SPARSEBALANCE: LOAD-BALANCED LONG CONTEXT TRAINING WITH DYNAMIC SPARSE ATTENTION

Anonymous authors
Paper under double-blind review

ABSTRACT

While sparse attention mitigates the computational bottleneck of long-context LLM training, its distributed training process exhibits extreme heterogeneity in both *1)* sequence length and *2)* sparsity sensitivity, leading to a severe imbalance problem and sub-optimal model accuracy. Existing algorithms and training frameworks typically focus on single issue, failing to systematically co-optimize these two problems. Therefore, we propose SparseBalance, a novel algorithm-system co-design framework, which exploits the sparsity and sequence heterogeneity to optimize model accuracy and system efficiency jointly. First, we propose workload-aware dynamic sparsity tuning, which employs a bidirectional sparsity adjustment to eliminate stragglers and exploit inherent bubbles for free accuracy. Second, we propose a sparsity-aware batching strategy to achieve coarse-grained balance, which complements dynamic sparsity tuning. Experimental results demonstrate that SparseBalance achieves up to a 1.33× end-to-end speedup while still improving the long-context capability by 0.46% on the LongBench benchmark.

1 INTRODUCTION

The long-context modeling capability is increasingly crucial for the evolution of Large Language Models (LLMs), serving as the backbone for advanced downstream applications such as code generation, in-depth reasoning, and autonomous agent systems. Mainstream LLMs typically perform an additional training stage on specific long-context datasets to extend the context windows Team et al. (2024); GLM et al. (2024); Qwen et al. (2024); Yang et al. (2025). However, as the context length increases, the standard attention mechanism becomes the primary computational bottleneck. Because standard attention requires computing affinities across all token pairs, it exhibits a quadratic computational complexity with respect to sequence length, thus severely hindering the context expansion. To mitigate this, sparse attention has emerged as a promising solution in recent released LLMs DeepSeek-AI et al. (2025); GLM-5-Team et al. (2026). By selectively computing the critical tokens, sparse attention reduces the computational complexity and breaks the performance bottleneck.

However, long-context sparse training still faces a severe load imbalance problem caused by the inherent heterogeneity in sequence length distributions. For instance, the Qwen2.5 technical report Qwen et al. (2024) discloses their data mix strategy at long context training stage, which comprises 40% long sequences and 60% short sequences. This heterogeneity in sequence length leads to severe load imbalance problem in distributed training, significantly degrading the system efficiency. Existing works attempt to alleviate this issue through batching or packing strategies. However, these methods can hardly solve the problem completely due to the inherently discrete nature in this bin-packing problem, especially for highly skewed datasets. This imbalance manifests across multiple parallelism dimensions in distributed training. In inner-level pipeline parallelism (PP), varying workloads exacerbate pipeline bubbles, and the processing time of extremely long sequences essentially dominates the overall pipeline latency. Worse still, this imbalance in PP ultimately propagates to the outer-level parallelism like data parallelism (DP), severely amplifying the degradation of system efficiency.

Meanwhile, sparse training introduces another critical heterogeneity in sparsity sensitivity. Although existing trainable sparse attention algorithms typically adopt a predefined sparsity degree, ***we observe distinct sparsity sensitivities across different sequences and transformer layers***. Furthermore, if simply transitioned to dynamic sparse training without system awareness, this inherent sparsity heterogeneity would lead to the exact same workload imbalance problem. Therefore, these two

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

dimensions of heterogeneity—sequence length and sparsity sensitivity—are deeply intertwined and jointly impact the runtime workload, highlighting the critical need for algorithm-system co-design. However, current algorithms and training frameworks typically address these challenges in isolation, solely focusing on workload balance or developing sophisticated sparse algorithms. Consequently, they fail to systematically co-optimize these two issues, resulting in sub-optimal performance in either training efficiency or model accuracy.

To address sequence length heterogeneity and sparsity sensitivity heterogeneity simultaneously, we propose SparseBalance, a novel algorithm–system co-design enabling bidirectional sparsity tuning for LLM training. SparseBalance jointly optimizes model accuracy and system efficiency while considering their inherent trade-off.

First, we propose a workload-aware **D**ynamic **S**parsity **T**uning (DST) strategy, which performs bidirectional sparsity adjustment at runtime to rebalance the workload at the layer level. Specifically, DST identifies bottleneck micro-batches and reduces their attention budget to accelerate execution, while simultaneously increasing the attention budget of non-bottleneck micro-batches to exploit pipeline bubbles for free accuracy improvements. To ensure the efficiency and accuracy during such tuning process, we introduce an anchor-guided thresholding mechanism, which determines the direction and bounds the magnitude for each micro-batch.

Second, to fully unleash the optimization potential of DST, we propose a **S**parsity-**A**ware **B**atching (SAB) strategy, which involves lightweight sparsity estimation and latency-based data packing. SAB provides a well-balanced initial workload distribution, which serves as a foundation for the fine-grained runtime adjustments in DST. Additionally, we implement the latency prediction module, which maps the sequence length and sparsity to practical execution latency through offline profiling, providing the accurate performance guide for both DST and SAB module. Together, SAB and DST form a unified optimization pipeline, spanning from coarse-grained data reorganization to fine-grained runtime tuning, enabling efficient load balancing without sacrificing model accuracy.

We evaluate SparseBalance on two real-world datasets and conduct comprehensive evaluations across three downstream benchmarks. Our key contributions are as follows:

- We provide a novel perspective on improving sparse attention training through algorithm-system co-design to jointly optimize system efficiency and model accuracy.
- We propose workload-aware dynamic sparsity tuning (DST) to dynamically rebalance the training workload at runtime, while preserving model accuracy.
- We propose sparsity-aware batching (SAB) dedicated for sparse training scenario, it involves a lightweight sparsity estimator and latency-based batching strategy, providing coarse-grained balance for DST.
- We implement SparseBalance and experimental results demonstrate that SparseBalance improves the end-to-end training efficiency by up to 1.33× while still improving the model's downstream capability.

2 Observation

2.1 Sequence Length Heterogeneity in Datasets

In long context training, the sequence length distribution could varying significantly. To illustrate this, we analyze two real-world, open-source datasets: nvidia/ChatQA2-Long-SFT-data (ChatQA2) NVIDIA (2025) and zai-org/LongAlign-10k Bai et al. (2024). As shown in Figure 1, both datasets exhibit highly skewed length distributions. ChatQA2 demonstrates a bimodal pattern, dominated by ultra-short (<4K) and long (>16K) sequences. In contrast, LongAlign-10k presents a pronounced long-tail distribution, with sequence lengths stretching continuously from 8K up to 72K tokens. Consequently, this extreme heterogeneity in sequence lengths leads to distinct runtime workloads across distributed workers, highlighting the critical need for a workload-aware solution that can dynamically balance workloads.

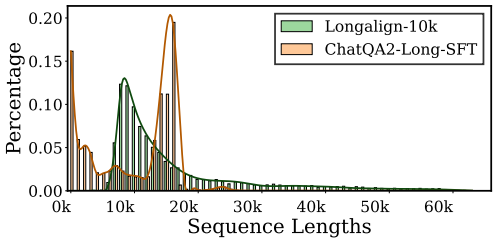

Figure 1: Extreme heterogeneity in sequence length distributions across two real-world long-context datasets. The x-axis represents the sequence length, and the y-axis denotes its corresponding proportion within the entire dataset.

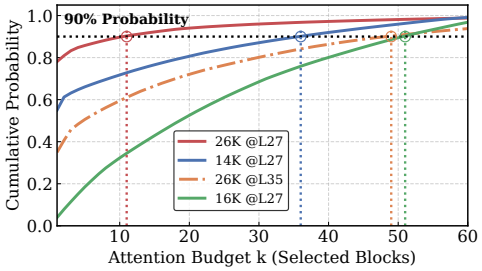

Figure 2: Cumulative score coverage under different attention budgets.

