CHARM+: Cross-Hardware Attention with Re-Merge Multistream Mechanism

Xinhang Zhang* **Boning Zhang*** **Chengchun Liu** **Chunpu Li**

Lei Sun **Weifeng Zhang** **Limin Xiao†**

Lenovo Research

Abstract

As sequence lengths continue to grow, distributed attention based on sequence parallelism (e.g., Ring-Attention) has become a mainstream solution for large language models (LLMs). Nevertheless, inherent communication dependencies limit computation–communication overlap, reducing GPU utilization. Moreover, in cross-hardware (NUMA or node) architectures, due to imbalanced communication bandwidth, frequent communication synchronization will limit overall performance. Based on these insights, we propose CHARM+, Cross-Hardware Attention with Re-Merge Multistream Mechanism. We reorganize sequential communications into several levels and achieve intra-level parallelism and computation overlap. Furthermore, we decouple cross-hardware communication from intra-hardware communication, eliminating synchronization overhead caused by bandwidth imbalance. Forward and backward algorithms were evaluated on multiple hardware architectures and consistently outperformed state-of-the-art methods. Compared to Ring-Attention, our forward and backward algorithms achieve average speedups of $1.3\times$ and $1.7\times$ on a dual-NUMA PCIe architecture, and $2.8\times$ and $3.9\times$ on a dual-node NVLink architecture, respectively. [3]

1 Introduction

As large language models (LLMs) scale up [3], longer context lengths are required to fully exploit their performance [29, 24]. Notably, LLM training requires processing long documents to enhance model capabilities [5]. Meanwhile, the inference of LLMs involves cross-document understanding [1], long program comprehension [18], and multi-turn interactive dialogues [15]. To handle this, self-attention is the core mechanism for capturing long-range dependencies using queries (q), keys (k), and values (v) [9, 25]. However, its computation and memory costs scale with sequence length, making long-context training and inference expensive [13, 27]. Therefore, sequence parallelism (SP) and distributed attention have become mainstream approaches. They partition the sequence across accelerators to reduce per-device overhead, enabling longer context windows [23, 12].

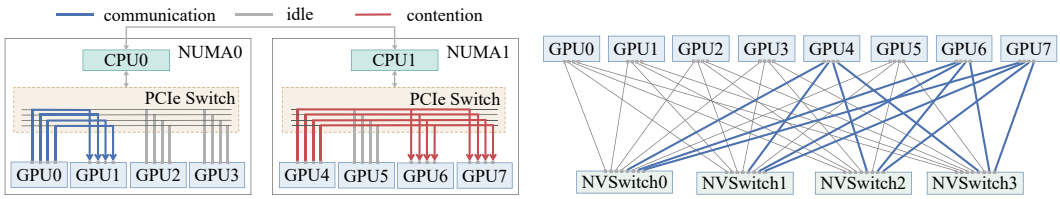
Nevertheless, existing SP methods (e.g., Ring-Attention) exhibit two limitations. Firstly, due to the communication dependency, each communication relies on the completion of the preceding communication [28]. Consequently, it limits the communication-computation overlap and reduces GPU utilization. Secondly, in the cross-hardware (Non-Uniform Memory Access, NUMA or node),

*Equal contribution.

†Corresponding author.

[3]Code: `https://github.com/Lenovo-ICI/CHARM`

40th Conference on Neural Information Processing Systems (NeurIPS 2026).



(a) Blocking comm due to link contention in PCIe (b) Non-blocking comm on dedicated links in NVLink

Figure 1: Communication under PCIe and NVLink. The left of (a) shows single communication at full bandwidth without contention, while the right shows multiple communications at limited bandwidth with contention. (b) shows multiple communications at full bandwidth without contention.

imbalanced bandwidth leads to extra synchronization overhead [21]. Therefore, communication dependencies cause GPUs to idle, degrading overall performance.

Based on these observations, we propose CHARM+, Cross-Hardware Attention with Re-Merge Multistream Mechanism. We design a merge-based communication mechanism to reorganize the sequential communications into several levels and achieve intra-level independence. Secondly, we leverage multiple communication streams to better overlap computation and communication. Furthermore, by decoupling intra- and inter-hardware communication, we avoid performance degradation due to synchronization overhead. As shown in Figure 2, GPU idle time has been significantly reduced, leading to substantial performance improvements. Our contributions are summarized as follows:

- **We propose a novel communication and parallel mechanism.** To optimize the ring pattern, we reorganize communications into logarithmic levels and design a multistream parallelism for kv and q to achieve intra-level parallelism.
- **We develop a Cross-Hardware Query Parallelism to decouple intra- and inter-hardware communication.** By isolating cross-hardware q and kv communications, the parallelism avoids synchronization induced by bandwidth imbalance and improves performance.
- **We implement both forward and backward of our method and conduct extensive experiments across different hardware architectures.** The results demonstrate that our forward and backward consistently outperform Ring-Attention, achieving maximum speedups of $1.4\times$ and $1.9\times$ on PCIe, and $3.2\times$ and $4.3\times$ on NVLink.

2 Related Work

2.1 Interconnection Architecture

Most large-scale training and inference are deployed on multi-GPU systems connected via either PCIe or NVLink [19, 6]. Figure 1 illustrates the interconnects of these two architectures. PCIe-based systems often suffer from link contention due to concurrent communication requests, while NVLink provides high-bandwidth, dedicated links that enable contention-free communication [11]. Furthermore, both architectures have imbalanced communication bandwidth between cross-hardware (i.e., cross-NUMA or cross-node) and intra-hardware [17]. Consequently, it incurs substantial synchronization overhead, severely degrading communication performance.

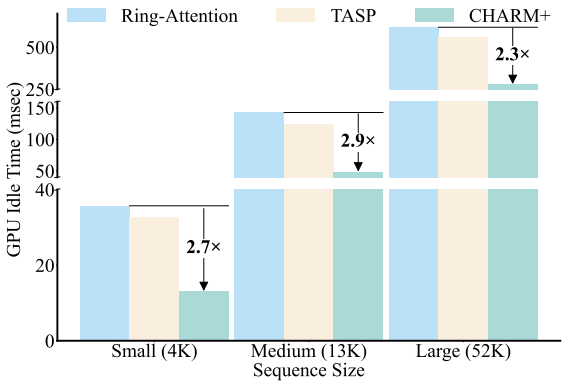


Figure 2: GPU Idle Time. By setting various sequence lengths, we compare GPU idle time with Ring-Attention and TASP. Our method achieves higher GPU utilization.

2.2 Sequence Parallelism

As long-context requirements in LLMs continue to grow, the sequence parallelism and the distributed attention mechanism have become critical techniques for scaling at-

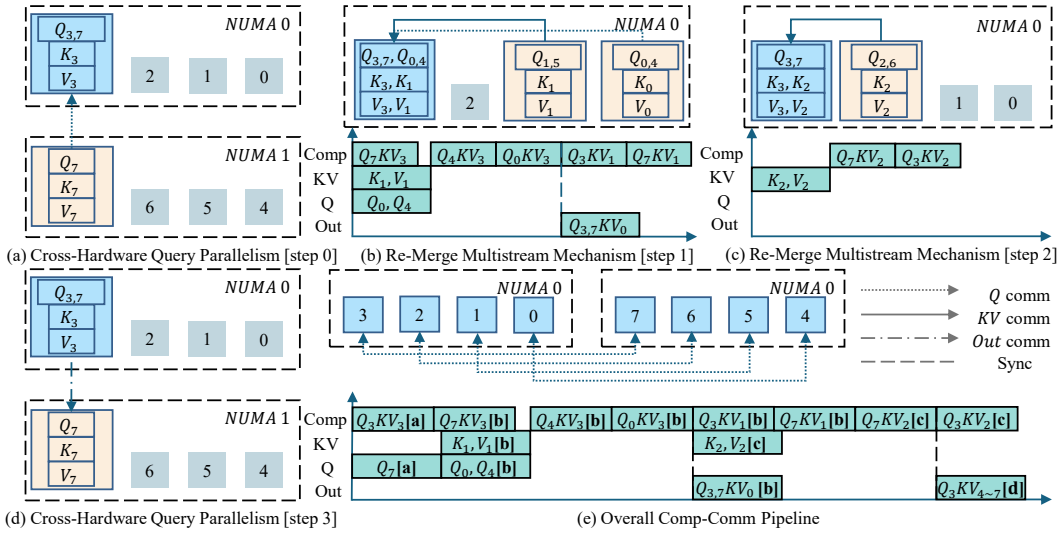


Figure 3: Overall Framework of CHARM+. (a) - (d) illustrate the complete process of the algorithm using an 8-GPU system with a dual-NUMA configuration. (a) displays parallel communication of q is first performed across NUMA. As a result, GPU 3 receives the q from GPU 7, eliminating the need for further cross-NUMA communication. (b) shows the parallel communication and computation of q and kv. In this step, GPU 3 communicates q with GPU 0 and kv with GPU 1. After completing the computation of $Q_{0,4}$, GPU3 communicates back to GPU 0 and receives the output. (c) performs the last step of RM+, where GPU 3 only communicates kv with GPU 2. (d) shows that once all kv results are computed, the output is communicated back to GPU 7. (e) illustrates the overall cross-NUMA communication and the full overlap of computation and communication of GPU3 happened in (a)-(d).

tention by distributing sequence computation across multiple devices [12, 23]. Existing methods (e.g., Ring-Attention) rely on distributed kv exchange, where kv blocks are transmitted across devices in a ring topology [14] and processed incrementally [16]. However, there exists a dependency in ring-based communication: each communication cannot start until the previous one completes. Moreover, each communication can only overlap with a single computation [22]. Consequently, as communication time grows, this dependency prolongs GPU idle time and reduces effective compute utilization.

