A UNIFIED SEQUENCE PARALLELISM APPROACH FOR LONG CONTEXT GENERATIVE AI

Jiarui Fang
Tencent
jiaruifang@tencent.com

Shangchun Zhao
Tencent
doctzhao@tencent.com

ABSTRACT

Sequence parallelism (SP), which divides the sequence dimension of input tensors across multiple computational devices, is becoming key to unlocking the long-context capabilities of generative AI models. This paper investigates the state-of-the-art SP approaches, i.e. DeepSpeed-Ulysses and Ring-Attention, and proposes a unified SP approach, which is more robust to transformer model architectures and network hardware topology. This paper compares the communication and memory cost of SP and existing parallelism, including data/tensor/zero/pipeline parallelism, and discusses the best practices for designing hybrid 4D parallelism involving SP. We achieved 86% MFU on two 8xA800 nodes using SP for sequence length 208K for the LLAMA3-8B model. Our code is publicly available on `https://github.com/feifeibear/long-context-attention`.

1 Introduction

The field of artificial intelligence is witnessing a trend as the context length in generative AI models grows ever longer. Claude has pioneered this trend in large language models (LLMs) by extending the sequence length to 100K tokens. Following closely in its footsteps, OpenAI's GPT-4 has expanded the context length to 128K tokens. The advent of multi-modality models is propelling this trend forward, with Gemini 1.5 Pro boasting a context length of a staggering 10 million tokens, and OpenAI's Sora, a Diffusion Model, accommodating at least 1 million visual tokens. These breakthroughs underscore the imperative for generative AI techniques to adeptly handle a larger context length.

Sequence Parallelism (SP), a technique that partitions input sequences, has emerged as a promising approach for the training or inference of longer sequences. Following an initial exploration period of two years, by the latter of 2023, two landmark works, DeepSpeed-Ulysses [1] and Ring-Attention [2], marked the maturation of the SP technique. DeepSpeed-Ulysses maintains constant communication volume when sequence length and compute devices are increased proportionally, while Ring-Attention hides P2P communication costs introduced by SP through overlapping computation and communication. However, challenges remain, such as the SP parallel degree of DeepSpeed-Ulysses is limited to less than the number of attention heads, and the computational efficiency of Ring-Attention degrading due to the subdivision of matrix multiplications. These limitations currently hinder the broader adoption of Sequence Parallelism in distributed Transformer computation.

In this paper, we delve deeper into the realm of SP. We begin by highlighting that Ulysses and Ring are not mutually exclusive approaches; they can be combined through a hybrid parallel strategy to mitigate their drawbacks. Then, we discussed the relationship between SP and data/tensor/zero/expert/pipeline parallelism. The most complex among these is the relationship between SP and tensor parallelism. Since tensor parallelism also has its specific sequence parallel optimizations to reduce activation memory cost [3]. For each parallelism approach, whether SP should replace it or is there some issue to using SP with it together, remains an open question. After addressing these questions, we have provided a list of best practices for building a 4D hybrid parallelism system.

The primary contributions of this paper include:

- We propose a unified sequence parallel method that integrates DeepSpeed-Ulysses and Ring-Attention, overcoming the shortcomings of both and demonstrating greater robustness to model architecture and network hardware.
- We systematically analyze the application of SP in conjunction with Tensor Parallelism, ZeRO, and Pipeline Parallelism as 4D parallelism, and provide a list of best practices to apply SP.

2 Sequence Parallelism Approaches

Before delving into the concept of SP, let's review the computational process of the standard Transformer Block. The Notion used in this paper is shown in Table 1, where $d = hc \times hs$.

Notation	Description
L	Sequence Length
d	Hidden Dimension
hc	Head Count
hs	Head Size
bs	Batch Size
N	Device Number

Table 1: Notation for Transformer Parameters

Given input sequences $Q, K, V \in \mathbb{R}^{L \times d}$, where L is the sequence length and d is the head dimension, we compute the matrix of outputs as follows:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d}}\right)V, \tag{1}$$

Each self-attention sub-layer is accompanied by a feedforward network (FFN), which is applied to each position separately and identically.

$$\text{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2. \tag{2}$$

Compared to Tensor Parallelism (TP) [4] and ZeRO [5], the research on SP for Transformer models has been relatively underdeveloped for a long time. The challenge lies in the characteristic of attention computation, where the sequence dimension serves as a common dimension in the V matrix multiplications, making it difficult to partition the tensors and distribute the computation across multiple nodes after slicing the sequence dimension.

Early attempts [6, 7, 8] at SP were not successful, often leading to redundant memory consumption [6] and inefficient communication pattern [8]. For long input sequences, the best practice is to adopt the *Sequence Parallelism* of Megatron-LM. This method optimizes the AllReduce operation of TP, reducing the memory cost of activations while maintaining the same communication overhead. As shown in Figure 1, the principle of Megatron-LM *Sequence Parallelism* is the similar to ZeRO-2 [5]. It replaces the AllReduce operation on replicated tensors (the left figure) into equivalent allgather and reduce-scatter operations on partitioned data (the right figure). Since an AllReduce operation is exactly a combination of Allgather and ReduceScatter, the communication cost remains the same. The size of the input and output tensors is reduced by a factor of $1/N$ across N computational devices. Because of the sequence dimension in input/output tensors is partitioned, it is named as *Sequence Parallelism*. However, this form of *Sequence Parallelism* cannot be used independently without tensor parallelism.

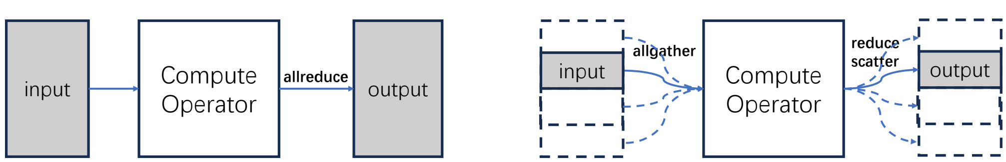

Figure 1: The Principle of Megatron-LM Sequence Parallelism.