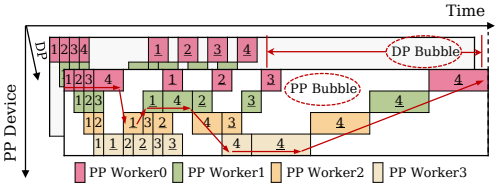

Figure 3: Illustration of the straggler effect caused by workload imbalance. Within PP group, a bottleneck micro-batch (e.g., Micro-batch 4) dictates the critical path and exacerbates pipeline bubbles. Furthermore, DP synchronization significantly amplifies this imbalance across the entire system.

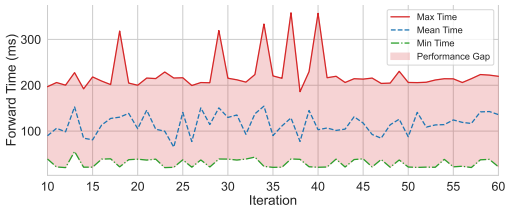

Figure 4: Forward-pass latency of micro-batches over continuous training iterations. The substantial gap between the maximum (T_{max}, red solid line), mean (T_{mean}, blue dashed line), and minimum (T_{min}, green dashed line) latencies reveals severe workload imbalance.

2.2 Inter- and Intra-Sequence Sparsity Heterogeneity

Beyond sequence length heterogeneity, sparse training also exhibits substantial ***heterogeneity in sparsity sensitivity***, at both the inter-sequence and intra-sequence levels. To quantify this effect, we profile the cumulative probability of attention score in our realistic training scenario. Figure 2 plots the recovery ratio under different attention budgets k (equivalently, the number of selected critical blocks (TopK), more budgets implies less sparsity degree). We report three sequences of lengths 14K, 16K, and 26K at layer 27, and additionally plot the same 26K sequence at layers 27 and 35.

First, The results reveal clearly distinct sparsity sensitivities across sequences, which we refer to ***inter-sequence sparsity heterogeneity***. For instance, the red curve in Figure 2 exhibits a highly concentrated attention pattern, where a minimal budget (about $k \approx 10$) is sufficient to recover 90% of the original attention representation. By contrast, the blue and green curves demonstrate a dispersed attention distribution, requiring a substantially larger budgets (about $k \approx 36$ and 50) to achieve the same recovery ratio.

Second, the same sequence can exhibit very different sparsity patterns across layers, which we term ***intra-sequence sparsity heterogeneity***. For the same 26K sample shown in Figure 2, layer 27 (red) remains sharply concentrated, whereas layer 35 (orange) is much flatter and more sensitive to aggressive budget reduction.

Taken together, these observations reveal substantial potential for dynamic sparsity tuning and directly motivates the layer-level dynamic sparsity tuning design in SparseBalance.

2.3 Imbalance Problem in Distributed Training

The heterogeneous sequence length distributions directly degrade the distributed LLM training, leading to severe load imbalance problem. The parallel training strategies always operate in a

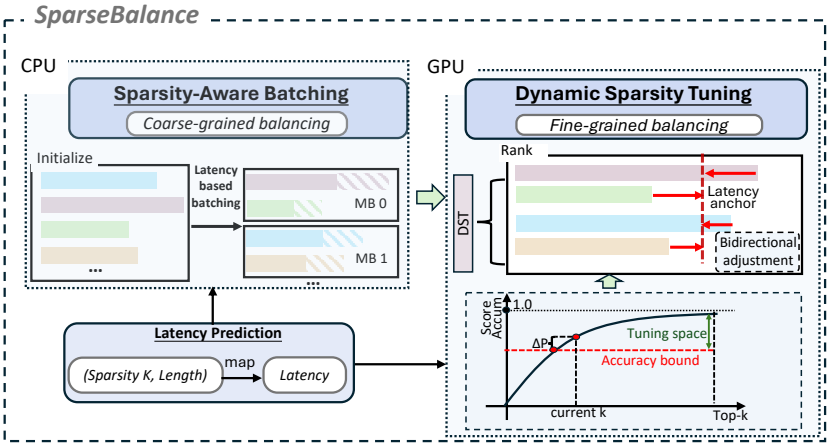

Figure 5: Overview of SparseBalance, which consists of three components: workload-aware dynamic sparsity tuning (DST), sparsity-aware batching (SAB) and latency prediction module.

hierarchical and synchronized manner. The longest micro-batch inevitably bottlenecks the entire system. From the perspective of pipeline parallelism (PP), stages process micro-batches sequentially and rely on synchronous point-to-point communication to pass middle states. As illustrated in Figure 3, the critical path of the PP execution time can be formulated as $T_{pp} \geq T_{max} \times PP_Size + \sum T_{other}$. Therefore, the longest micro-batch strictly bounds the overall pipeline throughput, causing severe pipeline bubbles. Data parallelism (DP) also suffers from the straggler effect due to the requirement of collective gradient synchronization at the end of each iteration. Worse still, the imbalance will be amplified when DP and PP are coupled, which is a standard configuration in large-scale LLM training. Consequently, the longest micro-batch fundamentally dominates the global step time, severely impacting the end-to-end training latency.

To empirically quantify this phenomenon, Figure 4 visualizes the forward-pass latency of various micro-batches over 50 continuous training iterations. Specifically, the red solid line represents the maximum latency (T_{max}), the blue dashed line indicates the mean latency (T_{mean}), and the green dashed line denotes the minimum latency (T_{min}). The substantial performance gap between the maximum and minimum latencies indicates a massive optimization potential. While existing works resort to more balanced batching strategies, they cannot perfectly eliminate the straggler effect due to the discrete nature of the batching problem, especially for highly skewed data distributions, highlighting the critical need for a more fundamental approach.

3 Design

Figure 5 illustrates the overall design of SparseBalance. From the execution perspective, SparseBalance balances the workload in a coarse-to-fine manner. SAB first reorganizes training samples using sparsity estimation and latency-based packing, achieving coarse-grained workload balance across micro-batches. Built on top of this, DST further conducts per-layer runtime sparsity tuning to mitigate residual imbalance at a finer granularity while preserving model quality. Both SAB and DST rely on accurate latency prediction as a performance guidance for optimization. To this end, we design a latency prediction module that maps sequence length and sparsity degree to practical execution latency under specific hardware and training configurations through offline profiling manner. For clarity, we first present DST in Section 3.1, then introduce the latency prediction module in Section 3.2, and finally describe the design of SAB in Section 3.3.

3.1 Workload-Aware Dynamic Sparsity Tuning

Existing methods for long-context training typically rely on batching or packing strategies to alleviate the imbalance problem induced by the heterogeneity in sequence length. However, due to the inherently discrete nature of batch construction, these approaches cannot fundamentally eliminate

the imbalance, especially under highly skewed sequence-length distributions. Even with careful data shuffling, a single micro-batch containing a difficult-to-pack sequence may still become a straggler on the critical path, delaying the entire iteration. Therefore, a finer-grained load balancing mechanism is required to mitigate this problem.

Fortunately, sparse training provides such an opportunity for finer-grained load balancing. As discussed in Section 2.2, long-context sparse training exhibits another important heterogeneity in sparsity sensitivity. This observation allows us to treat the sparsity (attention budget) not merely as a fixed algorithmic hyper-parameter, but as an adaptive degree of freedom for runtime load balancing. Based on this insight, we propose ***Workload-Aware Dynamic Sparsity Tuning (DST)***, which dynamically redistributes the attention budget across micro-batches according to their impact on the critical path.

At a high level, DST performs *bidirectional* budget adjustment. For bottleneck micro-batches on the critical path, DST reduces their attention budget, i.e., tunes them toward higher sparsity, to shorten their execution time and eliminate stragglers. Even a small reduction for these bottlenecks can improve the end-to-end step latency, since their delay is amplified by synchronization and dependencies in distributed training. In contrast, for non-bottleneck micro-batches whose latency does not affect the overall step time, DST allocates more attention budget, i.e., tunes them toward lower sparsity, to improve the fidelity of sparse attention. Although such adjustment may slightly increase local execution time, the additional cost can be absorbed by the inevitable idle time introduced by pipeline execution and synchronization. In this sense, DST converts this otherwise wasted time into *free accuracy*.