Several approaches have been proposed to improve the efficiency and scalability of Ring-Attention. TASP [26] aims to better utilize available communication links by adopting the all-to-all communication. Attention-2D [10] splits the self-attention computation into two dimensions along q and kv. Mesh-Attention [4] replaces one-dimensional Ring-Attention with a two-dimensional tiled assignment. Striped-Attention [2] reorders the input sequence into uniformly distributed blocks. However, these approaches do not optimize the ring-based pattern. Reducing the ring length or enabling multiple rings to communicate in parallel does not eliminate the communication dependencies, nor does it improve the communication–computation overlap. Therefore, efficient computation–communication overlap and high GPU utilization remain challenges in distributed attention.

3 Method

This section presents our method illustrated in Figure 3[4]. We first explain our motivation and introduce a merge-based communication mechanism. Secondly, we propose a multistream parallelism to improve the communication-computation pattern in the communication mechanism. Furthermore, we leverage a cross-hardware query parallelism to mitigate the imbalanced bandwidth between cross-hardware (NUMA or node) communication and intra-hardware communication and use intra-pipeline to hide cross-hardware communication. All notations utilized in this section are described in Table 1.

[4]For clarity, the computation of the FlashAttention in the illustration will ignore causal masks.

Table 1: Notation Description.

Notation	Description	Notation	Description
Q, q	Query matrix	K, k	Key matrix
V, v	Value matrix	n	Degree of sequence parallelism
O	Unhidden communication time	$\ell(\cdot)$	Function that clips value to zero
$T_{\text{comm}}(\cdot)$	Communication time of a tensor	$T_{\text{comp}}(\cdot)$	Computation time of Attention

3.1 Motivation

As introduced in Section 2, the current ring-based attention involves gradually passing kv through a ring path for attention computation and update. This mechanism enables partial overlap between communication and computation. However, dependencies between communications prevent a communication from starting before the previous one is complete. To maximize GPU utilization, we seek to maximize communication–computation overlap, defined as follows:

$$\min O = \sum_{i=1}^{N} \ell(T^i_{\text{comm}}(k, v) - T^i_{\text{comp}}(q, k, v)) \tag{1}$$

Let N denote the length of critical path that is non-overlapping, $T^i_{\text{comm}}(\cdot)$ and $T^i_{\text{comp}}(\cdot)$ are the i_{th} communication time, the i_{th} computation time of FlashAttention [8]. We define O as the unhidden communication time, where the GPU is idle. Therefore, minimizing O maximizes GPU utilization.

For Ring-Attention [16], due to the ring structure, this objective can be expressed as:

$$O_{\text{ring}} = \sum_{i=1}^{n-1} \ell(T^i_{\text{comm}}(k, v) - T^i_{\text{comp}}(q, k, v)) \tag{2}$$

This $O(n)$ critical path dependency becomes the primary bottleneck limiting the ability to overlap communication and computation. In particular, the bandwidth imbalance between cross-hardware (NUMA or node) and intra-hardware communication causes extra communication synchronization to significantly increase the latency of each step along the critical path. Therefore, this effect further exacerbates the overall bottleneck, we further corroborate these observations with Nsight profiling results in Figure 9a of Section C.1. Based on these observations, our motivation becomes:

> "*How can we efficiently overlap computation and communication while simultaneously minimizing the overall critical path of the execution?* "

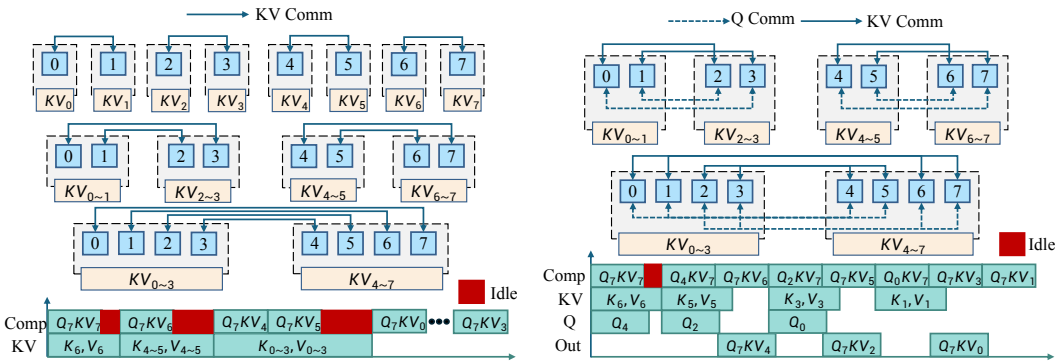
As inspired by the traditional merge algorithm [20, 7], we present a communication mechanism. Compared to the ring-based kv exchange, the mechanism redesigns this process as kv aggregation. The aggregation is structured as a logarithmic-level merge (in Figure 4a, a sequence parallelism of 8 corresponds to a three-level merge). At each level i, kv are aggregated in parallel between device pairs with a stride of 2^{i-1}. As communication dependencies persist between merge levels, the critical path of the merge mechanism becomes $O(\log n)$, which is significantly shorter than Ring-Attention.

For the merge communication mechanism, we model the communication time at each merge level as:

$$C^{(1)}_i = \sum_{j=1}^{2^{i-1}} T^j_{\text{comm}}(k, v) \tag{3}$$

At the i_{th} merge level, the communication mechanism performs 2^{i-1} communications. Meanwhile, the computation at level i processes the kv received at the communication of level $i-1$. Thus, KV communication at level i overlaps with computation at level $i-1$, whose cost is $2^{\ell(i-2)}$. Consequently, the unhidden communication time is given by:

$$O_{\text{merge}} = \sum_{i=1}^{\log_2(n)} \ell(C^{(1)}_i - \sum_{j=1}^{2^{\ell(i-2)}} T^j_{\text{comp}}(q, k, v)) \tag{4}$$



(a) Merge Communication Mechanism.

(b) Re-Merge Multistream Mechanism.

Figure 4: The Computation and Communication Overlap on GPU 7. For 8 GPUs, (a) contains three levels. Each GPU communicates at groups of 1, 2, and 4 at different levels, aggregating the communicated kv while overlapping computation. (b) maintains consistency with (a) in the first levels, but further enables multistream communication and computation of q and kv at other each merge level. Each GPU concurrently communicates q and kv with different GPUs, computes the q of other GPU and communicates the output back. The red region indicates GPU idle time.

As can be observed, compared to the Ring-Attention, the merge communication mechanism restructures the linear communication path into a merge path with logarithmic level, significantly shortening the communication critical path. Each merge level accommodates more computation, increasing opportunities to overlap communication with computation and improving efficiency.

3.2 Re-Merge Multistream Mechanism

The merge communication mechanism reduces dependencies between ring communications, but communication at each merge level still follows the ring pattern. To further increase communication–computation overlap, we design the Re-Merge Multistream Mechanism, named RM+. It still follows the structure of the merge communication mechanism, but optimizes the communication at each merge level to enable more communication-computation overlap.

As described in Section 3.1, Ring-Attention involves communication only for kv, and each round of kv communication can start only after the previous round completes. Notably, the communication of q is independent of the kv communication. By exploiting this independence, we parallelize q communication with kv communication, improving communication-computation overlap, as illustrated in Figure 4. Moreover, we aim to overlap the computation after the final merge level(e.g., $Q_7KV_0 \cdots Q_7KV_3$ in Figure 4a. During each merge level, kv is fetched sequentially from peer GPUs for computation. In addition, parallel communication of q and kv further reduces the communication time of each merge level. Consequently, as illustrated in Figure 4b, we obtain:

$$C_i^{(2)} = \sum_{j=1}^{2^{\ell(i-2)}} \max\left(T^j_{\text{comm}}(q), T^j_{\text{comm}}(k,v)\right) \tag{5}$$

Note that, Eq. (5) follows Eq. (3). Due to the parallel communication between q and kv, the communication time is bounded by the maximum. Meanwhile, advancing the computation after the final merge level increases computations at earlier merge levels, thereby reducing the unhidden communication time. Specifically, all communication except the final one at each level can be overlapped, with the last communication deferred to the next merge level. Therefore, for the i_{th} merge level, there are 2^{i-1} computations. Consequently, the unhidden communication time is:

$$O_{\text{RM+}} = \sum_{i=1}^{\log n} \ell(C_i^{(2)} - \sum_{j=1}^{2^{i-1}} T^j_{\text{comp}}(q,k,v)) \tag{6}$$

Under the assumption that all parallel communications sustain identical bandwidth, Eq. (4) outperforms Eq. (2) due to its shorter critical path and greater communication overlap. However, parallel communication of multistream can lead to PCIe link contention, increasing latency. Consequently, on

the PCIe architecture, the single communication of RM+ can be longer than that in Ring-Attention. Therefore, performance on a GPU system using NVLink is significantly better than on a PCIe architecture, as further demonstrated in Section 4.4.

