchao Zhu, Jiangfei Duan, Peng Sun, Xingcheng Zhang, and Chao Yang. Centauri: Enabling efficient scheduling for communication-computation overlap in large model training via communication partitioning. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3*, ASPLOS '24, page 178–191, New York, NY, USA, 2024. Association for Computing Machinery.

[7] Shenggan Cheng, Xuanlei Zhao, Guangyang Lu, Jiarui Fang, Tian Zheng, Ruidong Wu, Xiwen Zhang, Jian Peng, and Yang You. Fastfold: Optimizing alphafold training and inference on gpu clusters. In *Proceedings of the 29th ACM SIGPLAN Annual Symposium on Principles and Practice of Parallel Programming*, pages 417–430, 2024.

[8] Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness, 2022.

[9] Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, Dustin Podell, Tim Dockhorn, Zion English, Kyle Lacey, Alex Goodwin, Yannik Marek, and Robin Rombach. Scaling rectified flow transformers for high-resolution image synthesis, 2024.

[10] Shiqing Fan, Yi Rong, Chen Meng, Zongyan Cao, Siyu Wang, Zhen Zheng, Chuan Wu, Guoping Long, Jun Yang, Lixue Xia, Lansong Diao, Xiaoyong Liu, and Wei Lin. Dapple: A pipelined data parallel approach for training large models, 2020.

[11] Jiarui Fang and Shangchun Zhao. Usp: A unified sequence parallelism approach for long context generative ai, 2024.

[12] W Daniel Hillis and Guy L Steele Jr. Data parallel algorithms. *Communications of the ACM*, 29(12):1170–1183, 1986.

[13] Yanping Huang, Youlong Cheng, Ankur Bapna, Orhan Firat, Mia Xu Chen, Dehao Chen, HyoukJoong Lee, Jiquan Ngiam, Quoc V. Le, Yonghui Wu, and Zhifeng Chen. Gpipe: Efficient training of giant neural networks using pipeline parallelism, 2018.

[14] 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, 2023.

[15] John M. Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Zídek, Anna Potapenko, Alex Bridgland, Clemens Meyer, Simon A A Kohl, Andy Ballard, Andrew Cowie, Bernardino Romera-Paredes, Stanislav Nikolov, Rishub Jain, Jonas Adler, Trevor Back, Stig Petersen, David A. Reiman, Ellen Clancy, Michal Zielinski, Martin Steinegger, Michalina Pacholska, Tamas Berghammer, Sebastian Bodenstein, David Silver, Oriol Vinyals, Andrew W. Senior, Koray Kavukcuoglu, Pushmeet Kohli, and Demis Hassabis. Highly accurate protein structure prediction with alphafold. *Nature*, 596:583 – 589, 2021.

[16] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

[17] Huan Yee Koh, Jiaxin Ju, Ming Liu, and Shirui Pan. An empirical survey on long document summarization: Datasets, models, and metrics. *ACM Computing Surveys*, 55:1 – 35, 2022.

[18] Vijay Korthikanti, Jared Casper, Sangkug Lym, Lawrence McAfee, Michael Andersch, Mohammad Shoeybi, and Bryan Catanzaro. Reducing activation recomputation in large transformer models, 2022.

[19] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention, 2023.

[20] 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, 2024.

[21] Shenggui Li, Fuzhao Xue, Yongbin Li, and Yang You. Sequence parallelism: Long sequence training from system perspective. In *Annual Meeting of the Association for Computational Linguistics*, 2021.

[22] Shigang Li, Tal Ben-Nun, Salvatore Di Girolamo, Dan Alistarh, and Torsten Hoefler. Taming unbalanced training workloads in deep learning with partial collective operations. In *Proceedings of the 25th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, pages 45–61, 2020.

[23] 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.

[24] Hao Liu, Matei Zaharia, and Pieter Abbeel. Ring attention with blockwise transformers for near-infinite context, 2023.