To make this bidirectional tuning both effective and safe, we design an ***anchor-guided thresholding mechanism***. For a global batch containing N micro-batches, let k_i^{base} denote the attention budget assigned by the underlying sparse attention method. Using the latency predictor introduced in Section 3.2, we first estimate the latency of each micro-batch under its base budget, and then define an execution anchor as

$$T_{\text{anchor}} = \phi\left(\left\{\hat{T}_i(k_i^{base})\right\}_{i=1}^{N}\right) \tag{1}$$

where $\phi(\cdot)$ is an anchor operator, such as the minimum, mean, or maximum of the predicted micro-batch latencies. DST then aligns each micro-batch to this reference point and obtains the corresponding target budget k_i^{anchor}. Therefore, the anchor determines the tuning direction: micro-batches above the anchor are compressed, whereas those below the anchor can use more attention budget.

To preserve model quality, DST further constrains the tuning magnitude using the routing logits already produced by the sparse attention indexer, introducing almost no additional runtime overhead. Specifically, we normalize and sort the routing logits of micro-batch i to obtain importance scores $r_{i,1} \geq r_{i,2} \geq \cdots$, where $\sum_j r_{i,j} = 1$. We then define the cumulative score coverage under budget k as

$$C_i(k) = \sum_{j=1}^{k} r_{i,j} \tag{2}$$

$C_i(k)$ characterizes how well the selected sparse pattern preserves the high-importance regions indicated by the routing scores. For bottleneck micro-batches, reducing the budget from k_i^{base} to k_i^{anchor} may hurt model quality. We therefore bound the allowable coverage drop by a threshold p:

$$C_i(k_i^{base}) - C_i(k_i^{anchor}) \leq p \tag{3}$$

Here, p serves as the primary knob controlling the aggressiveness of dynamic sparsity tuning: a larger p permits more aggressive compression of bottleneck micro-batches and thus favors system efficiency, while a smaller p enforces stricter fidelity to the original sparse attention pattern.

In this way, the anchor determines the tuning direction and target balance point, while the threshold p controls the safe tuning magnitude. As a result, DST shortens the critical path by compressing bottleneck micro-batches and simultaneously improves the computation fidelity of non-bottleneck micro-batches, achieving better workload balance without sacrificing model quality.

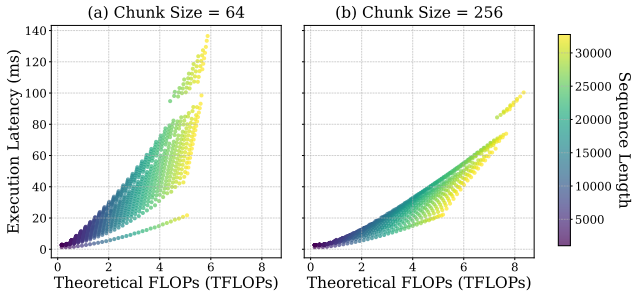

Figure 6: Practical latency of sparse attention cannot be reliably predicted by either theoretical FLOPs or sequence length alone, motivating our profiling-based latency predictor.

3.2 Profiling-Based Latency Modeling

The effectiveness of DST relies on accurate latency guidance. In particular, DST must estimate not only the execution latency of a micro-batch under its current attention budget, but also the budget required to align its latency to a target anchor. Therefore, we design a lightweight latency prediction module that maps sequence length and attention budget to practical execution latency under a given hardware environment, parallelization strategy, and sparse attention implementation.

However, building an accurate latency predictor for sparse attention is non-trivial. For standard attention, sequence length or theoretical FLOPs is often a reasonable proxy for workload. In sparse training, however, this approximation becomes unreliable, because the practical latency is jointly affected by sequence length, attention budget, sparse kernel implementation, and hardware characteristics. To empirically validate this mismatch, we profile the latency of one Transformer layer under different settings using the official implementation of MoBA. As shown in Figure 6, neither FLOPs-based nor length-based modeling can consistently predict the practical latency of sparse attention. Therefore, we adopt a profiling-based design instead of an analytical one.

3.3 Sparsity-Aware Batching

While DST provides fine-grained runtime balancing, it still benefits from a reasonably balanced initial workload distribution. Workload balanced batching strategy can provide this initial balance and enable smoother runtime sparsity tuning in DST. However, existing batching or packing strategies typically rely on sequence length or theoretical FLOPs as the workload partition metric Xu et al. (2025a); Wang et al. (2025), which is often inaccurate in sparse attention scenario as discussed in Section 3.2. Moreover, batching is executed on the CPU side before the GPU-side DST module, while the final sparsity is determined dynamically in the DST module. Therefore, the accurate workload estimation remains difficult.

To address the above issues, we implement ***Sparsity-Aware Batching*** (SAB), a batching strategy tailored to dynamic sparse training, which involves lightweight sparsity prediction and latency-based workload partition. First, we use the base sparsity as an initial estimate and periodically calibrate it using runtime statistics collected from DST. Concretely, for each sequence-length bin, we maintain an exponential moving average of the final attention budgets produced by DST, and periodically refresh the sparsity estimate used by SAB. Although this approximation is lightweight, it is sufficient to provide a more workload-aware batching signal in practice.

Based on the predicted sparsity, we reuse `Perf.predict` as a sample-level workload proxy to guide the batching process. As shown in Algorithm 2, SAB first estimates the workload weight of each sample through sparsity prediction and latency prediction. On top of that, SAB implements a two-level workload partition, which first balances the workload across DP groups and then forms micro-batches within each group. Here, `bin_packing` denotes a standard balanced partition solver that groups samples according to the provided workload weights. The output of SAB is a set of micro-batch index lists, which are then consumed by the dataloader to materialize the actual training batches and transfer to GPU. Since this reorganization is strictly confined within the global batch, it

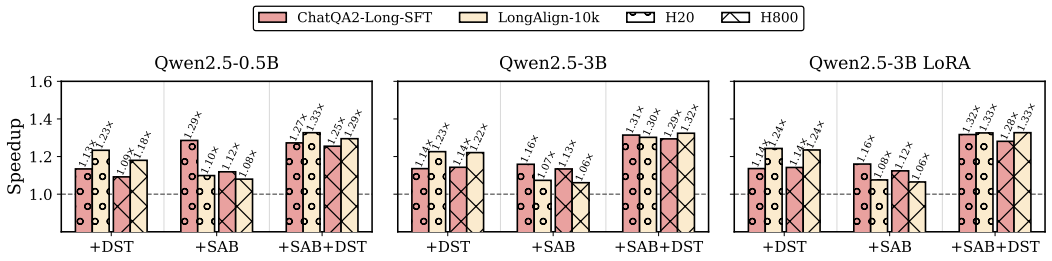

Figure 7: Overall performance and step-by-step speedups on two clusters and two datasets. We evaluate the normalized speedup ratios (measured by average iteration time) of SparseBalance and its two main components for specific model settings (from left to right: Qwen2.5-0.5B, Qwen2.5-3B, and Qwen2.5-3B in LoRA). Results demonstrate significant efficiency improvements in SparseBalance and confirm the effectiveness of each component.

preserves original data randomness. In this way, SAB provides a better coarse-grained initialization for DST, while DST further refines the residual imbalance at runtime.

4 EXPERIMENTS

We evaluate SparseBalance in both system efficiency and model accuracy.