3.3 Cross-Hardware Query Parallelism

In multi-hardware architectures (NUMA or Node), each KV communication in Ring-Attention simultaneously involves both intra-hardware and cross-hardware communication. Since cross-hardware communication typically provides lower bandwidth, the faster intra-hardware communication is forced to be idle while waiting for the communication synchronization, leading to substantial performance inefficiency. As illustrated in Figure 5, we propose a novel parallelism where cross-hardware communication of the q is interleaved with the intra-node pipeline. The pipelines process the current queries (Q) from different pipelines with local KV using the same computation and communication flow, as detailed in Section 3.2. Notably, since all kv and q communications occur in the pipeline come from the same NUMA or node, the bottleneck caused by synchronized communication is largely eliminated. With effective overlap illustrated in Figure 5, after all computations are completed, only a final out communication is not hidden. Therefore, as the number of nodes increases, effective overlap ensures that only a single communication remains exposed, making the advantage of our method increasingly pronounced, which is further validated in the ablation study in Section 4.4 and Section 4.5. We provide Nsight profiling results of CHARM+ in Figure 9b of Section C.1, demonstrating that our overlap strategy significantly outperforms Ring-Attention.

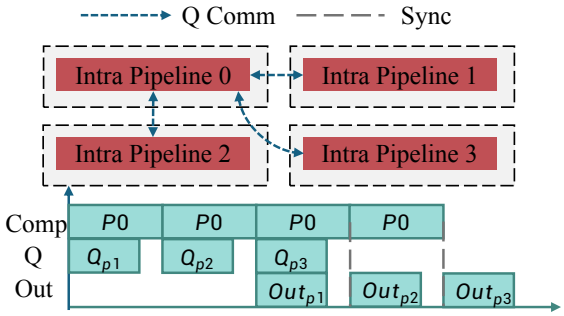


Figure 5: Cross-Hardware Query Parallelism. The complete process is illustrated with pipeline 0 as an example. Each intra-pipeline processes queries (Q) from other pipelines and uses the same workflow with its local KV, returning the final out back. Effective overlap allows inter-pipeline communication to be hidden by intra-pipeline, further improving performance.

4 Experiments

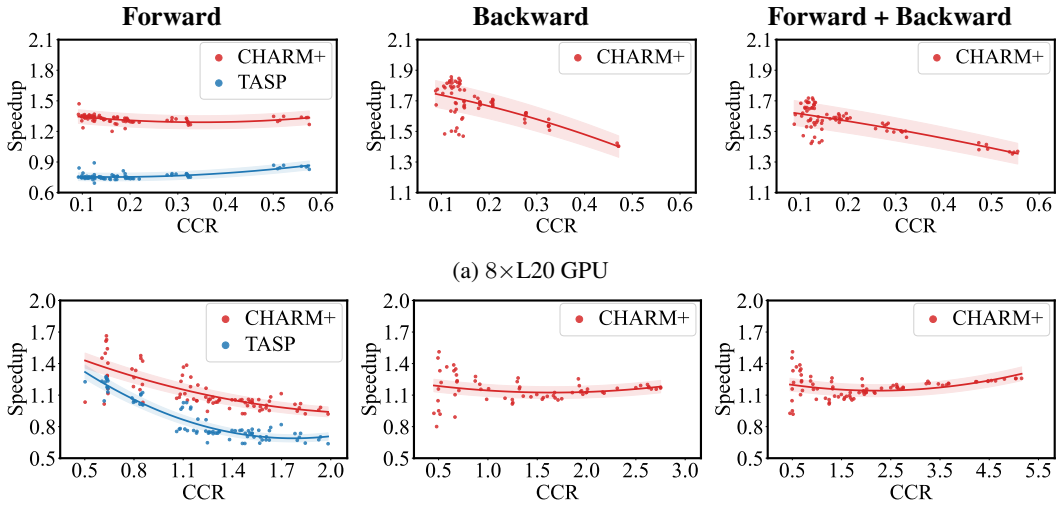
4.1 Experimental Setup

We conduct experiments on GPU systems with different configurations and detailed specifications are provided in Table 2. We use official NCCL and RDMA test benchmarks to measure the intra-hardware and cross-hardware unidirectional P2P bandwidth. On L20, we evaluate the performance in both cross-NUMA and intra-NUMA settings. Furthermore, we explore the impact of link contention in PCIe-based systems. On H20 and H100, we evaluate cross-node and intra-node performance and further compare results under NVLink architecture, where link contention is absent.

Table 2: The GPU Configuration and Bandwidth.

GPU	NVLink	Num	TFLOPS	Bandwidth (GB/s)	
				Cross	Intra
L20	✗	8	119.5	14.7	24.0
H20	✓	16	148.0	42.6	297.2
H100	✓	32	989.0	42.5	365.1

We evaluate our method under diverse configurations by varying batch size (B), sequence length (S), and hidden dimensions (H). Specifically, sequence lengths range from 1K to 100K (up to 200K in some scenarios), with four hidden-dimension configurations formed by combining dimensions of 12 or 20 with sizes of 128 or 192. We conduct a grid search over B, S, and H, ensuring that the product BS does not exceed 500K due to memory constraints. Each configuration is evaluated over 100 iterations following 20 warm-up iterations, with results averaged across the measured runs. To enable a unified characterization of different configurations, we use the computation-to-communication ratio



(a) 8×L20 GPU

(b) 8×H20 GPU

Figure 6: Single-node Experiments on 8×L20 (PCIe) and 8×H20 (NVLink). Speedup is normalized to Ring-Attention, comparing forward, backward, and end-to-end time. In different architectures, our method consistently outperforms the baseline and TASP.

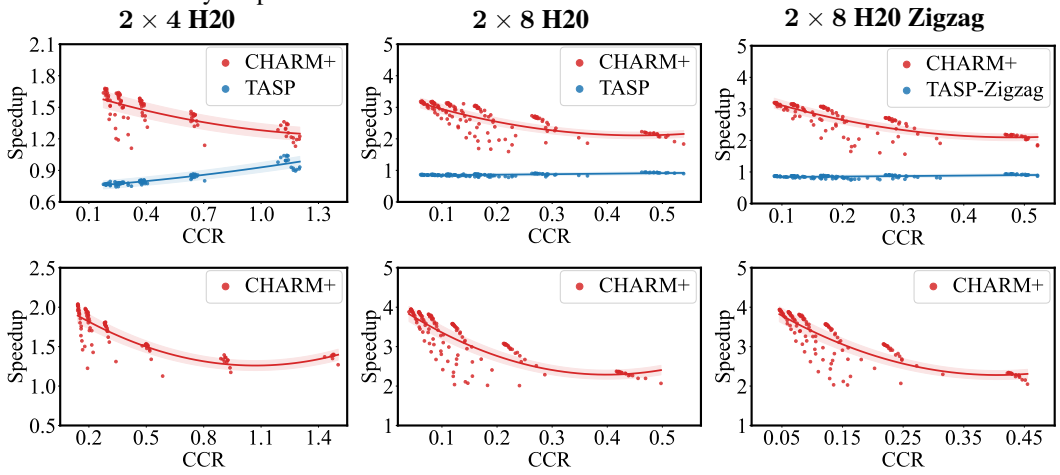


Figure 7: Multi-node Experiments on 8×H20 and 16×H20 (NVLink). Speedup is normalized to Ring-Attention and Ring-Zigzag. In the forward (top) and the backward (bottom) comparison, our method achieves consistently superior performance. Particularly, the 2×4 and 2×8 experiments highlight that our method's performance grows as the number of GPUs increases.

(CCR) as a single metric. Following TASP [26], we define the CCR as:

$$CCR = \sum_{i=1}^{N} T^{i}_{\text{comp}(q,k,v)} / \sum_{i=1}^{N} T^{i}_{\text{comm}(k,v)} \tag{7}$$

Here, q, k, and v are tensors that depend on the sequence length, the batch size, the hidden dimensions, and the number of attention heads. We accumulate the computation and communication time of Ring-Attention under sequential execution. The final metric is defined as the ratio of the computation time to the communication time (detailed analysis of CCR is provided in Section B.1). We compare our method against Ring-Attention [16], TASP, and their corresponding zigzag variants [2][5].