The readiness of the standalone SP technology is marked by the publication of two milestone papers in November 2023. The DeepSpeed-Ulysses [1], named as **SP-Ulysses**, and Ring-Attention [2], named as **SP-Ring**, solving the longstanding memory and communication issues inherent to SP from two distinct perspectives. For both methodologies, each computational device is allocated a distinct segment of the Q (query), K (key), V (value), and O (output) tensors, which are segregated along the dimensions of the sequence. There is no redundancy in their storage across devices, which is a primary distinction from the early SP design [6].

SP-Ring can be viewed as a distributed version of FlashAttention [9]. As shown in the right part of Figure 2, SP-Ring employs a two-level nested loop that orchestrates the communication and computation in a blockwise fashion. When computing for the blocks of the tensor O segment, if the required tensor K and V blocks are not locally available, Peer-to-Peer (P2P) communication is utilized to fetch them from other devices. The communication can be organized in a Ring fashion, where each device simultaneously sends and receives K, V blocks, allowing the communication to overlap with computation.

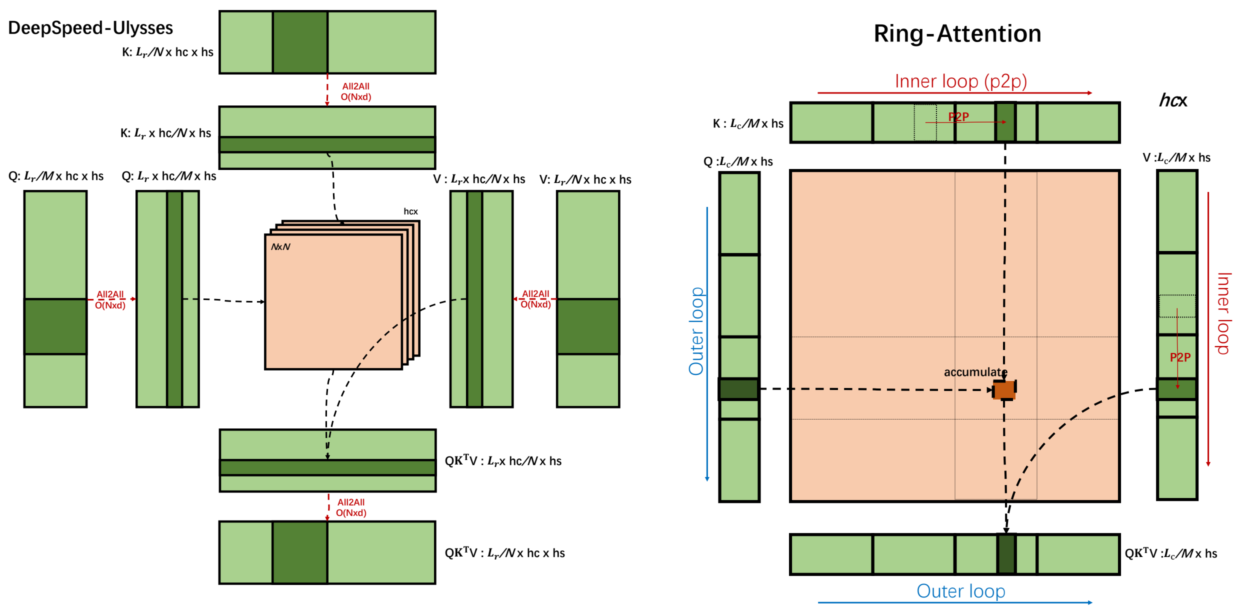

Figure 2: SP-Ulysses and SP-Ring.

SP-Ulysses leverages All2All communication for segments of the Q, K, V, and O tensors, as shown in the left part of Figure 2. After the All2All operation, the partitioning of these four tensors shifts from the sequence dimension L to the dimension of the attention number heads hc. Therefore, the computation of $softmax(QK^T)V$ for each attention head is maintained in its entirety, and can be implemented using the under the hood Attention operator library, like FlashAttention.

3 Unified Ulysses-Ring Sequence Parallelism

Currently, both SP-Ulysses and SP-Ring are facing certain issues that hinder their effectiveness in practical applications.

SP-Ulysses is sensitive to the number of attention heads. The parallelism degree [1] of DS-Ulysses cannot exceed the number of attention heads hc. Consequently, it is not suitable for GQA (Grouped Query Attention) [10] and MQA (Multi-Query Attention) [11] scenarios. For instance, Llama3-8B employs GQA with a KV head number of 8, which means that when using DS-Ulysses SP, the maximum SP degree is 8. However, if GQA is used and KV head number is 1, DS-Ulysses dose not function. In addition, since Tensor Parallelism also requires division across the hc dimension, SP-Ulysses, and TP are in conflict.

SP-Ring is inefficient in computation and communication. Ring-Attention segments the Q, K, V, O tensors into smaller blocks, which can lead to a decrease in computation efficiency of fused operator $Softmax(QK^T)V$. Even if the communication and computation fully overlap, the total execution time lags behind that of DS-Ulysses. When using a

[1]The parallelism degree is the number of devices that participate in parallel computation. In other words, it is the number of processes in a parallel process group.

causal mask has not been addressed, the DS-Ring has load unbalancing issue. DS-Ring does not impose any restrictions on hc.

SP-Ulysses and SP-Ring are currently considered alternative strategies for SP, with the choice of one precluding the other. This viewpoint was reinforced by the SP-Ring authors in their ICLR open review rebuttal[2]. Currently, Megatron-DeepSpeed utilizes SP-Ulysses, while Megatron-LM opts for SP-Ring for its SP implementation. However, we claim that rather than viewing them as rivals, they can work together as a unified SP approach.