4.1 EXPERIMENTAL SETTINGS

Testbeds. The experiments were conducted on two GPU clusters. The first cluster contains 4 nodes and each node equipped with 8×H800 GPUs. The second cluster contains 4 nodes and each node equipped with 8×H20 GPUs. Both clusters are interconnected via NVLink within the node and InfiniBand network across the node. The software stack includes CUDA 12.4 NVIDIA Corporation (2026a), PyTorch 2.5.1 Paszke et al. (2019), and NCCL 2.21.5 NVIDIA Corporation (2026b). We implement SparseBalance on top of Megatron-LM Shoeybi et al. (2020); Narayanan et al. (2021); Korthikanti et al. (2023) (core v0.13.2) and ms-swift Zhao et al. (2025) (v3.10.0).

Models, Datasets, and Baseline. We evaluate SparseBalance using two open-source LLMs: Qwen2.5-0.5B and Qwen2.5-3B Qwen et al. (2024). We use two real-world long-context datasets: ChatQA2-Long-SFT Xu et al. (2025b) and LongAlign-10k Bai et al. (2024), whose sequence length distributions are shown in Figure 1. To evaluate the model accuracy, we conduct long context supervised training using Qwen2.5-3B model on ChatQA2-Long-SFT dataset with 3 epoch, selecting MoBA Lu et al. (2025) as our base sparse attention algorithm. To assess the long context capabilities, we evaluate the model on three downstream tasks.

Hyper-parameters Configuration. We select three representative anchor strategies: Mean-Anchor (MEAN), Min-Anchor (MIN), and Max-Anchor (MAX), which align the workload to the mean, minimum, and maximum of predicted micro-batch latency, respectively. For the threshold p, we mainly evaluate two settings, $p = 0.1$ and $p = 0.2$, denoted as ANCHOR_p. By default, we use a hybrid 4D parallel configuration with DP $= 4$, PP $= 4$, TP $= 2$, and SP $= 2$, together with selective activation recomputation, a micro-batch size of 1, and a global batch size of 16. We set the peak learning rate to 1×10^{-6}, the warm-up fraction to 0.20, and the minimum learning rate to 1×10^{-7}. For Parameter-Efficient Fine-Tuning (PEFT) experiments with LoRA Hu et al. (2022); Chen et al. (2024), we use rank 32, alpha 64, and dropout 0.05.

4.2 EFFICIENCY EVALUATION

We use $\text{MEAN}_{0.1}$ as the default configuration; Section 4.4 analyzes its trade-off. Ablations of the anchor, threshold, and micro-batch size appear in Appendix C.2.

Table 1: LBB denotes a length-based batching strategy. Imbalance measures the average micro-batch-level imbalance degree (lower is better). Overheads report the additional cost introduced by SparseBalance.

Config.	Imbalance ↓	Iter. (ms)	Overheads (ms)	Speedup ↑
Baseline	1.81×	4202.09	0.00	1.00×
+DST	1.73×	3901.57	37.14	1.08×
+SAB	1.41×	3470.73	1.81	1.21×
+LBB	1.58×	3421.98	0.00	1.23×
+SAB+DST	1.34×	3107.83	44.23	**1.35**×
+LBB+DST	1.41×	3272.88	39.01	1.28×

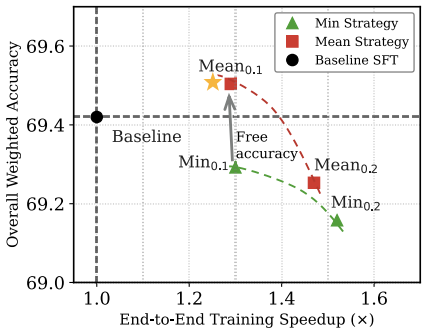

Figure 8: Trade-off between training speedup and downstream model performance under different configurations in SparseBalance.

4.2.1 End-to-End Performance

We evaluate the end-to-end performance of SparseBalance on the two datasets and two GPU clusters described in Section 4.1. We assess SparseBalance using three model configurations: 0.5B, 3B, and 3B with LoRA. As shown in Figure 7, SparseBalance achieves an average end-to-end speedup of 1.30×, peaking on the Qwen2.5-3B model trained on the LongAlign-10k dataset. We observe similar speedup trends on both the H800 and H20 clusters, which suggests that the efficiency benefit of SparseBalance is robust across different GPU platforms. Furthermore, we perform an ablation study to isolate the effectiveness of each component: "+DST" denotes the exclusive application of the Dynamic Sparsity Tuning module, while "+SAB" indicates the standalone use of the Sparsity-Aware Batching module.

Dataset- and model-specific speedup analysis is provided in Appendix C.1.

4.2.2 Case Study

To further understand where the efficiency gains of SparseBalance come from, we conduct a case study that jointly examines workload imbalance and overheads in iteration level. In particular, we compare SparseBalance with a *Length-Based Batching* (LBB) strategy, which uses sequence length as the batching weight and serves as a representative baseline for existing batching methods. In contrast, SparseBalance uses latency prediction to guide workload partitioning. We conduct this study on Qwen2.5-3B with ChatQA2 under a single-node setting with DP $= 2$, PP $= 4$, micro-batch size $= 2$, and global batch size $= 16$.

We quantify the workload imbalance in each iteration as

$$Imbalance = \frac{\max(T_{mb,i})}{\frac{1}{N}\sum T_{mb,i}} \tag{4}$$

where $T_{mb,i}$ denotes the average execution time of micro-batch i across ranks. This metric reflects how unevenly the workload is distributed within an iteration. Although it is not directly proportional to the end-to-end iteration time, it provides a useful diagnostic signal for analyzing the straggler effect.

Table 1 reveals three observations. First, the overheads of SparseBalance remains small. SAB introduces negligible overhead, while the full SAB+DST pipeline adds only 44.23 ms, which is below 1.5% of the total iteration time. Second, SAB achieves a slightly lower standalone speedup than LBB. We attribute this gap to the fact that SAB is designed to cooperate with the subsequent runtime adjustment in DST, and therefore performs a more aggressive data reorganization based on the predicted sparsity. When DST is disabled, this prediction cannot be fully realized at runtime, making sub-optimal efficiency. Third, once combined with DST, SAB consistently outperforms LBB in both imbalance degree and end-to-end speedup, highlights the strong synergy between SAB and DST.

4.3 Accuracy Evaluation

Training-loss curves and analysis appear in Appendix C.3; detailed NIAH, general-benchmark, and LongBench results appear in Appendix C.4.

4.3.1 Benchmark Evaluation

To thoroughly evaluate the model capabilities, we assess the fine-tuned models on three benchmarks, representing distinct core abilities: long-context understanding (LongBench), exact retrieval (Needle-in-a-Haystack, NIAH) and standard zero-shot reasoning (General Benchmarks).

Long-Context Understanding. Table 4 reports the category-level LongBench results, which serve as our primary downstream evaluation for long-context capability. Among all evaluated configurations, $\text{MEAN}_{0.1}$ achieves the best overall score (39.46), outperforming the Baseline SFT (39.28). In contrast, relaxing the threshold to $p = 0.2$ generally causes slight degradation, indicating that overly aggressive tuning hurts model fidelity. We also observe that $\text{MEAN}_{0.1}$ provides a better overall balance than $\text{MIN}_{0.1}$, which is consistent with the design intuition that bidirectional tuning can exploit non-critical-path computation for accuracy improvement. We further note that our fine-tuning data is QA-oriented (ChatQA2-Long-SFT), which helps to explain why the gains of SparseBalance are more apparent on QA tasks.

Exact Retrieval. Table 2 shows that all configurations achieve perfect retrieval up to 16K context length. Even at 64K, $\text{MEAN}_{0.1}$ remains nearly on par with the Baseline SFT (97.53% vs. 97.60%), indicating that DST does not noticeably harm exact long-context retrieval.

General Benchmarks. On standard zero-shot reasoning tasks, SparseBalance remains highly competitive with the baseline. In particular, $\text{MEAN}_{0.1}$ achieves the best average score (0.6967), slightly outperforming the Baseline SFT (0.6958), which suggests that the proposed dynamic tuning does not degrade general model capability.

Among the evaluated configurations, $\text{MEAN}_{0.1}$ consistently provides the best overall balance, while $p = 0.1$ serves as a necessary strict bound for maintaining model fidelity.