4.2 Single-node Evaluation

As illustrated in Figure 6, we perform single-node experiments on L20 with 8 GPUs under a dual-NUMA PCIe architecture and on H20 with 8 GPUs using NVLink. Our method outperforms TASP

[5]As TASP does not provide the implementation of the backward algorithm, we compare forward performance only.

Table 3: Memory Overhead. By setting various parameter scales, we compare the memory overhead with Ring-Attention. Our method shows comparable overhead to Ring-Attention.

Batch Size	Seq Len	Heads	Ring (MB)	CHARM+ (MB)	Batch Size	Seq Len	Heads	Ring (MB)	CHARM+ (MB)
16	1024	32	2,179	2,215	2	53,760	12	13,123	13,467
16	1024	64	3,701	3,881	4	53,760	12	19,301	20,565
64	8192	32	45,079	45,423	2	26,880	12	10,745	10,803
128	8192	16	44,720	45,432	4	26,880	12	14,029	14,425

and Ring-Attention, even in the presence of link contention (Figure 6a). The poor performance of TASP on L20 is mainly attributed to its all-to-all collective communication, which fully utilizes all links and induces severe contention. Furthermore, in the H20 experiments, the observed speedup is slightly lower. This is due to the very high intra-node bandwidth (as reported in Table 2), which increases the CCR relative to other experiments. In other words, computation outweighs communication, limiting the benefits of the communication-computation overlap.

Memory Overhead Analysis. We evaluate memory overhead on L20 with 8 GPUs across varying sequence lengths for both forward and backward passes, comparing Ring-Attention with our method in Table 3. The results indicate that our method incurs only a modest increase in memory consumption. Furthermore, as the number of additional buffers introduced by our approach is constant (i.e., $O(1)$), memory overhead does not scale with the number of GPUs. See Section B.2 for detail analysis.

Numerical correctness verification. We evaluate the numerical correctness of CHARM+ by comparing its outputs and gradients against FlashAttention. The results show that CHARM+ exhibits numerical differences of the same order of magnitude as Ring-Attention across the output, log-sum-exp (LSE), and Q/K/V gradients, indicating that CHARM+ introduces no substantial additional numerical error. Detailed experimental configurations and results are provided in Section C.3.

4.3 Multi-node Evaluation

We conducted multi-node experiments on the H20 system with 8 or 16 GPUs using NVLink. On the 16-GPU setup, we compare our method against Ring-Attention, TASP, and their load-balanced variants, Ring-Zigzag and TASP-Zigzag. As shown in Figure 7, our method shows a clear advantage in cross-node configurations. This is because cross-node bandwidth is significantly lower than intra-node bandwidth, as listed in Table 2. Consequently, Ring-Attention and TASP incur GPU idle time due to frequent synchronized communication, resulting in a performance bottleneck that degrades efficiency.

Impact of Scaling on Performance. To evaluate scalability, we conducted experiments on the H20 system with 2 nodes using 8 GPUs and 2 nodes using 16 GPUs. As shown in Figure 7, our method achieves substantial speedups at larger scales by leveraging additional GPUs for better parallelism and effectively overlapping the communication with the computation.

Table 4: Ablation Study. We compare the forward performance of Ring-Attention, RM+, and CHARM+ across various GPU configurations and sequence lengths. Notably, CHARM+ will be evaluated only in cross-hardware configurations.

GPUs	Cross	Seq	Forward Time (ms)		
			Ring	RM+	Ours
4 × L20	✗	1792	**9.35**	10.29	–
		7168	**36.86**	39.66	–
8 × L20	✓	1792	17.61	25.91	**14.81**
		7168	69.85	98.49	**52.40**
4 × H20	✗	1792	5.38	**4.76**	–
		7168	38.62	**33.20**	–
8 × H20	✗	1792	5.51	**3.65**	–
		7168	26.79	**13.84**	–
16 × H20	✓	1792	32.17	39.04	**12.16**
		7168	84.58	98.98	**28.91**

4.4 Ablation Study

The effectiveness of the Re-Merge Multi-stream Mechanism (RM+). We evaluate RM+ across sequence lengths on a single 8-GPU H20 node to isolate its effectiveness from link contention and cross-node communication. As shown in Table 4, RM+ consistently outperforms Ring-Attention. This improvement stems from the parallel communication enabled by multiple independent q and kv streams. As a result, RM+ achieves a maximum speedup of $1.9\times$ over Ring-Attention.

The effectiveness of Cross-Hardware Query Parallelism. We compare CHARM+ against Ring-Attention and RM+ in both cross-NUMA and cross-node configurations. Specifically, for cross-

Table 5: Performance on H100. Note that TASP requires the sequence length to be divisible by $n(n-1)$. Therefore, the sequence lengths for the four-node and two-node settings differ slightly.

B	Seq	Forward				Backward			
		Ring	TASP	CHARM+	Speedup	Ring	TASP	CHARM+	Speedup
GPUs = 2 × 8									
16	26,880	149.97	98.01	**63.93**	**2.35**×	449.58	–	**141.52**	**3.18**×
16	13,440	72.65	49.27	**31.84**	**2.28**×	226.46	–	**65.35**	**3.47**×
64	6,720	141.75	97.65	**61.80**	**2.29**×	448.84	–	**123.32**	**3.64**×
128	3,360	140.55	97.15	**61.54**	**2.28**×	450.01	–	**121.99**	**3.69**×
GPUs = 4 × 8									
16	31,744	185.42	183.22	**68.17**	**2.72**×	559.34	–	**146.01**	**3.83**×
16	15,872	89.27	93.30	**34.64**	**2.58**×	279.85	–	**70.62**	**3.96**×
64	7,936	179.13	182.93	**66.48**	**2.69**×	556.25	–	**134.48**	**4.14**×
128	3,968	178.51	183.09	**66.43**	**2.69**×	560.63	–	**133.05**	**4.21**×

NUMA experiments, we use 8 GPUs on the L20 system, while for cross-node experiments, we use 16 GPUs on the H20 system. As shown in Table 4, RM+ suffers significant performance degradation in cross-hardware configurations, whereas CHARM+ achieves substantial improvements, with up to 1.9× and 3.5× speedup in cross-NUMA and cross-node configurations, respectively. The performance gain is smaller on L20 than on H20 due to link contention on L20, as discussed in the next subsection. The performance drop of RM+ is primarily due to a bandwidth imbalance between intra- and cross-hardware communication, as well as increased cross-node communication resulting from multistream parallelism. In contrast, CHARM+ decouples inter-node and intra-node communication to reduce synchronization overhead and improve overall performance.

The impact of link contention in PCIe architectures. To avoid communication synchronization issues caused by cross-NUMA communication on the L20 system, we conducted single-NUMA experiments on an L20 node with 4 GPUs, and performed comparison experiments on an H20 node with 4 GPUs connected via NVLink, which eliminates link contention.

4.5 Evaluation on High-end GPU

As shown in Table 4, on L20, RM+ suffers significant performance degradation due to link contention, performing worse than Ring-Attention. In contrast, RM+ exhibits much better performance on H20. We further conduct a comprehensive evaluation with various batch sizes and sequence lengths, summarized in Figure 8. The results indicate that link contention degrades performance in almost all configurations. The exception occurs when both the batch size and sequence length are small.

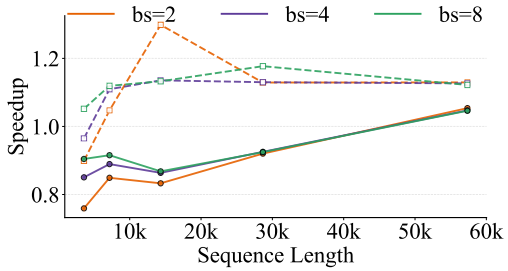


Figure 8: Performance of RM+ under PCIe and NVLink. We evaluate the forward time of RM+ on 4×H20 GPUs (dashed) and 4×L20 GPUs (solid) across different batch sizes (bs). Compared to Ring-Attention, performance on L20 is lower than on H20 due to the link contention.

In this case, the bandwidth is not fully utilized, so the link contention is negligible. Meanwhile, it is worth noting that in the 8× L20 experiments, link contention leads to smaller performance gains than on H20. Nevertheless, enabled by Cross-Hardware Query Parallelism, CHARM+ still significantly outperforms Ring-Attention and TASP, as shown in Figure 6a.