As shown in Algorithm 1, SP-Ring and SP-Ulysses are organized in a hybrid parallel manner to work together in partitioning the sequence dimension. The SP process group is segmented into two orthogonal process group sets: a set of SP-Ring process groups and a set of SP-Ulysses process groups. For a more intuitive understanding, an SP process group can be viewed as a 2D mesh, SP-Ring operates across each column of the mesh, while SP-Ulysses runs across each row. For example, a process group containing 8 processes could viewed as 2×4, where the SP-Ulysses process group of size 2 and the SP-Ring process group of size 4. This is the same as how data parallelism and tensor parallelism process groups are partitioned.

The inputs to the **Unified_SP_Attention** include the segment of the $Q, K, andV$ tensors after being partitioned along the sequence dimension and the output is a tensor O shard. The size of a tensor segment is (bs, L/N, hs, hd). Note that when using MAQ, the shape of the hc for the K and V tensor segments differs from that of the Q tensor. During forward propagation, $scatter\_idx$ is set to 1 and $scatter\_idx$ is set to 2. The AllToAll4D merges the L dimension and partitions the hc dimension of the Q, K, and V tensors, transforming them into (hc/N, bs, L, d). They also partition the O tensor along the L dimension and merge the hc dimension. During the backward propagation, $scatter\_idx$ is set to 2, and $scatter\_idx$ is set to 1.

Algorithm 1 Unified Sequence Parallelism Attention Implementation

function UNIFIED_SP_ATTENTION($ulysses\_pg$, $ring\_pg$, Q, K, V, $scatter\_idx$, $gather\_idx$)
  $Q \leftarrow$ **AllToAll4D**$(Q, scatter\_idx, gather\_idx, group = ulysses\_pg)$
  $K \leftarrow$ **AllToAll4D**$(K, scatter\_idx, gather\_idx, group = ulysses\_pg)$
  $V \leftarrow$ **AllToAll4D**$(V, scatter\_idx, gather\_idx, group = ulysses\_pg)$
  $O \leftarrow$ **LoadBalance-RingAttention**$(Q, K, V, group = ring\_pg)$
  $O \leftarrow$ **AllToAll4D**$(ulysses\_pg, O, gather\_idx, scatter\_idx, group = ulysses\_pg)$
  return O
end function

The vanilla SP-Ring introduces load-unbalancing issues when applying a causal attention mask, as only the lower triangular matrix needs to be computed. As shown in the Figure 3 If the sequence dimension is divided evenly, the computational tasks are not evenly distributed among the devices. As shown in the left side of the figure, on 4 GPUs the computation load of GPU3 is nearly 7 times that of GPU0.

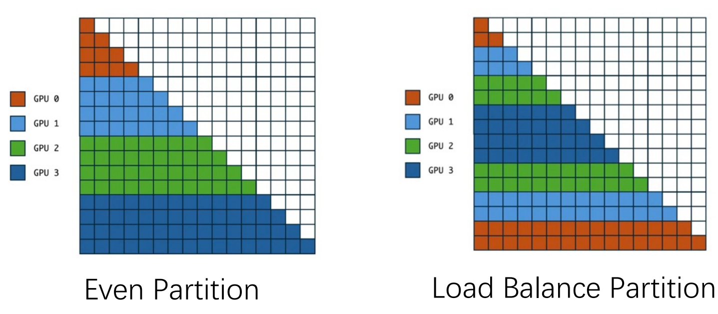

Figure 3: Loading Balancing Partition for SP-Ring.

The solution to the load-unbalancing issue is to reorder the input sequence tokens along the sequence dimension, as depicted on the right side of Figure 3. In the figure, the input sequence consists of 16 tokens. Under even partition, GPU0 processes tokens 0-3, while GPU3 handles tokens 12-15. After the reorder operation for load balance partitioning, GPU0 now processes tokens 0, 1, 14, 15, and GPU3 processes tokens 5,6,11,12. The workloads handled by each GPU are perfectly balanced, which is a superior solution to SP-Ring load balance [12].

[2] https://openreview.net/forum?id=WsRHpHH4s0&noteId=HIY0tae4Gz

The aforementioned load balancing method is quite straightforward and similar implementation has already been applied in Megatron-LM. We present its application for Unified SP method. The Algorithm 2 extracts and reorders the input sequence from the global input sequence in hybrid Ulysses and Ring parallelism. For the positional encoding only involves element-wise operations, i.e. RoPE [13], the same reordering operation also needs to be applied to the positional encoding parameters of the model. Since the operation is applied to the model's input token sequence, which is an integer vector of length $bs * L$, the additional overhead for load balancing is negligible.

Algorithm 2 Prepare Load Balance Sequence Segment from The Global Input Sequence

function LOCALBALANCELOCALSEQ(seq, $ring\_process\_group$, $ulysses\_process\_group$)
  $ring\_degree \leftarrow ring\_process\_group.get\_world\_size()$
  $ring\_rank \leftarrow ring\_process\_group.get\_rank()$
  $ulysses\_rank \leftarrow ulysses\_process\_group.get\_rank()$
  $seq\_chunks \leftarrow seq.chunk(2 \times ring\_degree)$
  $reorder\_seq \leftarrow concat([seq\_chunks[r\_rank], seq\_chunks[2 \times rd - r\_rank - 1]])$
  $local\_seq \leftarrow reorder\_seq.chunk(ud, dim = dim)[u\_rank]$
  return $local\_seq$
end function

Unified SP is highly flexible and robust, allowing for the various combinations of the Ulysses degree and the Ring degree, as long as the product of the Ulysses degree and the Ring degree equals the SP degree. When the Ulysses degree equals N, it becomes SP-Ulysses, and when the ring degree equals N, it becomes SP-Ring. Therefore, it covers the ability of both SP-Ulysses and SP-Ring.