[25] 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*, SC '23, New York, NY, USA, 2023. Association for Computing Machinery.

[26] Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, and Hao Wu. Mixed precision training, 2018.

[27] Deepak Narayanan, Mohammad Shoeybi, Jared Casper, Patrick LeGresley, Mostofa Patwary, Vijay Anand Korthikanti, Dmitri Vainbrand, Prethvi Kashinkunti, Julie Bernauer, Bryan Catanzaro, Amar Phanishayee, and Matei Zaharia. Efficient large-scale language model training on gpu clusters using megatron-lm, 2021.

[28] NVIDIA. Nvidia collective communications library, 2020.

[29] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library, 2019.

[30] William Peebles and Saining Xie. Scalable diffusion models with transformers, 2023.

[31] William Peebles and Saining Xie. Scalable diffusion models with transformers, 2023.

[32] Markus N. Rabe and Charles Staats. Self-attention does not need $o(n^2)$ memory, 2022.

[33] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2018.

[34] Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. Zero: Memory optimizations toward training trillion parameter models, 2020.

[35] Jay Shah, Ganesh Bikshandi, Ying Zhang, Vijay Thakkar, Pradeep Ramani, and Tri Dao. Flashattention-3: Fast and accurate attention with asynchrony and low-precision, 2024.

[36] Noam Shazeer. Fast transformer decoding: One write-head is all you need, 2019.

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

[38] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023.

[39] Boxiang Wang, Qifan Xu, Zhengda Bian, and Yang You. Tesseract: Parallelize the tensor parallelism efficiently. In *Proceedings of the 51st International Conference on Parallel Processing*, pages 1–11, 2022.

[40] Qifan Xu, Shenggui Li, Chaoyu Gong, and Yang You. An efficient 2d method for training super-large deep learning models. *arXiv preprint arXiv:2104.05343*, 2021.

[41] Yang You, Zhao Zhang, Cho-Jui Hsieh, James Demmel, and Kurt Keutzer. Imagenet training in minutes. In *Proceedings of the 47th International Conference on Parallel Processing*, pages 1–10, 2018.

[42] Zilin Zhu et al. Ring flash attention, 2024.

NeurIPS Paper Checklist

The checklist is designed to encourage best practices for responsible machine learning research, addressing issues of reproducibility, transparency, research ethics, and societal impact. Do not remove the checklist: **The papers not including the checklist will be desk rejected.** The checklist should follow the references and follow the (optional) supplemental material. The checklist does NOT count towards the page limit.

Please read the checklist guidelines carefully for information on how to answer these questions. For each question in the checklist:

- You should answer [Yes] , [No] , or [NA] .
- [NA] means either that the question is Not Applicable for that particular paper or the relevant information is Not Available.
- Please provide a short (1–2 sentence) justification right after your answer (even for NA).

The checklist answers are an integral part of your paper submission. They are visible to the reviewers, area chairs, senior area chairs, and ethics reviewers. You will be asked to also include it (after eventual revisions) with the final version of your paper, and its final version will be published with the paper.

The reviewers of your paper will be asked to use the checklist as one of the factors in their evaluation. While "[Yes] " is generally preferable to "[No] ", it is perfectly acceptable to answer "[No] " provided a proper justification is given (e.g., "error bars are not reported because it would be too computationally expensive" or "we were unable to find the license for the dataset we used"). In general, answering "[No] " or "[NA] " is not grounds for rejection. While the questions are phrased in a binary way, we acknowledge that the true answer is often more nuanced, so please just use your best judgment and write a justification to elaborate. All supporting evidence can appear either in the main paper or the supplemental material, provided in appendix. If you answer [Yes] to a question, in the justification please point to the section(s) where related material for the question can be found.

IMPORTANT, please:

- **Delete this instruction block, but keep the section heading "NeurIPS Paper Checklist"**,
- **Keep the checklist subsection headings, questions/answers and guidelines below.**
- **Do not modify the questions and only use the provided macros for your answers**.