To completely validate our method on the high-end GPUs, we rented 32 H100 GPUs equipped with high-bandwidth NVLink. We conducted full forward and backward experiments under both 2×8 and 4×8 configurations, and the results are reported in Table 5 (detailed results can be found in Section C.2). When TASP begins to leverage high-performance GPUs, our method still maintains a leading performance, achieving an average speedup of 2× in the forward and 3× in the backward compared to ring-attention. Additionally, as described in Section 3.3, our method consistently makes

Table 6: End-to-end Performance Comparison of Ring-Attention and CHARM+ with Model Scaling

Batch Size	Seq Len	Model	Ring		CHARM+		Speedup
			Latency	Memory	Latency	Memory	
16	2,048	GPT-350M	10082.6	11,372	**4847.0**	11,518	**2.08**×
8	4,096	GPT-350M	10032.6	11,348	**4871.5**	11,544	**2.06**×
4	8,192	GPT-350M	9914.5	11,436	**5173.6**	11,552	**1.92**×
2	16,384	GPT-350M	10089.5	12,112	**5637.3**	12,170	**1.79**×
16	2,048	GPT-1.3B	21505.5	30,816	**11738.3**	31,602	**1.83**×
8	4,096	GPT-1.3B	22073.7	31,190	**11768.1**	31,546	**1.88**×
4	8,192	GPT-1.3B	21857.8	31,414	**12097.5**	32,030	**1.81**×
2	16,384	GPT-1.3B	22064.5	32,270	**12796.6**	32,312	**1.72**×

intermediate cross-node communication effectively overlapped by the intra-node pipeline. In contrast, other approaches suffer from synchronization constraints, making them susceptible to the slower cross-node communication. Consequently, our method exhibits greater performance as the scale increases (speedups of $2.28\times$/$3.69\times$ on 16 H100 GPUs and $2.69\times$/$4.21\times$ on 32 H100 GPUs at maximum performance), highlighting the advantage of our method.

4.6 End-to-end Training Performance

High-end GPU. We integrate our operator into Megatron-LM and compare it with Megatron's ring implementation. The experimental results are shown in Table 7. We use context parallelism (CP) for training on 16 high-end H20/H100 GPUs and report the average per-iteration time in milliseconds.

Table 7: End-to-end Performance Comparison of Ring-Attention and CHARM+ with High-end GPU

Batch Size	Seq Len	Model	Ring		CHARM+		Speedup	
			H20	H100	H20	H100	H20	H100
16	16,384	GPT3-350M	7304.6	3385.1	**5601.3**	**1925.7**	**1.30**×	**1.76**×
32	8,192	GPT3-350M	7193.7	3291.5	**4669.7**	**1679.2**	**1.54**×	**1.96**×
64	4,096	GPT3-350M	7169.6	3234.6	**4452.5**	**1538.6**	**1.61**×	**2.10**×
128	2,048	GPT3-350M	7143.6	3210.8	**4242.5**	**1482.0**	**1.68**×	**2.17**×

Notably, Megatron-LM includes extensive optimizations for ring attention that our method does not incorporate. Moreover, non-attention linear layers dominate end-to-end training, limiting the impact of distributed attention optimization. Despite this, our method still significantly outperforms ring-attention on H20 and H100, achieving max $1.68\times$ and $2.17\times$ speedup, respectively. Furthermore, our method performs better at larger batch sizes, making it more suitable for real-world workloads.

Model scaling. To evaluate the scalability of CHARM+ across model sizes, we conduct additional end-to-end experiments on 16 L20 GPUs with model sizes of 350M and 1.3B, measuring iteration latency (ms) and GPU memory consumption (MB). Although model configurations affect the relative computational costs of attention and MLP layers, they do not fundamentally change the communication patterns targeted by CHARM+. As shown in Table 6, CHARM+ achieves consistent speedups of $1.72\times$–$2.08\times$ across both model sizes and all tested configurations, with nearly identical memory overhead. These results demonstrate the effectiveness of CHARM+ across different model scales.

5 Conclusion

In this work, we present CHARM+, a novel approach that reorganizes the communication process of Ring-Attention using a merge-based strategy. Furthermore, it leverages a multistream parallelism of q and kv to reduce communication dependencies and decouples cross-hardware and intra-hardware communication. We implement both the forward and backward algorithms of the mechanism and conduct extensive experiments and ablation studies across various settings. Experiment results across diverse hardware platforms and GPUs with varying computational throughput demonstrate that CHARM+ consistently achieves significant performance improvements over state-of-the-art methods.

References

[1] Y. Bai, S. Tu, J. Zhang, H. Peng, X. Wang, X. Lv, S. Cao, J. Xu, L. Hou, Y. Dong, et al. Longbench v2: Towards deeper understanding and reasoning on realistic long-context multitasks. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3639–3664, 2025.

[2] W. Brandon, A. Nrusimha, K. Qian, Z. Ankner, T. Jin, Z. Song, and J. Ragan-Kelley. Striped attention: Faster ring attention for causal transformers. *arXiv preprint arXiv:2311.09431*, 2023.

[3] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[4] S. Chen, J. Chen, S. Zhu, Z. Jiang, Y. Peng, and X. Qian. Mesh-attention: A new communication-efficient distributed attention with improved data locality. *arXiv preprint arXiv:2512.20968*, 2025.

[5] Y. Chen, S. Qian, H. Tang, X. Lai, Z. Liu, S. Han, and J. Jia. Longlora: Efficient fine-tuning of long-context large language models. *arXiv preprint arXiv:2309.12307*, 2023.

[6] J. Choquette, E. Lee, R. Krashinsky, V. Balan, and B. Khailany. 3.2 the a100 datacenter gpu and ampere architecture. In *2021 IEEE International Solid-State Circuits Conference (ISSCC)*, volume 64, pages 48–50. IEEE, 2021.

[7] B. Citko. On the nature of merge: External merge, internal merge, and parallel merge. *Linguistic Inquiry*, 36(4):475–496, 2005.

[8] T. Dao, D. Fu, S. Ermon, A. Rudra, and C. Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in neural information processing systems*, 35:16344–16359, 2022.

[9] P. Doukhan, G. Oppenheim, and M. Taqqu. *Theory and applications of long-range dependence*. Springer Science & Business Media, 2002.

[10] V. Elango. Attention2d: Communication efficient distributed self-attention mechanism. *arXiv preprint arXiv:2503.15758*, 2025.

[11] A. Li, S. L. Song, J. Chen, J. Li, X. Liu, N. R. Tallent, and K. J. Barker. Evaluating modern gpu interconnect: Pcie, nvlink, nv-sli, nvswitch and gpudirect. *IEEE Transactions on Parallel and Distributed Systems*, 31(1):94–110, 2019.

[12] S. Li, F. Xue, C. Baranwal, Y. Li, and Y. You. Sequence parallelism: Long sequence training from system perspective. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2391–2404, 2023.

[13] T. Li, G. Zhang, Q. D. Do, X. Yue, and W. Chen. Long-context llms struggle with long in-context learning. *arXiv preprint arXiv:2404.02060*, 2024.

[14] X. Li. Niching without niching parameters: particle swarm optimization using a ring topology. *IEEE Transactions on Evolutionary Computation*, 14(1):150–169, 2009.

[15] Y. Li, X. Shen, X. Yao, X. Ding, Y. Miao, R. Krishnan, and R. Padman. Beyond single-turn: A survey on multi-turn interactions with large language models. *arXiv preprint arXiv:2504.04717*, 2025.

[16] H. Liu, M. Zaharia, and P. Abbeel. Ring attention with blockwise transformers for near-infinite context. *arXiv preprint arXiv:2310.01889*, 2023.

[17] P.-J. Lu, M.-C. Lai, and J.-S. Chang. A survey of high-performance interconnection networks in high-performance computer systems. *Electronics*, 11(9):1369, 2022.

[18] S. Lu, D. Guo, S. Ren, J. Huang, A. Svyatkovskiy, A. Blanco, C. Clement, D. Drain, D. Jiang, D. Tang, et al. Codexglue: A machine learning benchmark dataset for code understanding and generation. *arXiv preprint arXiv:2102.04664*, 2021.

[19] R. Neugebauer, G. Antichi, J. F. Zazo, Y. Audzevich, S. López-Buedo, and A. W. Moore. Understanding pcie performance for end host networking. In *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*, pages 327–341, 2018.

[20] P. O'Neil, E. Cheng, D. Gawlick, and E. O'Neil. The log-structured merge-tree (lsm-tree). *Acta informatica*, 33(4):351–385, 1996.

[21] D. A. Patterson. Latency lags bandwith. *Communications of the ACM*, 47(10):71–75, 2004.

[22] K. Punniyamurthy, K. Hamidouche, and B. M. Beckmann. Optimizing distributed ml communication with fused computation-collective operations. In *SC24: International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–17. IEEE, 2024.

[23] A. Sun, W. Zhao, X. Han, C. Yang, Z. Liu, C. Shi, and M. Sun. Burstattention: An efficient distributed attention framework for extremely long sequences. *arXiv preprint arXiv:2403.09347*, 2024.

[24] A. J. Thirunavukarasu, D. S. J. Ting, K. Elangovan, L. Gutierrez, T. F. Tan, and D. S. W. Ting. Large language models in medicine. *Nature medicine*, 29(8):1930–1940, 2023.

[25] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[26] Y. Wang, K. Hong, X. Li, Y. Xu, W. Wang, G. Dai, and Y. Wang. Tasp: Topology-aware sequence parallelism. *arXiv preprint arXiv:2509.26541*, 2025.