Firstly, Unified SP can remove the head number limitation of SP-Ulysses. For example, to run llama3-8B [3] with hc=8 in a 16-degree SP, one can set the Ulysses degree to 8 and the ring degree to 2.

Secondly, Unified SP allows lower bandwidth and topology requirements for network infrastructure by offering a more robust communication pattern. By setting the Ulysses degree to a value between 1 and N, the Attention communication pattern will be a mix of P2P and All2All as shown in Figure 4. Such a communication pattern is particularly well-suited for heterogeneous communication networks, allowing All2All operations to operate in high-bandwidth interconnections while asynchronous P2P communications operate in lower-bandwidth sections. This is applicable to scenarios such as a node in which GPUs connected via PCIe Switch or a cluster of GPU nodes between which the network is Ethernet connected.

Tip 1: We suggest using Unified-SP in place of SP-Ring and SP-Ulysses, as it encompasses the capabilities of both while offering additional benefits.

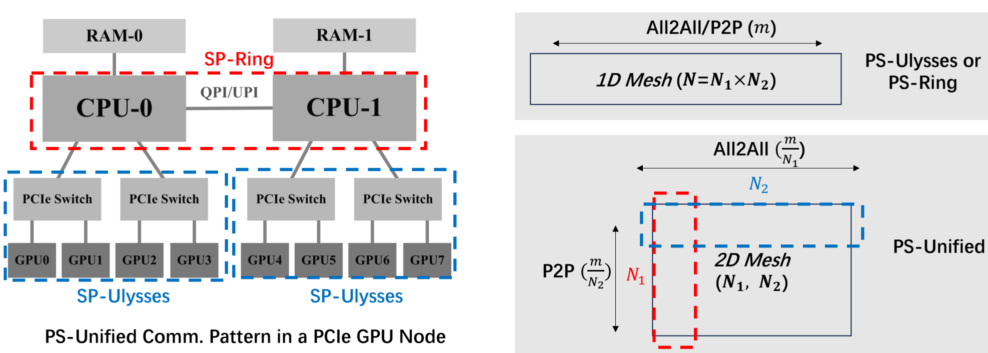

Figure 4: The Unified-SP is more robust to network hardware topology.

4 SP in 4D Parallelism

SP, as a newly emerged parallelism method, how to integrate it into the existing hybrid parallelism framework of Data Parallelism (DP), Tensor Parallelism (TP), and Data Parallelism (PP) has not been thoroughly studied. This section will

[3]https://github.com/meta-llama/llama3

analyze the relationships between SP and DP, TP, and PP, and discuss the best practices for designing 4D parallelism involving sequence dimension parallelism.

As shown in Table 2, we analyze the communication and memory cost of a standard transformer block for different parallelism. The impact of GQA is not reflected in the table, but we will analyze it later. Entries in the Communications Params columns indicate the collective communication operations are conducted on the parameters and gradients of the transformer block. It includes the parameters/gradients of the weight and bias tensors of 4 Linear Layers in the Self-Attention layer, as well as 2 Linear Layers of the FFN layer, amounting to a volume of $12 \times O(d^2)$ elements in the GPT-2 model. Note that llama3 and llama2 has $12 \times O(d^2)$ elements since the intermediate size is lower. Entries in Communications Act columns indicate the collective communications are conducted on a single hidden states tensor belonging to activations, containing $bs \times L \times d$ elements. The memory cost is broken down into model parameters and gradients (**P/G**), Optimizer States (**OS**), and intermediate activation tensors (**Act**).

The Cost of the Communication represents the bandwidth requirements. It is calculated by the product of the communicated element number by an *algorithm bandwidth (algobw)* factor related to the algorithm of collective communication [4]. For the collective communication algorithm of AllReduce, AllGather, ReduceScatter, and AllToAll, the respective algobw factors are $2\frac{n-1}{n}$, $\frac{n-1}{n}$, $\frac{n-1}{n}$, and 1. In the table, we approximate the term $O(\frac{n-1}{n})$ to $O(1)$ for simplicity.

The table is built for the mix-precision training using the fp16(bf16) format. The memory requirement for the model parameters and gradients are P and G Bytes. The Optimizer States (OS), which includes the parameter in fp32, momentum, and variance in the Adam optimizer, is 6 times that of the fp16 parameters. The memory requirement for the peak activation is A bytes. The parallel degree is N.

Table 2: Comparison of Communications and Memory Cost of SP, DP, TP, and ZeRO for a standard transformer block

	Communication (FWD+BWD)				Split	Memory		
	Param	Cost	Act	Cost	Dim	P/G	OS	Act
SP-Ulysses	allreduce	$12O(d^2)$	8*all2all	$8O(bs*L*d)$	hc/L	$P+G$	$6P$	A/N
SP-Ring	allreduce	$12O(d^2)$	P2P	Overlap	L/L	$P+G$	$6P$	A/N
DP	allreduce	$12O(d^2)$	0	0	bs/bs	$P+G$	$6P$	A/N
ZeRO1	allgather+ reducescatter	$12O(d^2)$	0	0	hc/L	$P+G$	$6P/N$	A/N
SP-Unified+ ZeRO1	allgather+ reducescatter	$12O(d^2)$	P2P+8*all2all	$\leq 8O(bs*L*d)$	hc/L	$P+G$	$6P/N$	A/N
SP-Unified+ ZeRO2	allgather+ reducescatter	$12O(d^2)$	P2P+8*all2all	$\leq 8O(bs*L*d)$	hc/L	$P+\frac{G}{N}$	$6P/N$	A/N
SP-Unified+ ZeRO3	2*allgather+ reducescatter	$18O(d^2)$	P2P+8*all2all	$\leq 8O(bs*L*d)$	hc/L	$\frac{P+G}{N}$	$6P/N$	A/N
TP	0	0	4*allreduce	$8O(bs*L*d)$	hc/d	$\frac{P+G}{N}$	$6P/N$	αA
TP-sp	0	0	4*allgather+ 4*reducescatter	$8O(bs*L*d)$	hc/d	$\frac{P+G}{N}$	$6P/N$	A/N