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We have clearly claimed that we improved the efficiency of ring-style sequence parallelism and provide analysis and experiments in the following sections.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

Justification: As mentioned in the conclusion, although orthogonal to other parallelism, we have not consider the co-design with other parallelism for better performances.

Guidelines:

- The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

   Justification: We do not have any theoretical result.

   Guidelines:

   - The answer NA means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We have provided the open-source models and GPU configurations we used in the experiments. The algorithm of the parallelism is also provided in this paper.

   Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [No]

   Justification: Our code is not currently available publicly, but we will prepare for open-source if accepted.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.

- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: We have provided all the details in the experiment sections.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: Our experiment results are measured by measuring a large number of training iterations and taking the average. It is not suitable for our case to prove statistical significance.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
   - If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: All details are mentioned in the experiment section.

   Guidelines:

- The answer NA means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We reviewed the NeurIPS Code of Ethics and this research conforms with the guidelines.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [NA]

    Justification: This work is simply improving the training efficiency and has no societal impact.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: We do not introduce any new model or dataset.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: We have credited the assets properly.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [NA]

Justification: Our work does not release new assets.

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA]

Justification: Our research does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [NA]

Justification: Our research does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

Answer: [NA]

Justification: Our key method is modifying the parallelism technique and is not related with LLM itself.

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.

A Technical Appendices and Supplementary Material

A.1 Sequence Parallelism Dataloader

The causal masks present a challenge due to the unbalanced computational load across GPUs: subsequences at the beginning of a sequence require significantly more computation than those at the end. To address this imbalance and achieve load equilibrium among GPUs, we modify the ZigZag scheme introduced by [42], illustrated in Figure 11. The figure illustrates the simplest case of zigzag load-balancing. Notably, the effectiveness of this strategy improves as the number of GPUs increases. This improvement correlates with the expanding difference in computation volume between the first and the last token, which escalates as the sequence length extends. This approach ensures that the total workload on each GPU is balanced, eliminating the need for additional communication mechanisms like those employed in DistFlashAttention[20].

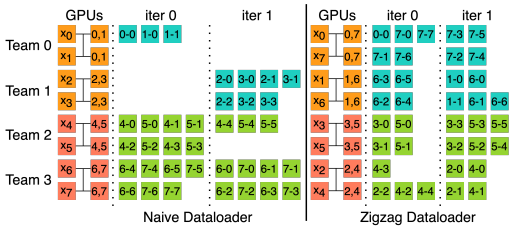


Figure 11: A comparison between naive and zigzag dataloader for 8 GPUs with attention parallel dimension of 2. The corresponding initialization can be found in Figure 6 with the same configuration. The improvement of efficiency from load-balancing increases with the number of GPUs.

A.2 Details in the training process

Apart from the attention process described in Algorithm 1, we would like do provide a few other details to be more comprehensive. First, before the concentric ring attention, we need to initialize the the Keys and Values and determine each GPU's position within the rings.

Initially, for the setup stage, it is essential to establish the sub-rings by rearranging the activation positions. Specifically, during forward propagation, the queries do not require rearrangement; however, the keys and values must be transmitted to their corresponding positions in the ring prior to commencing the loop. As illustrated in Figure 6 and algorithn, this initialization ensures that each team member holds a different shard of keys and values. Moreover, it guarantees that no two teams within the same ring possess identical keys and values. Initially, for the setup stage, it is essential to establish the sub-rings by rearranging the activation positions. Specifically, during forward propagation, the queries do not require rearrangement; however, the keys and values must be transmitted to their corresponding positions in the ring prior to commencing the loop. As illustrated in Figure 6 and algorithm, this initialization ensures that each team member holds a different shard of keys and values. Moreover, it guarantees that no two teams within the same ring possess identical keys and values.