[27] G. Xiao, J. Tang, J. Zuo, J. Guo, S. Yang, H. Tang, Y. Fu, and S. Han. Duoattention: Efficient long-context llm inference with retrieval and streaming heads. *arXiv preprint arXiv:2410.10819*, 2024.

[28] L. Zhao, S. Pal, T. Chugh, W. Wang, J. Fantl, P. Basu, J. Khoury, and A. Krishnamurthy. Efficient {Direct-Connect} topologies for collective communications. In *22nd USENIX Symposium on Networked Systems Design and Implementation (NSDI 25)*, pages 705–737, 2025.

[29] W. X. Zhao, K. Zhou, J. Li, T. Tang, X. Wang, Y. Hou, Y. Min, B. Zhang, J. Zhang, Z. Dong, et al. A survey of large language models. *arXiv preprint arXiv:2303.18223*, 1(2), 2023.

Appendix

A Implementation

We implement both the forward and backward of CHARM+. Algorithm 1 presents the logic for constructing the communication groups used in our method. Specifically, the communication peers are partitioned into two disjoint subgroups corresponding to kv communication and q communication at each merge level. To avoid imbalanced communication overhead where multiple GPUs simultaneously initiate communication to the same GPU, we determine the order of kv and q communication based on each GPU's relative position within the global communication group. For example, when GPU 0 and 1 communicate with GPU 2 and 3, GPU 0 sends kv to GPU 2 and q to GPU 3, whereas GPU 1 sends q to GPU 2 and kv to GPU 3. This complementary scheduling ensures that both the kv and q processing streams on each GPU remain active, effectively preventing stream idling and improving overall communication efficiency.

Algorithm 1 GenerateCommList

Input: processGroup pg
Output: commList$_q$, commList$_{kv}$
Initialize $tmp \leftarrow [], n \leftarrow \text{len}(pg), x \leftarrow \text{getRank}(pg)$
$l \leftarrow 1$
while $l < n$ **do**
  $g \leftarrow \{x \oplus i \mid i \in [l, 2 \cdot l)\}$
  Append g to tmp
  $l \leftarrow 2 \cdot l$
end while
Initialize $g_1, g_2 \leftarrow []$, append $tmp[0]$ to g_1
$g_3 \leftarrow \text{Sort}(tmp[0] \cup \{x\})$
for each g in $tmp[1:]$ **do**
  first, second $\leftarrow$ split(g), index $\leftarrow$ getPosition(x, g_3)
  half $\leftarrow |g_3|/2$
  if index $<$ half **then**
    Append first to g_1, append second to g_2
  else
    index $\leftarrow$ index $-$ half
    Append second to g_1, append first to g_2
  end if
  $g_[3] \leftarrow \text{Sort}(g_3 \cup g)$
end for
commList$_{kv}$, commList$_q \leftarrow$ concatenate(g_1, g_2)
Return commList$_{kv}$, commList$_q$

Algorithm 2 provides the core logic chain of the forward algorithm [6], which is consistent with the overall pipeline in Figure 3(f). Notably, for the q communication, the attention output needs to be communicated back to the original GPU after the attention computation finishes (i.e., the *Out* communication in Figure 3). Therefore, we prioritize the q communication stream during execution, enabling earlier output communications and effective overlap with the subsequent computation of the kv communication stream.

B Analysis

We conduct a detailed analysis of the factors affecting CCR, as well as the memory usage of Ring-Attention and CHARM+.

[6] Since the forward and backward share the same overall logic, we omit the detailed description of the backward pass for brevity.

Algorithm 2 CHARM+ Forward

Input: tensor q, k, v, processGroup pg
Output: out
commList$_q$, commList$_{kv}$ $\leftarrow$ GenerateCommList(pg)
$q_{\text{cross}} \leftarrow$ CrossComm(q)
Initialize $out, out_{\text{cross}} \leftarrow None, n \leftarrow$ getsize(pg)
Commwait()
for $i \leftarrow 1$ **to** $\log(n)$ **do**
  $k_{\text{other}}, v_{\text{other}} \leftarrow$ KVComm(k, v)
  $q_{\text{other}}, q_{\text{otherC}} \leftarrow$ QComm(q, q_{cross})
  if $i == 1$ **then**
    $out \leftarrow$ CompAttention(q, k, v, out)
    $out_{\text{cross}} \leftarrow$ CompAttention($q_{\text{cross}}, k, v, out$)
  else
    $out_{\text{cross}} \leftarrow$ CompAttentionComm($q_{\text{otherC}}, k, v, out$)
    $out \leftarrow$ CompAttentionComm($q_{\text{other}}, k, v, out$)
    $out_{\text{cross}} \leftarrow$ CompAttention($q_{\text{cross}}, k_{\text{other}}, v_{\text{other}}, out$)
    $out \leftarrow$ CompAttention($q, k_{\text{other}}, v_{\text{other}}, out$)
  end if
  Commwait()
end for
$out \leftarrow$ CommAndUpdate(out, out_{cross})

B.1 Analysis of CCR

As mentioned in Eq. (7), CCR is defined by us as:

$$CCR = \sum_{i=1}^{N} T^i_{\text{comp}(q,k,v)} / \sum_{i=1}^{N} T^i_{\text{comm}(k,v)} \tag{8}$$

The communication time depends only on the amount of data being transferred and the available bandwidth. Different on-device and off-device connections have different bandwidths. Therefore, the relation between communication time and factors can be defined as:

$$T_{comm} \propto \frac{BSH}{Bandwidth} \tag{9}$$

And the computation time is essentially the time complexity of FlashAttention, which is

$$T_{comp} \propto \frac{BS^2H + BSH^2}{FLOPS} \tag{10}$$

Note that, while the theoretical time remains the relation above, the actual runtime is dominated by memory IO in standard attention and both the actual FLOPS and bandwidth exhibit some variability. It can be seen that once the input size and computation pattern are fixed, the time per step is constant. Thus, the total time equals the sum of non-overlappable times along the critical path, which means reducing the number of steps (minimizing the overall critical path) directly reduces the total time. In addition, we provide some quantitative results under different hardware and input specifications, as shown in Section C.2. Due to fluctuations in actual time, we collected the time data in real-time through the interface provided by CUDA events.

B.2 Analysis of Memory Overhead

Since we do not change the computation pattern but instead improve the overlap between computation and communication, we focus our analysis on the major memory-consuming buffers and ignore the memory overhead of the attention computation.

We assume that the input query, key, and value are all $[b, s, h]$ (i.e., batch size, sequence length and hidden dims). For Ring-Attention, the memory overhead of the buffers is:

$$M_{Ring} \approx 3bsh + 2bsh + 1.5bsh + 5bsh + 3bsh + 1.5bsh + 2bsh + 2bsh \approx 20bsh \tag{11}$$

Specifically, under FP32 precision, each buffer occupies BSH memory. In Ring-Attention, one buffer stores the local query block, two buffers hold the key and $value$ blocks being processed, and two more buffers receive the next KV blocks to overlap computation and communication. One buffer stores the accumulated $output$, and half a buffer holds intermediate FlashAttention results (usually in BF16/FP16). During backward, five buffers store q, k, v, the $output$, and its gradient; three buffers store total gradients of q, k, v; 1.5 buffers store FlashAttention gradients; and two buffers handle the next KV block computation. Finally, because KVs propagate across GPUs in a ring, the accumulated gradients must be sent back to the original GPU, requiring two additional buffers.

We further analyze the memory overhead of CHARM+, which can be formulated as:

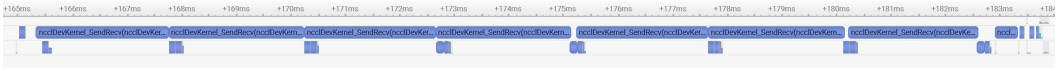
$$\begin{aligned} M_{\text{CHARMS+}} &\approx 3bsh + 2bsh + 1.5bsh + 5bsh + 3bsh + 1.5bsh + 2bsh + \mathbf{6bsh} - 2bsh \\ &\approx 24bsh \end{aligned} \tag{12}$$

During forward and backward, each GPU must retain its own KV and receive Q of cross-hardware, requiring three extra buffers. Moreover, FlashAttention results can be updated immediately, so the buffers used in ring-attention for gradient sending can be omitted.

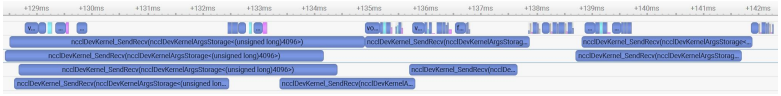
Overall, our improved method logically uses four more buffers than ring-attention. However, GPU memory is temporally reused across forward and backward and buffer lifetimes do not fully overlap, the peak buffer usage remains roughly the same, resulting in only a 3%~5% increase in observed memory.