Data Parallelism (DP): In terms of communication performance, SP is inferior to DP. Both SP and DP require the allreduce operation on gradients during backward propagation. The difference in their communication performance lies in the attention module, where SP introduces additional communications overhead for activations. When the Ulysses degree is set to be greater than 1, the communication overhead of SP will be larger than that of DP due to the all2all operations. When using the Ring method, although the additional P2P communication for attention is overlapped, it introduces extra performance issues. The ideal performance is only to reach the performance of DP without communication for attention.

In terms of memory performance, both SP and DP are equivalent, as they all can reduce the activation footprint to 1/N.

Tip 2: We suggest prioritizing the use of DP over SP if possible. Only when the batch size (bs) is insufficient for partitioning should one consider whether to employ SP.

[4] https://github.com/NVIDIA/nccl-tests/blob/master/doc/PERFORMANCE.md

ZeRO: ZeRO [14] is a distributed parameter management method that reduces the storage space requirements of each computing device by sharding the Optimizer States (ZeRO-1), Gradients (ZeRO-2), and Parameters (ZeRO-3) across multiple devices. The memory cost for Optimizer States, Gradients, and Parameters is reduced to $1/N$ of the original.

ZeRO can also operate within an SP process group because partitioning along the batch dimension (bs) or the sequence dimension (L) is equivalent to ZeRO's approach. For ZeRO, SP and DP process groups can be viewed as a combined process group of size $N_sp \times N_dp$.

Tip 3: We suggest that when utilizing SP, it should always be used in conjunction wit ZeRO-1/2. One can also consider employing ZeRO-3, and Offload techniques [5, 15] to trade off communication cost for memory savings.

Tensor Parallelism (TP): The Tensor Parallelism (TP) approach, pioneered by Megatron-LM [4], shards the parameters of models across computing devices. In the TP part of activation tensors, not all are partitioned and distributed across multiple computational devices. Consequently, the memory cost for activations, as reflected in the Table 2, is denoted by the αA, where 0<α<1. Please refer to the Equation(2) of paper [3] for a more precise α. TP has been further refined by Megatron-LM Sequence Parallelism [3], which replaces the an allreduce in TP with an allgather and a reducescatter, and therefore reduces the activation memory cost to A/P. To distinguish Megatron-LM Sequence Parallelism from Ring and Ulysses Sequence Parallelism, we named it TP-sp here.

As shown in Table 2, in terms of communication cost, TP is lower than SP. TP and SP-Ulysses have similar communication costs for activations, but TP dose not require to communicate the parameters. However, in long input sequence scenarios, the communication cost of Activations tends to dominate. For the GPT model with the hidden size is 4K, if the input length during training is 8K, 32K, or 64K, the communication cost for Activations is 1.3x, 5.3x, and 10.7x that of the parameter communication cost, respectively. Also, the gradients and parameters communication can be overlapped by computations. We can consider that when dealing with long sequences, the communication cost of TP-sp and SP is similar.

GQA/MQA can reduce SP communication costs, while the communication cost of TP-sp remains unchanged. Assuming the GQA group number is G, the Ulysses communication cost for K and V is reduced to $1/G$, and the activation communication cost is reduced to $4O(bs * L * d) + 4O(bs * L * d/G)$. Additionally, SP-Ring, which only communicates K, V, also leads to a $1/G$ reduction in P2P bandwidth demand.

Tip 4: We suggest that SP may have an advantage over TP when employing GQA in terms of communication cost, as GQA can reduce the communication cost of SP without affecting TP.

In terms of memory cost, TP-sp holds an advantage over SP. Even when SP employs ZeRO-1/2 to align the memory cost of OS and activations with that of TP-sp, the memory cost of the parameter remains more substantial. However, when SP employs ZeRO3, it achieves a similar memory cost with TP-sp. The SP-Ulysses+ZeRO3 is the exact strategy to scale sequence length employed by the authors of DS-Ulysses [1].

Based on the above analysis, under the same model and input configurations 1, simply switching parallelism from TP-sp to SP-Unified does not increase the trainable sequence length; on the contrary, it is very likely to result in a reduction of the sequence length.

However, SP can still extend the sequence length compared to TP-sp in high parallel degree. When the sequence is very long, as previously analyzed, the proportion of parameter communication volume in the total communication is relatively small. Therefore, the additional communication overhead of an allgather operation introduced by ZeRO has a limited impact.

Tip 5: We suggest switching TP-sp to SP cannot increase the sequence length in training. SP+ZeRO3 can train a similar sequence length as TP-sp.

Due to the inherent limitation of TP-sp's parallelism, which is capped at hc, it is not possible to further reduce the memory cost of activations by increasing the TP-sp paralleled degree. In contrast, SP can continue to expand its parallel degree by leveraging the SP-Ring technique, thereby enabling the training of larger models on a larger scale.

Tip 6: We suggest a higher degree of SP parallelism, which may need to set a large ring degree when the head number is limited, to train a long sequence across a greater number of computational devices. This is an advantage that cannot be achieved with TP-sp approaches.

Pipeline Parallelism (PP): PP partitions transformer blocks across layers, so it is complementary with the SP, which partitions tensors inside a transformer block. Therefore, SP and PP are fully compatible.