4.4 Trade-off Analysis

The previous sections evaluate the efficiency and model quality of SparseBalance separately. Here, we jointly examine these two aspects to understand the practical operating point of dynamic sparsity tuning and the inherent trade-off in SparseBalance. To directly visualize this trade-off, we plot the relation between training speedup and model accuracy under different configurations in Figure 8, where the accuracy is represented by the weighted average score across all evaluated benchmarks.

The resulting configurations form a clear Pareto frontier. In general, increasing the threshold p moves the operating point toward higher speedup but lower model quality, confirming that p directly governs how aggressive we trade accuracy for efficiency. Meanwhile, anchor point MEAN generally achieves better accuracy than the MIN while delivering comparable system efficiency. Among all evaluated configurations, $\text{MEAN}_{0.1}$ lies at the most favorable point on this frontier, which improves both efficiency and accuracy. Therefore, we use $\text{MEAN}_{0.1}$ as the default practical setting in the efficiency evaluation of Section 4.2.

Related Work. See Appendix D.

5 Conclusion

In this paper, we propose SparseBalance, a novel algorithm-system co-design framework for long-context sparse training. It addresses a central challenge in these scenarios, where sequence-length heterogeneity, sparsity heterogeneity, and distributed workload imbalance are tightly coupled during training. To tackle this problem, we develop workload-aware dynamic sparsity tuning for fine-grained runtime balancing, a profiling-based latency predictor for practical workload estimation, and sparsity-aware batching for coarse-grained initialization. Experiments on real-world long-context datasets and evaluations on downstream tasks show that SparseBalance improves end-to-end training efficiency by up to 1.33× while still improving the model's long-context capability.

AI USE STATEMENT

An LLM assistant was used to reorganize and format this manuscript for ICLR, review submission policies, and audit existing results and implementation code. It did not generate new training results. The authors remain responsible for verifying all text, data, references, and claims and should disclose this use in the submission form.

REFERENCES

Yushi Bai, Xin Lv, Jiajie Zhang, Yuze He, Ji Qi, Lei Hou, Jie Tang, Yuxiao Dong, and Juanzi Li. LongAlign: A recipe for long context alignment of large language models. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 1376–1395, Miami, Florida, USA, November 2024. Association for Computational Linguistics. URL https://doi.org/10.18653/v1/2024.findings-emnlp.74.

Yukang Chen, Shengju Qian, Haotian Tang, Xin Lai, Zhijian Liu, Song Han, and Jiaya Jia. LongloRA: Efficient fine-tuning of long-context large language models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=6PmJoRfdaK.

DeepSeek-AI, Aixin Liu, Aoxue Mei, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenhao Xu, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Erhang Li, Fangqi Zhou, Fangyun Lin, Fucong Dai, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Hanwei Xu, Hao Li, Haofen Liang, Haoran Wei, Haowei Zhang, Haowen Luo, Haozhe Ji, Honghui Ding, Hongxuan Tang, Huanqi Cao, Huazuo Gao, Hui Qu, Hui Zeng, Jialiang Huang, Jiashi Li, Jiaxin Xu, Jiewen Hu, Jingchang Chen, Jingting Xiang, Jingyang Yuan, Jingyuan Cheng, Jinhua Zhu, Jun Ran, Junguang Jiang, Junjie Qiu, Junlong Li, Junxiao Song, Kai Dong, Kaige Gao, Kang Guan, Kexin Huang, Kexing Zhou, Kezhao Huang, Kuai Yu, Lean Wang, Lecong Zhang, Lei Wang, Liang Zhao, Liangsheng Yin, Lihua Guo, Lingxiao Luo, Linwang Ma, Litong Wang, Liyue Zhang, M. S. Di, M. Y Xu, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingxu Zhou, Panpan Huang, Peixin Cong, Peiyi Wang, Qiancheng Wang, Qihao Zhu, Qingyang Li, Qinyu Chen, Qiushi Du, Ruiling Xu, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, Runqiu Yin, Runxin Xu, Ruomeng Shen, Ruoyu Zhang, S. H. Liu, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shaofei Cai, Shaoyuan Chen, Shengding Hu, Shengyu Liu, Shiqiang Hu, Shirong Ma, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, Songyang Zhou, Tao Ni, Tao Yun, Tian Pei, Tian Ye, Tianyuan Yue, Wangding Zeng, Wen Liu, Wenfeng Liang, Wenjie Pang, Wenjing Luo, Wenjun Gao, Wentao Zhang, Xi Gao, Xiangwen Wang, Xiao Bi, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaokang Zhang, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xingkai Yu, Xingyou Li, Xinyu Yang, Xinyuan Li, Xu Chen, Xuecheng Su, Xuehai Pan, Xuheng Lin, Xuwei Fu, Y. Q. Wang, Yang Zhang, Yanhong Xu, Yanru Ma, Yao Li, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Qian, Yi Yu, Yichao Zhang, Yifan Ding, Yifan Shi, Yiliang Xiong, Ying He, Ying Zhou, Yinmin Zhong, Yishi Piao, Yisong Wang, Yixiao Chen, Yixuan Tan, Yixuan Wei, Yiyang Ma, Yiyuan Liu, Yonglun Yang, Yongqiang Guo, Yongtong Wu, Yu Wu, Yuan Cheng, Yuan Ou, Yuanfan Xu, Yuduan Wang, Yue Gong, Yuhan Wu, Yuheng Zou, Yukun Li, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Z. F. Wu, Z. Z. Ren, Zehua Zhao, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhibin Gou, Zhicheng Ma, Zhigang Yan, Zhihong Shao, Zhixian Huang, Zhiyu Wu, Zhuoshu Li, Zhuping Zhang, Zian Xu, Zihao Wang, Zihui Gu, Zijia Zhu, Zilin Li, Zipeng Zhang, Ziwei Xie, Ziyi Gao, Zizheng Pan, Zongqing Yao, Bei Feng, Hui Li, J. L. Cai, Jiaqi Ni, Lei Xu, Meng Li, Ning Tian, R. J. Chen, R. L. Jin, S. S. Li, Shuang Zhou, Tianyu Sun, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xinnan Song, Xinyi Zhou, Y. X. Zhu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, Dongjie Ji, Jian Liang, Jianzhong Guo, Jin Chen, Leyi Xia, Miaojun Wang, Mingming Li, Peng Zhang, Ruyi Chen, Shangmian Sun, Shaoqing Wu, Shengfeng Ye, T. Wang, W. L. Xiao, Wei An, Xianzu Wang, Xiaowen Sun, Xiaoxiang Wang, Ying Tang, Yukun Zha, Zekai Zhang, Zhe Ju, Zhen Zhang, and Zihua Qu. Deepseek-v3.2: Pushing the frontier of open large language models, 2025. URL https://doi.org/10.48550/arXiv.2512.02556.

Team GLM, :, Aohan Zeng, Bin Xu, Bowen Wang, Chenhui Zhang, Da Yin, Dan Zhang, Diego Rojas, Guanyu Feng, Hanlin Zhao, Hanyu Lai, Hao Yu, Hongning Wang, Jiadai Sun, Jiajie Zhang, Jiale Cheng, Jiayi Gui, Jie Tang, Jing Zhang, Jingyu Sun, Juanzi Li, Lei Zhao, Lindong Wu, Lucen Zhong, Mingdao Liu, Minlie Huang, Peng Zhang, Qinkai Zheng, Rui Lu, Shuaiqi Duan, Shudan Zhang, Shulin Cao, Shuxun Yang, Weng Lam Tam, Wenyi Zhao, Xiao Liu, Xiao Xia, Xiaohan Zhang, Xiaotao Gu, Xin Lv, Xinghan Liu, Xinyi Liu, Xinyue Yang, Xixuan Song, Xunkai Zhang, Yifan An, Yifan Xu, Yilin Niu, Yuantao Yang, Yueyan Li, Yushi Bai, Yuxiao Dong, Zehan Qi, Zhaoyu Wang, Zhen Yang, Zhengxiao Du, Zhenyu Hou, and Zihan Wang. Chatglm: A family of large language models from glm-130b to glm-4 all tools, 2024. URL https://doi.org/10.48550/arXiv.2406.12793.