C Experimental Details

C.1 Nsight Result



(a) Ring-Attention



(b) CHARM+

Figure 9: Nsight Analysis of Ring-Attention and CHARM+

To better understand the performance improvement, we profile the forward pass of Ring-Attention and CHARM+ on 8 L20 GPUs using Nsight Systems, as shown in Figure 9a and Figure 9b, respectively. The L20 platform has relatively limited communication bandwidth, making communication kernels substantially longer than computation kernels. This imbalance makes communication–computation overlap particularly important, as the overall execution time is largely determined by how effectively communication latency can be hidden behind computation.

In Figure 9a, Ring-Attention exhibits substantial GPU idle time due to its sequential communication dependencies. Each communication step depends on the completion of the preceding step, limiting scheduling flexibility and leaving computation unable to fully hide communication latency. In contrast, as shown in Figure 9b, CHARM+ reorganizes ring-based communication into a merge-based structure, shortening the dependency chain, and leverages multiple streams to execute independent communication and computation tasks concurrently. This design substantially reduces idle gaps and improves communication–computation overlap. Notably, both traces are plotted on the same time scale, clearly showing that CHARM+ completes execution in less time than Ring-Attention, demonstrating the effectiveness of CHARM+ in improving overall execution performance.

C.2 Evaluation on High-end GPU

We validate our method on the high-performance H100 GPUs, with detailed configurations provided in Table 2. We conduct both forward and backward experiments under 2×8 and 4×8 setups (with a

hidden dimensions of 12×192). We additionally conducted experiments on ultra-long sequences, specifically those exceeding 100k tokens. Table 9 reports the forward results for the 2×8 configuration, Table 10 reports the corresponding backward results, Table 11 presents the forward results for the 4×8 configuration, and Table 12 presents the corresponding backward results. We report the CCR as well as the detailed computation time (T_{comp}) and communication time (T_{comm}).

Our method shows better performance on 32 GPUs than on 16 GPUs. This is because the cross-node communication can be effectively overlapped with the intra-node pipeline. After decoupling via cross-hardware query parallelism, the only dependency is that cross-node query communication must finish before the intra-node pipeline begins. Accordingly, the query required for the next step can be communicated concurrently during the intra-node pipeline.

Based on this analysis and results, our method consistently confines cross-node communication to the start and end, with intermediate cross-node communication effectively overlapped by the intra-node pipeline. In contrast, other approaches suffer from synchronization constraints, making them susceptible to the slower cross-node communication. Consequently, our method exhibits greater performance as the scale increases (speedups of $2.28\times$/$3.69\times$ on 16 H100 GPUs and $2.69\times$/$4.21\times$ on 32 H100 GPUs at maximum performance), demonstrating our advantage.

In the scenario of ultra-long sequence lengths (over 200k tokens) on 16 H100 GPUs in Table 9 and Table 10, the CCR exceeds 1, indicating that the computation time is greater than the communication time. At this point, computation becomes the main bottleneck, diminishing the benefits of overlap of communication and computation. Nevertheless, our method still significantly outperforms all other methods.

C.3 Numerical correctness verification

In Ring-Attention, updates on each GPU occur in different orders, and the only precision loss comes from $\mathrm{BF16} \rightarrow \mathrm{FP32} \rightarrow \mathrm{BF16}$ conversion. This type of error exists in all distributed attention methods, and our method does not introduce any additional precision errors.

We verified numerical correctness of our method. The results were obtained on an 8-GPU L20 setup. Specifically, we set the sequence length to 8K and generated 100 groups of randomly initialized Q, K, V, and output gradients, all sampled from a normal distribution. We executed FlashAttention, Ring-Attention, and CHARM+, respectively.

By performing element-wise subtraction, we computed the difference matrices between Ring-Attention and FlashAttention, as well as between CHARM+ and FlashAttention. We further evaluated these differences by calculating the mean and maximum, and finally reported the averages over the 100 runs, as summarized in Table 8.

Table 8: Comparison of numerical differences between CHARM+ and Ring-Attention.

Diff	CHARM+ (Max)	CHARM+ (Mean)	Ring (Max)	Ring (Mean)
out	8.34147e-04	4.51229e-05	8.07648e-04	4.24147e-05
lse	1.65642e-06	2.94155e-07	1.75893e-06	3.04284e-07
dq	1.26290e-03	3.29077e-05	1.25977e-03	3.27659e-05
dk	2.87713e-03	3.53419e-05	2.89413e-03	3.53193e-05
dv	4.47177e-03	3.42940e-05	4.48120e-03	3.42894e-05

These results demonstrate that the numerical differences between CHARM+ and FlashAttention remain at the same order of magnitude as those between Ring-Attention and FlashAttention, indicating that CHARM+ does not introduce additional numerical errors.

Table 9: Forward performance comparison on 16-H100.

B	Seq	T_{comm}	T_{comp}	CCR	Ring	TASP	CHARM+	Speedup
Seq = 215,040								
1	215,040	81.49	86.01	1.06	116.76	86.93	**66.74**	**1.75**×
2	215,040	164.31	171.51	1.04	241.03	171.91	**125.43**	**1.92**×
Seq = 107,520								
1	107,520	38.34	23.16	0.60	44.08	26.36	**21.43**	**2.06**×
2	107,520	74.25	45.20	0.61	89.20	51.46	**42.28**	**2.11**×
4	107,520	146.17	88.94	0.61	182.15	101.74	**81.88**	**2.22**×
Seq = 53,760								
2	53,760	38.27	14.39	0.36	39.18	25.81	**18.25**	**2.15**×
4	53,760	74.39	26.50	0.36	78.95	50.29	**34.36**	**2.30**×
8	53,760	146.24	51.13	0.35	156.57	99.39	**66.91**	**2.34**×
Seq = 26,880								
2	26,880	20.13	5.63	0.28	18.35	13.56	**9.43**	**1.95**×
4	26,880	38.24	9.87	0.26	37.90	25.64	**17.03**	**2.22**×
8	26,880	74.44	17.22	0.23	75.12	49.75	**32.60**	**2.30**×
16	26,880	146.53	32.05	0.22	149.97	98.01	**63.93**	**2.35**×
Seq = 13,440								
2	13,440	10.89	2.92	0.27	9.15	8.18	**5.89**	**1.55**×
4	13,440	20.31	4.10	0.20	17.89	13.56	**9.07**	**1.97**×
8	13,440	38.31	6.87	0.18	35.72	25.48	**16.57**	**2.16**×
16	13,440	74.55	11.83	0.16	72.65	49.27	**31.84**	**2.28**×
32	13,440	145.95	21.79	0.15	143.05	97.30	**62.34**	**2.29**×
Seq = 6,720								
4	6,720	10.92	2.53	0.23	9.12	8.19	**5.63**	**1.62**×
8	6,720	20.12	3.27	0.16	17.59	13.47	**8.91**	**1.98**×
16	6,720	38.26	4.61	0.14	35.76	25.27	**16.31**	**2.19**×
32	6,720	74.53	7.90	0.12	70.84	49.23	**31.53**	**2.25**×
64	6,720	146.14	17.05	0.11	141.75	97.65	**61.80**	**2.29**×
Seq = 3,360								
8	3,360	10.98	2.55	0.25	9.17	8.10	**5.62**	**1.63**×
16	3,360	20.17	3.05	0.15	17.49	13.37	**8.82**	**1.98**×
32	3,360	38.23	4.61	0.12	34.50	25.13	**16.23**	**2.13**×
64	3,360	74.53	7.90	0.11	70.54	49.06	**31.21**	**2.26**×
128	3,360	146.20	14.36	0.10	140.55	97.15	**61.54**	**2.28**×

Table 10: Backward performance comparison on 16-H100.