Since SP can form a unified parallel group with ZeRO for collective communication, we believe TP should still be placed at the lowest dimension of the 4D parallel group partitioning.

Tip 7: We suggest that in 4D hybrid parallelism, the order of process group dimensionality from low to high is TP, SP-Ulysses, SP-Ring, ZeRO-DP, PP.

5 Experimental Results

5.1 Performance of Unified SP

We first investigate the performance of the attention module using SP-Unified individually. We conducted a benchmark on two GPU nodes to measure the throughput of parallel Attention computations. We selected the same model parameter configuration as used in llama3-8B. Our code is publicly available at github repository`https://github.com/feifeibear/long-context-attention`.

As shown in Table 3, on an 8xA100-SXM4 NVLink node, the highest throughput for both 32K and 128K sequence lengths was achieved when the ulysses-degree is set to 8, that is the same as SP-Ulysses. Ulysses demonstrated a significant advantage over SP-Ring. The lb-ring indicates the load balance Ring-Attention. It consistently outperforms the basic ring attention implementation in terms of efficiency, with the advantage becoming more pronounced as the sequence length increases. This verified the argument in Sec. 3 of SP-Ring, that although communication overhead can be hidden through overlapping with computation, it results in a reduction in computational efficiency.

Table 3: Throughput (iters/sec) of SP-Unified on 8xA100-SXM4 NVLink fwd-only (Ring-Degree x Ulysses-Degree=8)

group_num	bs	seqlen	head_num	head_size	ulysses_degree	basic-ring	lb-ring
4	1	32K	32	128	8	135.569	**136.375**
4	1	32K	32	128	4	103.525	132.979
4	1	32K	32	128	2	91.365	132.979
4	1	32K	32	128	1	81.985	113.79
4	1	128K	32	128	8	2.782	**2.785**
4	1	128K	32	128	4	2.024	2.771
4	1	128K	32	128	2	1.73	2.89
4	1	128K	32	128	1	1.628	2.91

As shown in Table 4, on an 8xL20 PCIe node, the highest throughput for both 32K and 128K sequence lengths was achieved when the ulysses_degree is set to 4, when ring_degree=2.

Table 4: Throughput (iters/sec) of SP-Unified on 8xL20 PCIe fwd-only (Ring-Degree x Ulysses-Degree=8)

group_num	bs	seqlen	head_num	head_size	ulysses_degree	basic-ring	lb-ring
4	1	8K	32	128	8	57.346	57.098
4	1	8K	32	128	4	**153.134**	152.189
4	1	8K	32	128	2	415.5	454.93
4	1	8K	32	128	1	358.595	361.969
4	1	32K	32	128	8	15.229	14.262
4	1	32K	32	128	4	28.584	**32.818**
4	1	32K	32	128	2	44.348	62.754
4	1	32K	32	128	1	40.478	58.377
4	1	128K	32	128	8	2.563	2.586
4	1	128K	32	128	4	3.217	**4.235**
4	1	128K	32	128	2	3.399	5.476
4	1	128K	32	128	1	3.131	5.186

5.2 End-to-end SP Performance in Megatron-LM

We have integrated the SP-Unified method into Megatron-LM. As of now, Megatron-LM has an initial version of SP-Ring, but there is no implementation of Ulysses available. We did not utilize the open-source implementation of SP-Ulysses in Megatron-DeepSpeed, as we found that the performance of Megatron-DeepSpeed significantly lagged behind that of Megatron-LM. Additionally, we also discovered that the SP-Ulysses in Megatron-DeepSpeed was not compatible with Tensor Parallelism.

Our software is built on Megatron-LM commit ID 2196398, dated April 12, 2024. The SP-Ring implementation is based on Megatron-LM's built-in Context Parallel, while SP-Ulysses is implemented using the code from our repository. It has already incorporated the load-balancing strategy discussed in this paper. We used the docker image nvcr.io/nvidia/nemo:24.03 for our experiments. We use ZeRO-1 for DP and SP by default. We always employ Sequence Parallelism for Tensor Parallelism, a.k.a TP-sp. We do not use gradient accumulation.

Our experimental hardware setup consists of two GPU nodes, each with 8xA800 GPUs connected via 400GB/s NVLink, and a node-to-node communication bandwidth of 1.6 Tbps RDMA.

5.3 SP vs. DP

Firstly, we compare the performance of SP and DP under the same LLAMA2-7B workload on a single node of A800 GPUs. The global batch size is 8. We use the SP-Unified, and pick the best ulysses-degree and ring-degree settings. In a single node, ulysses-degree as 8 usually achieves the best performance. As shown in Figure 5, DP outperforms SP-Unified across various input sequence lengths, which confirms our conclusion in Tip 2.

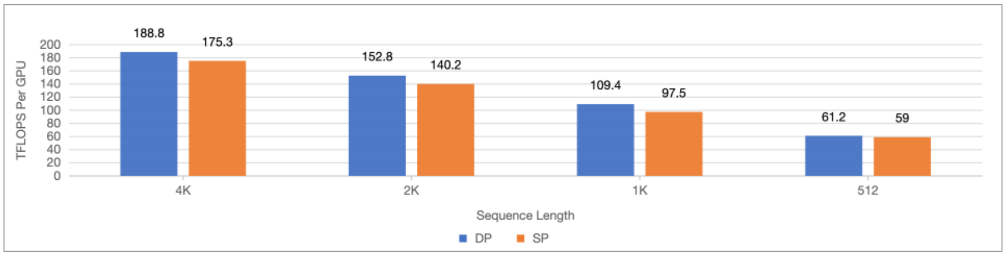

Figure 5: Comparison DP and SP on LLAMA2-7B Task with global bs=8.