GLM-5-Team, :, Aohan Zeng, Xin Lv, Zhenyu Hou, Zhengxiao Du, Qinkai Zheng, Bin Chen, Da Yin, Chendi Ge, Chenghua Huang, Chengxing Xie, Chenzheng Zhu, Congfeng Yin, Cunxiang Wang, Gengzheng Pan, Hao Zeng, Haoke Zhang, Haoran Wang, Huilong Chen, Jiajie Zhang, Jian Jiao, Jiaqi Guo, Jingsen Wang, Jingzhao Du, Jinzhu Wu, Kedong Wang, Lei Li, Lin Fan, Lucen Zhong, Mingdao Liu, Mingming Zhao, Pengfan Du, Qian Dong, Rui Lu, Shuang-Li, Shulin Cao, Song Liu, Ting Jiang, Xiaodong Chen, Xiaohan Zhang, Xuancheng Huang, Xuezhen Dong, Yabo Xu, Yao Wei, Yifan An, Yilin Niu, Yitong Zhu, Yuanhao Wen, Yukuo Cen, Yushi Bai, Zhongpei Qiao, Zihan Wang, Zikang Wang, Zilin Zhu, Ziqiang Liu, Zixuan Li, Bojie Wang, Bosi Wen, Can Huang, Changpeng Cai, Chao Yu, Chen Li, Chengwei Hu, Chenhui Zhang, Dan Zhang, Daoyan Lin, Dayong Yang, Di Wang, Ding Ai, Erle Zhu, Fangzhou Yi, Feiyu Chen, Guohong Wen, Hailong Sun, Haisha Zhao, Haiyi Hu, Hanchen Zhang, Hanrui Liu, Hanyu Zhang, Hao Peng, Hao Tai, Haobo Zhang, He Liu, Hongwei Wang, Hongxi Yan, Hongyu Ge, Huan Liu, Huanpeng Chu, Jia'ni Zhao, Jiachen Wang, Jiajing Zhao, Jiamin Ren, Jiapeng Wang, Jiaxin Zhang, Jiayi Gui, Jiayue Zhao, Jijie Li, Jing An, Jing Li, Jingwei Yuan, Jinhua Du, Jinxin Liu, Junkai Zhi, Junwen Duan, Kaiyue Zhou, Kangjian Wei, Ke Wang, Keyun Luo, Laiqiang Zhang, Leigang Sha, Liang Xu, Lindong Wu, Lintao Ding, Lu Chen, Minghao Li, Nianyi Lin, Pan Ta, Qiang Zou, Rongjun Song, Ruiqi Yang, Shangqing Tu, Shangtong Yang, Shaoxiang Wu, Shengyan Zhang, Shijie Li, Shuang Li, Shuyi Fan, Wei Qin, Wei Tian, Weining Zhang, Wenbo Yu, Wenjie Liang, Xiang Kuang, Xiangmeng Cheng, Xiangyang Li, Xiaoquan Yan, Xiaowei Hu, Xiaoying Ling, Xing Fan, Xingye Xia, Xinyuan Zhang, Xinze Zhang, Xirui Pan, Xu Zou, Xunkai Zhang, Yadi Liu, Yandong Wu, Yanfu Li, Yidong Wang, Yifan Zhu, Yijun Tan, Yilin Zhou, Yiming Pan, Ying Zhang, Yinpei Su, Yipeng Geng, Yong Yan, Yonglin Tan, Yuean Bi, Yuhan Shen, Yuhao Yang, Yujiang Li, Yunan Liu, Yunqing Wang, Yuntao Li, Yurong Wu, Yutao Zhang, Yuxi Duan, Yuxuan Zhang, Zezhen Liu, Zhengtao Jiang, Zhenhe Yan, Zheyu Zhang, Zhixiang Wei, Zhuo Chen, Zhuoer Feng, Zijun Yao, Ziwei Chai, Ziyuan Wang, Zuzhou Zhang, Bin Xu, Minlie Huang, Hongning Wang, Juanzi Li, Yuxiao Dong, and Jie Tang. Glm-5: from vibe coding to agentic engineering, 2026. URL https://doi.org/10.48550/arXiv.2602.15763.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurelien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Roziere, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, Danny Wyatt, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Francisco Guzmán, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Govind Thattai, Graeme Nail, Gregoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel Kloumann, Ishan Misra, Ivan Evtimov, Jack Zhang, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Karthik Prasad, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, Khalid El-Arini, Krithika Iyer, Kshitiz Malik, Kuenley Chiu, Kunal Bhalla, Kushal Lakhotia, Lauren Rantala-Yeary, Laurens van der Maaten, Lawrence Chen, Liang Tan, Liz Jenkins, Louis Martin, Lovish Madaan, Lubo Malo, Lukas Blecher, Lukas Landzaat, Luke de Oliveira, Madeline Muzzi, Mahesh Pasupuleti, Mannat Singh, Manohar Paluri,