B	Seq	T_{comm}	T_{comp}	CCR	Ring	TASP	CHARM+	Speedup
Seq = 215,040								
1	215,040	218.48	257.76	1.18	312.45	–	**207.07**	**1.51**×
2	215,040	437.82	503.06	1.15	628.21	–	**405.90**	**1.55**×
Seq = 107,520								
1	107,520	109.45	68.03	0.62	114.64	–	**64.09**	**1.79**×
2	107,520	218.37	135.44	0.62	228.70	–	**125.01**	**1.83**×
4	107,520	436.10	264.15	0.61	451.53	–	**242.15**	**1.86**×
Seq = 53,760								
2	53,760	110.04	37.64	0.34	113.30	–	**45.91**	**2.47**×
4	53,760	219.95	74.53	0.34	226.22	–	**88.55**	**2.55**×
8	53,760	439.16	145.33	0.33	450.33	–	**173.18**	**2.60**×
Seq = 26,880								
2	26,880	56.73	11.47	0.20	57.03	–	**20.42**	**2.79**×
4	26,880	112.12	22.35	0.20	112.24	–	**37.51**	**2.99**×
8	26,880	221.73	43.86	0.20	226.13	–	**72.59**	**3.12**×
16	26,880	442.86	86.72	0.20	449.58	–	**141.52**	**3.18**×
Seq = 13,440								
2	13,440	28.73	4.50	0.16	28.93	–	**10.81**	**2.67**×
4	13,440	56.86	8.42	0.15	57.08	–	**18.48**	**3.09**×
8	13,440	112.90	15.44	0.14	113.19	–	**33.84**	**3.34**×
16	13,440	222.95	29.43	0.13	226.46	–	**65.35**	**3.47**×
32	13,440	444.64	57.34	0.13	448.54	–	**128.53**	**3.49**×
Seq = 6,720								
4	6,720	28.88	3.23	0.11	29.11	–	**10.22**	**2.85**×
8	6,720	56.98	5.94	0.10	56.67	–	**17.34**	**3.27**×
16	6,720	112.98	10.90	0.10	112.72	–	**32.38**	**3.48**×
32	6,720	224.56	20.72	0.09	224.23	–	**62.46**	**3.59**×
64	6,720	445.33	40.32	0.09	448.84	–	**123.32**	**3.64**×
Seq = 3,360								
8	3,360	28.72	2.74	0.10	28.11	–	**10.04**	**2.80**×
16	3,360	56.94	4.98	0.09	56.74	–	**17.06**	**3.33**×
32	3,360	112.99	9.41	0.08	112.68	–	**32.14**	**3.51**×
64	3,360	224.63	18.13	0.08	224.41	–	**62.20**	**3.61**×
128	3,360	445.33	35.34	0.08	450.01	–	**121.99**	**3.69**×

Table 11: Forward performance comparison on 32-H100.

B	Seq	T_{comm}	T_{comp}	CCR	Ring	TASP	CHARM+	Speedup
Seq = 126,976								
2	126,976	95.28	36.30	0.38	102.34	94.17	**39.95**	**2.56**×
4	126,976	186.17	70.03	0.38	208.12	184.59	**77.02**	**2.70**×
Seq = 63,488								
2	63,488	49.95	12.50	0.25	46.32	48.54	**19.80**	**2.34**×
4	63,488	96.13	22.28	0.23	95.79	93.60	**36.62**	**2.62**×
8	63,488	187.15	41.96	0.22	193.04	184.12	**70.62**	**2.73**×
Seq = 31,744								
2	31,744	26.70	6.28	0.24	22.59	25.38	**11.47**	**1.97**×
4	31,744	50.19	8.99	0.18	45.25	48.38	**18.91**	**2.39**×
8	31,744	96.07	15.58	0.16	92.39	93.35	**35.27**	**2.62**×
16	31,744	187.54	28.46	0.15	185.42	183.22	**68.17**	**2.72**×
Seq = 15,872								
2	15,872	15.54	4.98	0.32	12.06	14.09	**9.19**	**1.31**×
4	15,872	26.73	5.09	0.19	22.29	25.24	**10.93**	**2.04**×
8	15,872	50.16	7.13	0.14	44.77	48.23	**18.47**	**2.42**×
16	15,872	96.20	11.93	0.12	89.27	93.30	**34.64**	**2.58**×
32	15,872	186.74	21.26	0.11	180.66	183.29	**66.85**	**2.70**×
Seq = 7,936								
4	7,936	15.63	4.96	0.32	12.02	14.09	**8.97**	**1.34**×
8	7,936	27.24	5.01	0.18	22.31	25.31	**10.83**	**2.06**×
16	7,936	50.01	6.33	0.13	43.95	48.34	**18.34**	**2.40**×
32	7,936	96.10	10.17	0.11	89.66	93.28	**34.44**	**2.60**×
64	7,936	186.69	17.81	0.10	179.13	182.93	**66.48**	**2.69**×
Seq = 3,968								
8	3,968	15.55	4.71	0.30	11.91	14.08	**8.69**	**1.37**×
16	3,968	27.24	5.12	0.19	22.29	25.23	**10.90**	**2.04**×
32	3,968	50.60	6.31	0.12	44.73	48.17	**18.39**	**2.43**×
64	3,968	96.26	9.56	0.10	89.71	93.11	**34.44**	**2.60**×
128	3,968	189.89	16.43	0.09	178.51	183.09	**66.43**	**2.69**×

Table 12: Backward performance comparison on 32-H100.

B	Seq	T_{comm}	T_{comp}	CCR	Ring	TASP	CHARM+	Speedup
Seq = 126,976								
2	126,976	259.62	102.57	0.40	280.48	–	**111.60**	**2.51**×
4	126,976	521.26	196.56	0.38	559.74	–	**213.85**	**2.62**×
Seq = 63,488								
2	63,488	138.65	29.95	0.22	141.18	–	**46.30**	**3.05**×
4	63,488	269.38	58.46	0.22	280.96	–	**86.70**	**3.24**×
8	63,488	529.05	111.96	0.21	560.30	–	**166.44**	**3.37**×
Seq = 31,744								
2	31,744	70.47	10.16	0.14	70.57	–	**24.09**	**2.93**×
4	31,744	140.16	19.39	0.14	141.71	–	**40.55**	**3.49**×
8	31,744	275.78	37.57	0.14	280.92	–	**75.16**	**3.74**×
16	31,744	536.42	73.45	0.14	559.34	–	**146.01**	**3.83**×
Seq = 15,872								
2	15,872	35.78	4.67	0.13	35.77	–	**14.92**	**2.40**×
4	15,872	70.70	7.28	0.10	71.35	–	**22.11**	**3.23**×
8	15,872	140.84	13.69	0.10	141.53	–	**37.75**	**3.75**×
16	15,872	277.48	26.20	0.09	279.85	–	**70.62**	**3.96**×
32	15,872	548.62	50.62	0.09	558.53	–	**137.84**	**4.05**×
Seq = 7,936								
4	7,936	35.82	3.71	0.10	35.18	–	**14.51**	**2.42**×
8	7,936	70.72	5.82	0.08	71.06	–	**21.27**	**3.34**×
16	7,936	140.74	10.90	0.08	141.24	–	**36.61**	**3.86**×
32	7,936	278.53	20.68	0.07	278.38	–	**69.02**	**4.03**×
64	7,936	550.51	39.76	0.07	556.25	–	**134.48**	**4.14**×
Seq = 3,968								
8	3,968	35.79	3.24	0.09	35.86	–	**13.97**	**2.57**×
16	3,968	70.73	5.16	0.07	71.33	–	**20.95**	**3.41**×
32	3,968	140.73	9.70	0.07	141.84	–	**36.35**	**3.90**×
64	3,968	278.31	18.40	0.07	279.62	–	**68.40**	**4.09**×
128	3,968	551.62	35.58	0.06	560.63	–	**133.05**	**4.21**×

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: The abstract and introduction accurately reflect our paper's contributions and scope, and the claimed improvements are consistent with the theoretical design and experimental results.

   Guidelines:

   - The answer [N/A] 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 [N/A] 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: We discuss the limitations in the main text and provide experimental evidence, with further detailed analysis included in the appendix for clarification and completeness.

   Guidelines:

   - The answer [N/A] 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: [Yes]

Justification: All assumptions and full theorem statements are in the paper.

Guidelines:

- The answer [N/A] 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 provide detailed experimental setups and results in the main text, with additional results and supplementary experiments included in the appendix. The implementation code has also been submitted as supplementary material to facilitate reproduction of the main results.

Guidelines:

- The answer [N/A] 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: [Yes]

   Justification: We provide the code and relevant data in the supplementary materials.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.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://neurips.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) necessary to understand the results?

   Answer: [Yes]

   Justification: We provide detailed descriptions in the experimental section of the main text and in the Experimental Details section of the appendix.

   Guidelines:

   - The answer [N/A] 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: [Yes]

   Justification: During the experiments, we performed warm-up steps to avoid measurement bias and conducted extensive repeated runs for each result and reported the averages. A detailed description is provided in the experimental setup section.

   Guidelines:

   - The answer [N/A] 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: We provide the information in the experimental setup section.

   Guidelines:

   - The answer [N/A] 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: The research conducted in this paper conforms to the NeurIPS Code of Ethics in all respects.

   Guidelines:

   - The answer [N/A] 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: [N/A]

    Justification: We focus on algorithmic and system-level architectural research, and there is no societal impact of the work performed.

    Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] 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., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper poses no such risks.

    Guidelines:

    - The answer [N/A] 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: The creators or original owners of assets (e.g., code, data, models) used in the paper are properly credited. The license and terms of use are explicitly mentioned and properly respected.

    Guidelines:

    - The answer [N/A] 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: [N/A]

    Justification: The paper does not release new assets.

    Guidelines:

    - The answer [N/A] 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: [N/A]

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

    Guidelines:

    - The answer [N/A] 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: [N/A]

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

    Guidelines:

    - The answer [N/A] 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 rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: The core method development in this research does not involve LLMs as any important, original, or non-standard components.

    Guidelines:

    - The answer [N/A] 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 in the NeurIPS handbook for what should or should not be described.