5.4 Hybrid SP and TP

Table 5 presents the performance of the llama2-7B model on a single node of 8xA800 GPU 80GB. The longest sequence length that can be achieved is 64K, and the best performance is obtained when tp-degree=4 and ulysses-degree=2. It outperforms TP-sp-only by 10%. When seqlen=64K, SP-only will bring OOM issue, which echos our conclusion in Tip 5 that SP is less memory-efficient than TP-sp.

When global-bs=16 and seqlen=30K, SP-Ulysses delivers the optimal performance, significantly surpassing other SP and TP hybrid strategies. It is also 26% better than TP-sp-only in throughput. This indicates that, despite the similar communication cost of SP-Ulysses and TP-sp, SP-Ulysses is more communication efficient than TP-sp in practice in an NVLINK-connected setting, and communication patterns of hybrid SP-Ulysses and SP-Ring sometimes bring better performance.

Table 5: LLAMA2-7B FLOPS per GPU of hybrid parallelism using TP and SP-Unified on 1 8xA800 NVLink node.

seqlen	global-bs	tp-degree	ulysses-degree	ring-degree	FLOPS/GPU	MFU
64K	1	4	2	1	**260.2**	**0.83**
64K	1	4	1	2	255	0.82
64K	1	8	1	1	238.9	0.77
30K	16	2	4	1	239.2	0.77
30K	16	2	1	4	226.6	0.73
30K	16	4	1	1	230.1	0.74
30K	16	1	8	1	**250.6**	**0.80**
30K	16	1	1	8	218	0.70
30K	16	8	1	1	198	0.63

We employed a hybrid parallel strategy combining TP and SP-Unified to benchmark the training throughput of LLAMA3-8B across two nodes. The results are shown in Table 6. In most of cases, the global batch size is fixed to be 1, aiming to maximize the sequence length. Since llama3-8B has only 8 heads, the maximum product of ulysses-degree and tp-degree is 8. Table 6 presents throughput in FLOPS on 64K, 80K and 120K.

The optimal performance for sequence lengths of 64K and 80K is achieved with SP-only without TP-sp, the optimal setting of both is ulysses-degree=4 and ring-degree=4. For sequence length is 64K and 80K, the unified SP outperforms the SP-Ring by 13% and 12%, respectively. These results echo our conclusion in Tip 1.

Table 6: LLAMA3-8B FLOPS per GPU of hybrid parallelism using TP and SP-Unified on 2 8xA800 NVLink nodes.

seqlen	global-bs	tp-degree	ulysses-degree	ring-degree	FLOPS/GPU	MFU
64K	1	1	8	2	220.3	0.71
64K	1	1	4	4	**222.2**	**0.71**
64K	1	1	2	8	209.2	0.67
64K	1	1	1	16	196.9	0.63
64K	1	8	1	2	**209.7**	**0.67**
64K	1	4	2	2	197.9	0.63
64K	1	2	4	2	141.7	0.45
64K	1	2	2	4	144.4	0.46
64K	1	4	1	4	198.1	0.63
64K	1	2	1	8	163.8	0.53
64K	2	8	1	1	228.2	0.73
80K	1	1	8	2	245.4	0.79
80K	1	1	4	4	**247.8**	**0.79**
80K	1	1	2	8	233.2	0.75
80K	1	1	1	16	221.1	0.71
80K	1	8	1	2	226.6	0.73
80K	1	4	2	2	**228.8**	**0.73**
80K	1	2	4	2	184.8	0.59
80K	1	2	2	4	184.4	0.59
80K	1	4	1	4	216.8	0.69
80K	1	2	1	8	201.6	0.65
80K	2	8	1	1	240.3	0.77
120K	1	4	2	2	**266**	**0.85**
120K	1	2	4	2	238.3	0.76
120K	1	8	1	2	254.5	0.82
120K	1	4	1	4	263.3	0.84

For sequence length are 64K and 80K, we can increase the global batch size (global-bs) for TP+SP hybrid setting when tp-degree=8. However, SP-only always has OOM issue with global-bs=2. Increasing the global-bs to 2, TP+SP has 2.7% improvement in throughput to SP-only at the sequence length 64K. But at sequence length 80K, TP+SP with global-bs as 2 is still worse than SP-only with global-bs as 1.

When the sequence length reaches 120K, the optimal performance is achieved with tp-degree=4 and ulysses-degree=2, reaching 266 TFLOPS, with an MFU of 0.85. At this sequence length, SP-only meets an OOM issue. This confirms our Tip 5 again, which implies that the memory efficiency of SP is inferior to that of TP-sp. It is noteworthy that two other TP+SP hybrid settings yield similar performance, 0.82 and 0.84 in MFU, which is quite close to the optimal.

Table 7: Exploring Upper Bound of Sequence Length for LLAMA3-8B using TP and SP-Unified on 2 8xA800 NVLink nodes.

seqlen	global-bs	tp-degree	ulysses-degree	ring-degree	FLOPS/GPU	MFU
160K	1	4	2	2	284.5	0.91
160K	1	8	1	2	280.9	0.90
208K	1	8	1	2	269.8	0.86
160K	1	4	1	4	285.8	0.92
190K	1	4	1	4	**285.8**	**0.92**

We explored the upper bound of sequence length on 2 nodes, and the results are presented in Table 7. The largest sequence length was achieved with tp-degree=8 and ring-degree=2. At this point, SP-only could not run due to OOM. Compared to SP, TP-sp is more memory-efficient, hence for training the longest sequences, parallel degrees are limited by 8 attention head numbers should all be assigned to TP-sp.

As the sequence length increases from 64K to 208K, MFU is generally improving. This is due to the $O(L^2)$ computational cost of attention quadratic increases, which results in a decreasing communication-to-computation ratio.