Marcin Kardas, Maria Tsimpoukelli, Mathew Oldham, Mathieu Rita, Maya Pavlova, Melanie Kambadur, Mike Lewis, Min Si, Mitesh Kumar Singh, Mona Hassan, Naman Goyal, Narjes Torabi, Nikolay Bashlykov, Nikolay Bogoychev, Niladri Chatterji, Ning Zhang, Olivier Duchenne, Onur Çelebi, Patrick Alrassy, Pengchuan Zhang, Pengwei Li, Petar Vasic, Peter Weng, Prajjwal Bhargava, Pratik Dubal, Praveen Krishnan, Punit Singh Koura, Puxin Xu, Qing He, Qingxiao Dong, Ragavan Srinivasan, Raj Ganapathy, Ramon Calderer, Ricardo Silveira Cabral, Robert Stojnic, Roberta Raileanu, Rohan Maheswari, Rohit Girdhar, Rohit Patel, Romain Sauvestre, Ronnie Polidoro, Roshan Sumbaly, Ross Taylor, Ruan Silva, Rui Hou, Rui Wang, Saghar Hosseini, Sahana Chennabasappa, Sanjay Singh, Sean Bell, Seohyun Sonia Kim, Sergey Edunov, Shaoliang Nie, Sharan Narang, Sharath Raparthy, Sheng Shen, Shengye Wan, Shruti Bhosale, Shun Zhang, Simon Vandenhende, Soumya Batra, Spencer Whitman, Sten Sootla, Stephane Collot, Suchin Gururangan, Sydney Borodinsky, Tamar Herman, Tara Fowler, Tarek Sheasha, Thomas Georgiou, Thomas Scialom, Tobias Speckbacher, Todor Mihaylov, Tong Xiao, Ujjwal Karn, Vedanuj Goswami, Vibhor Gupta, Vignesh Ramanathan, Viktor Kerkez, Vincent Gonguet, Virginie Do, Vish Vogeti, Vítor Albiero, Vladan Petrovic, Weiwei Chu, Wenhan Xiong, Wenyin Fu, Whitney Meers, Xavier Martinet, Xiaodong Wang, Xiaofang Wang, Xiaoqing Ellen Tan, Xide Xia, Xinfeng Xie, Xuchao Jia, Xuewei Wang, Yaelle Goldschlag, Yashesh Gaur, Yasmine Babaei, Yi Wen, Yiwen Song, Yuchen Zhang, Yue Li, Yuning Mao, Zacharie Delpierre Coudert, Zheng Yan, Zhengxing Chen, Zoe Papakipos, Aaditya Singh, Aayushi Srivastava, Abha Jain, Adam Kelsey, Adam Shajnfeld, Adithya Gangidi, Adolfo Victoria, Ahuva Goldstand, Ajay Menon, Ajay Sharma, Alex Boesenberg, Alexei Baevski, Allie Feinstein, Amanda Kallet, Amit Sangani, Amos Teo, Anam Yunus, Andrei Lupu, Andres Alvarado, Andrew Caples, Andrew Gu, Andrew Ho, Andrew Poulton, Andrew Ryan, Ankit Ramchandani, Annie Dong, Annie Franco, Anuj Goyal, Aparajita Saraf, Arkabandhu Chowdhury, Ashley Gabriel, Ashwin Bharambe, Assaf Eisenman, Azadeh Yazdan, Beau James, Ben Maurer, Benjamin Leonhardi, Bernie Huang, Beth Loyd, Beto De Paola, Bhargavi Paranjape, Bing Liu, Bo Wu, Boyu Ni, Braden Hancock, Bram Wasti, Brandon Spence, Brani Stojkovic, Brian Gamido, Britt Montalvo, Carl Parker, Carly Burton, Catalina Mejia, Ce Liu, Changhan Wang, Changkyu Kim, Chao Zhou, Chester Hu, Ching-Hsiang Chu, Chris Cai, Chris Tindal, Christoph Feichtenhofer, Cynthia Gao, Damon Civin, Dana Beaty, Daniel Kreymer, Daniel Li, David Adkins, David Xu, Davide Testuggine, Delia David, Devi Parikh, Diana Liskovich, Didem Foss, Dingkang Wang, Duc Le, Dustin Holland, Edward Dowling, Eissa Jamil, Elaine Montgomery, Eleonora Presani, Emily Hahn, Emily Wood, Eric-Tuan Le, Erik Brinkman, Esteban Arcaute, Evan Dunbar, Evan Smothers, Fei Sun, Felix Kreuk, Feng Tian, Filippos Kokkinos, Firat Ozgenel, Francesco Caggioni, Frank Kanayet, Frank Seide, Gabriela Medina Florez, Gabriella Schwarz, Gada Badeer, Georgia Swee, Gil Halpern, Grant Herman, Grigory Sizov, Guangyi, Zhang, Guna Lakshminarayanan, Hakan Inan, Hamid Shojanazeri, Han Zou, Hannah Wang, Hanwen Zha, Haroun Habeeb, Harrison Rudolph, Helen Suk, Henry Aspegren, Hunter Goldman, Hongyuan Zhan, Ibrahim Damlaj, Igor Molybog, Igor Tufanov, Ilias Leontiadis, Irina-Elena Veliche, Itai Gat, Jake Weissman, James Geboski, James Kohli, Janice Lam, Japhet Asher, Jean-Baptiste Gaya, Jeff Marcus, Jeff Tang, Jennifer Chan, Jenny Zhen, Jeremy Reizenstein, Jeremy Teboul, Jessica Zhong, Jian Jin, Jingyi Yang, Joe Cummings, Jon Carvill, Jon Shepard, Jonathan McPhie, Jonathan Torres, Josh Ginsburg, Junjie Wang, Kai Wu, Kam Hou U, Karan Saxena, Kartikay Khandelwal, Katayoun Zand, Kathy Matosich, Kaushik Veeraraghavan, Kelly Michelena, Keqian Li, Kiran Jagadeesh, Kun Huang, Kunal Chawla, Kyle Huang, Lailin Chen, Lakshya Garg, Lavender A, Leandro Silva, Lee Bell, Lei Zhang, Liangpeng Guo, Licheng Yu, Liron Moshkovich, Luca Wehrstedt, Madian Khabsa, Manav Avalani, Manish Bhatt, Martynas Mankus, Matan Hasson, Matthew Lennie, Matthias Reso, Maxim Groshev, Maxim Naumov, Maya Lathi, Meghan Keneally, Miao Liu, Michael L. Seltzer, Michal Valko, Michelle Restrepo, Mihir Patel, Mik Vyatskov, Mikayel Samvelyan, Mike Clark, Mike Macey, Mike Wang, Miquel Jubert Hermoso, Mo Metanat, Mohammad Rastegari, Munish Bansal, Nandhini Santhanam, Natascha Parks, Natasha White, Navyata Bawa, Nayan Singhal, Nick Egebo, Nicolas Usunier, Nikhil Mehta, Nikolay Pavlovich Laptev, Ning Dong, Norman Cheng, Oleg Chernoguz, Olivia Hart, Omkar Salpekar, Ozlem Kalinli, Parkin Kent, Parth Parekh, Paul Saab, Pavan Balaji, Pedro Rittner, Philip Bontrager, Pierre Roux, Piotr Dollar, Polina Zvyagina, Prashant Ratanchandani, Pritish Yuvraj, Qian Liang, Rachad Alao, Rachel Rodriguez, Rafi Ayub, Raghotham Murthy, Raghu Nayani, Rahul Mitra, Rangaprabhu Parthasarathy, Raymond Li, Rebekkah Hogan, Robin Battey, Rocky Wang, Russ Howes, Ruty Rinott, Sachin Mehta, Sachin Siby, Sai Jayesh Bondu, Samyak Datta, Sara Chugh, Sara Hunt, Sargun Dhillon, Sasha Sidorov, Satadru Pan, Saurabh Mahajan, Saurabh Verma, Seiji Yamamoto, Sharadh Ramaswamy, Shaun Lindsay, Shaun Lindsay, Sheng Feng, Shenghao Lin, Shengxin Cindy Zha, Shishir Patil, Shiva Shankar, Shuqiang Zhang, Shuqiang

Zhang, Sinong Wang, Sneha Agarwal, Soji Sajuyigbe, Soumith Chintala, Stephanie Max, Stephen Chen, Steve Kehoe, Steve Satterfield, Sudarshan Govindaprasad, Sumit Gupta, Summer Deng, Sungmin Cho, Sunny Virk, Suraj Subramanian, Sy Choudhury, Sydney Goldman, Tal Remez, Tamar Glaser, Tamara Best, Thilo Koehler, Thomas Robinson, Tianhe Li, Tianjun Zhang, Tim Matthews, Timothy Chou, Tzook Shaked, Varun Vontimitta, Victoria Ajayi, Victoria Montanez, Vijai Mohan, Vinay Satish Kumar, Vishal Mangla, Vlad Ionescu, Vlad Poenaru, Vlad Tiberiu Mihailescu, Vladimir Ivanov, Wei Li, Wenchen Wang, Wenwen Jiang, Wes Bouaziz, Will Constable, Xiaocheng Tang, Xiaojian Wu, Xiaolan Wang, Xilun Wu, Xinbo Gao, Yaniv Kleinman, Yanjun Chen, Ye Hu, Ye Jia, Ye Qi, Yenda Li, Yilin Zhang, Ying Zhang, Yossi Adi, Youngjin Nam, Yu, Wang, Yu Zhao, Yuchen Hao, Yundi Qian, Yunlu Li, Yuzi He, Zach Rait, Zachary DeVito, Zef Rosnbrick, Zhaoduo Wen, Zhenyu Yang, Zhiwei Zhao, and Zhiyu Ma. The llama 3 herd of models, 2024. URL https://doi.org/10.48550/arXiv.2407.21783.

Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Yanping Huang, Youlong Cheng, Ankur Bapna, Orhan Firat, Dehao Chen, Mia Chen, HyoukJoong Lee, Jiquan Ngiam, Quoc V Le, Yonghui Wu, and zhifeng Chen. Gpipe: Efficient training of giant neural networks using pipeline parallelism. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://dl.acm.org/doi/10.5555/3454287.3454297.

Sam Ade Jacobs, Masahiro Tanaka, Chengming Zhang, Minjia Zhang, Reza Yazdani Aminadabi, Shuaiwen Leon Song, Samyam Rajbhandari, and Yuxiong He. System optimizations for enabling training of extreme long sequence transformer models. In *Proceedings of the 43rd ACM Symposium on Principles of Distributed Computing*, pp. 121–130. ACM, 2024. ISBN 979-8-4007-0668-4. doi: 10.1145/3662158.3662806. URL https://dl.acm.org/doi/10.1145/3662158.3662806.