Algorithm 2 get_init_send()

Require: inter-team rank $\mathbf{r}_t$, intra-team rank $\mathbf{r}_a$, inter-team dimension $\mathbf{d}_t$, intra-team dimension $\mathbf{d}_a$

team group size = $\mathbf{d}_t$ / $\mathbf{d}_a$
target team group rank = $\mathbf{r}_a$
target team = target team group rank * team group size + $\mathbf{r}_t$ // $\mathbf{d}_a$
target device intra-team rank = $\mathbf{r}_t$ % $\mathbf{d}_a$
target global rank = target team * $\mathbf{d}_a$ + target device intra-team rank
return target global rank

After the initialization of activations, we can set up the rings by providing the GPUs their last and next GPU within their rings, as is described in Algorithm 3.

A.3 StarTrail Runtime

StarTrail is written in PyTorch[29] and uses the PyTorch torch.autograd.function and NCCL[28] backend for forward and backward implementation. StarTrail also employs multiple techniques during runtime to improve its overall training efficiency.

Ingetrate Flash Attention. The StarTrail attention mechanism involves multiple iterations that loop over Keys and Values, with each iteration still using traditional self-attention with corresponding Query, Key, and Value (QKV). This approach enables StarTrail to incorporate flash attention effectively, extending its capability by preserving intermediate states across iterations. Additionally,

Algorithm 3 get_P2P_config()

Require: inter-team rank $\mathbf{r}_t$, intra-team rank $\mathbf{r}_a$, inter-team dimension $\mathbf{d}_t$, intra-team dimension $\mathbf{d}_a$

team group size = $\mathbf{d}_t$ / $\mathbf{d}_a$

self team group rank = $\frac{r_t}{team\ group\ size}$

next team in group = $(r_t + 1)\%$ team group size + team group size $\times$self team group rank

last team in group = $(r_t - 1)\%$ team group size + team group size $\times$self team group rank

next device global rank = r_a+ next team in group $\times d_a$

last device global rank = r_a+ last team in group $\times d_a$

return next device global rank, last device global rank

Table 2: Supported Sequence Length of Ring Attention and StarTrail on one Nvidia A100 80GB GPU.

Supported Seq Len on one 80GB A100 GPU (K Tokens)			
Model Size	Length	Ring Attention	StarTrail
3B	128	✓	✓
	256	✓	✓
	512	✓	✓
7B	128	✓	✓
	256	✓	✓
	512	✗	✗
13B	128	✓	✓
	256	✗	✗
	512	✗	✗

StarTrail enhances the efficiency of the forward process with the help of torch JIT to fuse kernels aside from flash attention.

Overlap communication with computing. In StarTrail attention, P2P communication and self-attention computing are interleaved across iterations, each incurring considerable time. To mitigate this, StarTrail employs a double buffering technique to asynchronously execute communication and computing kernels, effectively overlapping these processes and enhancing GPU utilization.

Save recomputation with checkpoints. StarTrail adopts the checkpointing strategy introduced by DistFlashAttn[20], placing checkpoints at the end of the self-attention phase rather than the FFN of each transformer layer. This checkpoint placement effectively obviates the need to recompute the self-attention forward process during the backward pass, avoiding redundant attention computation.

B Additional Experiment

To comprehensively evaluate the memory consumption of StarTrail and Ring Attention, we compared the maximum supported sequence lengths of StarTrail with those reported in the Ring Attention paper [24]. As shown in Table 2, although StarTrail requires slightly more memory, it still supports sequence lengths commonly used in training tasks.

B.1 Discussion

B.2 Larger Batch Sizes and Models

We utilized small batches and model sizes in our experiments because these choices do not affect the underlying improvements in communication efficiency and computation-to-communication ratio that StarTrail provides. For larger batch sizes, both communication and computation scale proportionally, leaving the overlapping ability unchanged. Similarly, while larger models involve more layers or

larger hidden sizes, the key attention computations and corresponding ratios remain unaffected. Hence, our conclusions naturally extend to scenarios with larger batches and models.