6 Future Work

SP on Large Scale Cluster We believe that SP is highly beneficial for extremely large scale LLM training tasks. Currently, for publicly disclosed large-scale model training tasks [16, 17] over 10K GPUs, SP has not been utilized. This is because these training tasks were started before November 2023, when the SP methods were not yet mature.

Firstly, SP can introduce the dimension that can be partitioned from the sequence length, alleviating the constraints hindered by batch size limitations. MegaScale project [17] uses a large global batch size to increase DP degree which impacts convergence. However, we can increase the SP degree instead of DP degree to reduce the global batch size, thereby avoiding the convergence problem.

Secondly, increasing the SP degree can decrease the activation cost, allowing for longer model context length during training. Theoretically, the SP-Ring degree can be increased arbitrarily, whereas the TP degree is limited by the number of heads. Our experimental results on two nodes have not yet demonstrated this advantage of SP.

SP+ZeRO-3 Megatron-LM does not officially support ZeRO-3, possibly because the TP-sp already reduces memory cost for parameters and gradients, which is also the target of ZeRO-3. We have claimed that ZeRO-3 is highly compatible with SP in Tip 2. Given that Megatron-LM already has an implementation of SP-Ring, ZeRO-3 becomes a necessary feature.

SP+MoE As more and more models transition to a Mixture-of-Experts (MoE) architecture, the research on combining SP with expert parallelism (EP) becomes increasingly significant. Since TP has been proven to be better replaced by EP (Expert Parallelism) [18] due to its inferior communication cost, we argue that SP should also be given a lower priority than EP. This paper does not conduct experiments on MoE models, but we plan to test it in the future.

7 Conclusion

In this paper, we propose an approach that unifies DeepSpeed-Ulysses and Ring-Attention for sequence parallelism. This method encompasses the capabilities of both techniques, broadening the applicability and delivering superior performance in some cases. We systematically analyzed the interplay between sequence parallelism and other established parallelism methods, deriving a set of best practice suggestions. These suggestions have been validated through experimental results obtained from two GPU nodes.

8 Acknowledgements

We express our gratitude to Zilin Zhu from Tencent. The code utilized in our research was built from his GitHub repository, and he also owns the authorship of Figure 3 presented in this paper.

References

[1] Sam Ade Jacobs, Masahiro Tanaka, Chengming Zhang, Minjia Zhang, Leon Song, Samyam Rajbhandari, and Yuxiong He. Deepspeed ulysses: System optimizations for enabling training of extreme long sequence transformer models. *arXiv preprint arXiv:2309.14509*, 2023.

[2] Hao Liu, Matei Zaharia, and Pieter Abbeel. Ring attention with blockwise transformers for near-infinite context. *arXiv preprint arXiv:2310.01889*, 2023.

[3] Vijay Anand Korthikanti, Jared Casper, Sangkug Lym, Lawrence McAfee, Michael Andersch, Mohammad Shoeybi, and Bryan Catanzaro. Reducing activation recomputation in large transformer models. *Proceedings of Machine Learning and Systems*, 5, 2023.

[4] 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.

[5] Jie Ren, Samyam Rajbhandari, Reza Yazdani Aminabadi, Olatunji Ruwase, Shuangyan Yang, Minjia Zhang, Dong Li, and Yuxiong He. Zero-offload: Democratizing billion-scale model training. In *2021 USENIX Annual Technical Conference (USENIX ATC 21)*, pages 551–564, 2021.

[6] Shenggui Li, Fuzhao Xue, Yongbin Li, and Yang You. Sequence parallelism: Making 4d parallelism possible. *arXiv preprint arXiv:2105.13120*, 2021.

[7] Shenggui Li, Hongxin Liu, Zhengda Bian, Jiarui Fang, Haichen Huang, Yuliang Liu, Boxiang Wang, and Yang You. Colossal-ai: A unified deep learning system for large-scale parallel training. In *Proceedings of the 52nd International Conference on Parallel Processing*, pages 766–775, 2023.

[8] 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. *arXiv preprint arXiv:2310.03294*, 2023.

[9] Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in Neural Information Processing Systems*, 35:16344–16359, 2022.

[10] 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. *arXiv preprint arXiv:2305.13245*, 2023.

[11] Noam Shazeer. Fast transformer decoding: One write-head is all you need. *arXiv preprint arXiv:1911.02150*, 2019.

[12] William Brandon, Aniruddha Nrusimha, Kevin Qian, Zachary Ankner, Tian Jin, Zhiye Song, and Jonathan Ragan-Kelley. Striped attention: Faster ring attention for causal transformers. *arXiv preprint arXiv:2311.09431*, 2023.

[13] Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

[14] Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. Zero: Memory optimizations toward training trillion parameter models. In *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–16. IEEE, 2020.

[15] Jiarui Fang, Zilin Zhu, Shenggui Li, Hui Su, Yang Yu, Jie Zhou, and Yang You. Parallel training of pre-trained models via chunk-based dynamic memory management. *IEEE Transactions on Parallel and Distributed Systems*, 34(1):304–315, 2022.

[16] Ziheng Jiang, Haibin Lin, Yinmin Zhong, Qi Huang, Yangrui Chen, Zhi Zhang, Yanghua Peng, Xiang Li, Cong Xie, Shibiao Nong, et al. Megascale: Scaling large language model training to more than 10,000 gpus. *arXiv preprint arXiv:2402.15627*, 2024.

[17] Meta AI Team. Introducing meta llama 3: The most capable openly available llm to date. `https://ai.meta.com/blog/meta-llama-3/`, 2024. Accessed: May 2024.

[18] Changho Hwang, Wei Cui, Yifan Xiong, Ziyue Yang, Ze Liu, Han Hu, Zilong Wang, Rafael Salas, Jithin Jose, Prabhat Ram, et al. Tutel: Adaptive mixture-of-experts at scale. *Proceedings of Machine Learning and Systems*, 5, 2023.