B.2.1 FlashAttention3 and Hopper GPUs

In addition to the original FlashAttention [8] used in our experiments, FlashAttention3 [35] has been introduced, specifically designed for Hopper and newer Nvidia GPUs. For FP16 precision, which is utilized in this paper, FlashAttention3 achieves a 1.5-2.0x speedup on Hopper GPUs. As discussed in Section 3, reducing attention computation overhead results in more P2P communication not being overlapped, further emphasizing the need to reduce communication volume. With the increasing adoption of Hopper GPUs, the significance of the StarTrail system will also grow.

B.2.2 StarTrail and Other Parallelisms

Model Parallelism As is well known, **tensor parallelism** shards activations by attention heads during attention computation, making it easily combinable with StarTrail with minimal effort. However, when combined with tensor parallelism, the need for attention heads can limit the scalability of head-based sequence parallel methods like DeepSpeed-Ulysses. **Pipeline parallelism**, on the other hand, divides the model across layers without altering the computation patterns within Transformer blocks, making StarTrail orthogonal to it.

Other Sequence Parallelism StarTrail is orthogonal with other attention-head-sharding-based sequence parallelism approaches, such as DeepSpeed-Ulysses [14]. While DeepSpeed-Ulysses distributes attention heads across different devices, StarTrail can independently partition activations along the sequence length dimension. In future work, we can explore combining StarTrail with DeepSpeed-Ulysses to expand the communication scheduling space, harnessing the scalability of StarTrail alongside the efficiency of DeepSpeed-Ulysses.

In summary, StarTrail can be seamlessly integrated with other parallel training techniques, enabling the creation of a hybrid distributed training system.

B.3 Other Related Works

Attention Optimization. Traditional full attention mechanisms necessitate $O(n^2)$ memory for storing the outputs of QK^T, leading to significant computational and memory demands. To address these challenges within the GPU, several approaches have been devised to reduce both memory and computational requirements. Memory-efficient attention[32] introduces a straightforward algorithm that requires only $O(1)$ memory relative to the sequence length, with an extension for self-attention that needs only $O(\log n)$ memory. FlashAttention further minimizes I/O overhead and enhances overall efficiency. Additionally, optimization methods specifically tailored for inference, such as PagedAttention[19], are also being developed to improve the efficiency of attention computations. In this work, we utilize FlashAttention within each iteration to reduce the computation overhead.

Long-Sequence Training Techniques. Sequence Parallelism[21] was initially introduced to enhance the efficiency of parallel long-sequence training. Ring Attention[24] improved communication efficiency through memory-efficient methods[32], supporting near-infinite sequence lengths. DeepSpeed-Ulysses[14] employs attention head splitting to achieve high efficiency, though it is constrained by the number of heads. Megatron Sequence Parallelism focuses on reducing memory costs during Tensor Parallelism, while DistFlashAttention[20] features a load-balance scheme and a novel gradient checkpoint method. Our work builds on these innovations, introducing a system that supports large-scale training with an efficient communication scheme.

Techniques for Distributed Model Training. Distributed model training encompasses two primary areas: 1) **Memory Management**: Various techniques aim to conserve GPU memory during distributed training, such as mixed precision training[26] and the ZeRO series[34]. In this work, we implement ZeRO-2 to manage optimizer states and gradients efficiently. 2) **Hybrid Parallelism**: Frameworks like Megatron[27] and Colossal AI[4] integrate multiple forms of parallelism. There are various existing Parallelism techniques like Pipeline Parallelism[13, 10, 23, 25] and Tensor Parallelism[37], which can be combined with StarTrail Parallelism to facilitate large-scale training. We are also considering the integration of additional frameworks such as [6] to enhance overlapping capabilities in future implementations.