Vijay Anand Korthikanti, Jared Casper, Sangkug Lym, Lawrence McAfee, Michael Andersch, Mohammad Shoeybi, and Bryan Catanzaro. Reducing activation recomputation in large transformer models. In *MLSys*, 2023. URL https://proceedings.mlsys.org/paper_files/paper/2023/hash/80083951326cf5b35e5100260d64ed81-Abstract-mlsys2023.html.

Shen Li, Yanli Zhao, Rohan Varma, Omkar Salpekar, Pieter Noordhuis, Teng Li, Adam Paszke, Jeff Smith, Brian Vaughan, Pritam Damania, and Soumith Chintala. Pytorch distributed: experiences on accelerating data parallel training. *Proc. VLDB Endow.*, 13(12):3005–3018, August 2020. ISSN 2150-8097. doi: 10.14778/3415478.3415530. URL https://doi.org/10.14778/3415478.3415530.

Wenxuan Li, Chengruidong Zhang, Huiqiang Jiang, Yucheng Li, Yuqing Yang, and Lili Qiu. MTraining: Efficient distributed training for ultra-long contexts via dynamic sparse attention. In *ES-FoMo III: 3rd Workshop on Efficient Systems for Foundation Models*, 2025. URL https://openreview.net/forum?id=uOKzCkrV5L.

Chaofan Lin, Jiaming Tang, Shuo Yang, Hanshuo Wang, Tian Tang, Boyu Tian, Ion Stoica, Song Han, and Mingyu Gao. Twilight: Adaptive attention sparsity with hierarchical top-p pruning. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=Ve693NkzcU.

Hao Liu, Matei Zaharia, and Pieter Abbeel. Ringattention with blockwise transformers for near-infinite context. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=WsRHpHH4s0.

Enzhe Lu, Zhejun Jiang, Jingyuan Liu, Yulun Du, Tao Jiang, Chao Hong, Shaowei Liu, Weiran He, Enming Yuan, Yuzhi Wang, Zhiqi Huang, Huan Yuan, Suting Xu, Xinran Xu, Guokun Lai, Yanru Chen, Huabin Zheng, Junjie Yan, Jianlin Su, Yuxin Wu, Yutao Zhang, Zhilin Yang, Xinyu Zhou, Mingxing Zhang, and Jiezhong Qiu. MoBA: Mixture of block attention for long-context LLMs.

In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=RlqYCpTu1P.

Deepak Narayanan, Aaron Harlap, Amar Phanishayee, Vivek Seshadri, Nikhil R. Devanur, Gregory R. Ganger, Phillip B. Gibbons, and Matei Zaharia. Pipedream: generalized pipeline parallelism for dnn training. In *Proceedings of the 27th ACM Symposium on Operating Systems Principles*, SOSP '19, pp. 1–15, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450368735. doi: 10.1145/3341301.3359646. URL https://doi.org/10.1145/3341301.3359646.

Deepak Narayanan, Mohammad Shoeybi, Jared Casper, Patrick LeGresley, Mostofa Patwary, Vijay Korthikanti, Dmitri Vainbrand, Prethvi Kashinkunti, Julie Bernauer, Bryan Catanzaro, Amar Phanishayee, and Matei Zaharia. Efficient large-scale language model training on GPU clusters using megatron-LM. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '21, pp. 1–15. Association for Computing Machinery, 2021. ISBN 978-1-4503-8442-1. doi: 10.1145/3458817.3476209. URL https://dl.acm.org/doi/10.1145/3458817.3476209.

NVIDIA. nvidia/ChatQA2-long-SFT-data · datasets at hugging face, 2025. URL https://huggingface.co/datasets/nvidia/ChatQA2-Long-SFT-data.

NVIDIA Corporation. Nvidia cuda toolkit documentation,, 2026a. URL https://developer.nvidia.com/cuda/toolkit. Accessed: Mar. 14, 2026.

NVIDIA Corporation. Nvidia collective communications library (nccl) documentation, 2026b. URL https://docs.nvidia.com/deeplearning/nccl/. Accessed: Mar. 14, 2026.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://dl.acm.org/doi/10.5555/3454287.3455008.

Penghui Qi, Xinyi Wan, Guangxing Huang, and Min Lin. Zero bubble (almost) pipeline parallelism. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=tuzTN0eIO5.

Qwen, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report, 2024. URL https://doi.org/10.48550/arXiv.2412.15115.

Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. Megatron-lm: Training multi-billion parameter language models using model parallelism, 2020. URL https://doi.org/10.48550/arXiv.1909.08053.

Shaden Smith, Mostofa Patwary, Brandon Norick, Patrick LeGresley, Samyam Rajbhandari, Jared Casper, Zhun Liu, Shrimai Prabhumoye, George Zerveas, Vijay Korthikanti, Elton Zhang, Rewon Child, Reza Yazdani Aminabadi, Julie Bernauer, Xia Song, Mohammad Shoeybi, Yuxiong He, Michael Houston, Saurabh Tiwary, and Bryan Catanzaro. Using deepspeed and megatron to train megatron-turing nlg 530b, a large-scale generative language model, 2022. URL https://doi.org/10.48550/arXiv.2201.11990.

Jiaming Tang, Yilong Zhao, Kan Zhu, Guangxuan Xiao, Baris Kasikci, and Song Han. Quest: query-aware sparsity for efficient long-context llm inference. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024. URL https://dl.acm.org/doi/10.5555/3692070.3694025.

Gemini Team, Petko Georgiev, Ving Ian Lei, Ryan Burnell, Libin Bai, Anmol Gulati, Garrett Tanzer, Damien Vincent, Zhufeng Pan, Shibo Wang, Soroosh Mariooryad, Yifan Ding, Xinyang Geng, Fred Alcober, Roy Frostig, Mark Omernick, Lexi Walker, Cosmin Paduraru, Christina Sorokin, Andrea Tacchetti, Colin Gaffney, Samira Daruki, Olcan Sercinoglu, Zach Gleicher, Juliette Love, Paul Voigtlaender, Rohan Jain, Gabriela Surita, Kareem Mohamed, Rory Blevins, Junwhan Ahn, Tao Zhu, Kornraphop Kawintiranon, Orhan Firat, Yiming Gu, Yujing Zhang, Matthew Rahtz, Manaal Faruqui, Natalie Clay, Justin Gilmer, JD Co-Reyes, Ivo Penchev, Rui Zhu, Nobuyuki Morioka, Kevin Hui, Krishna Haridasan, Victor Campos, Mahdis Mahdieh, Mandy Guo, Samer Hassan, Kevin Kilgour, Arpi Vezer, Heng-Tze Cheng, Raoul de Liedekerke, Siddharth Goyal, Paul Barham, DJ Strouse, Seb Noury, Jonas Adler, Mukund Sundararajan, Sharad Vikram, Dmitry Lepikhin, Michela Paganini, Xavier Garcia, Fan Yang, Dasha Valter, Maja Trebacz, Kiran Vodrahalli, Chulayuth Asawaroengchai, Roman Ring, Norbert Kalb, Livio Baldini Soares, Siddhartha Brahma, David Steiner, Tianhe Yu, Fabian Mentzer, Antoine He, Lucas Gonzalez, Bibo Xu, Raphael Lopez Kaufman, Laurent El Shafey, Junhyuk Oh, Tom Hennigan, George van den Driessche, Seth Odoom, Mario Lucic, Becca Roelofs, Sid Lall, Amit Marathe, Betty Chan, Santiago Ontanon, Luheng He, Denis Teplyashin, Jonathan Lai, Phil Crone, Bogdan Damoc, Lewis Ho, Sebastian Riedel, Karel Lenc, Chih-Kuan Yeh, Aakanksha Chowdhery, Yang Xu, Mehran Kazemi, Ehsan Amid, Anastasia Petrushkina, Kevin Swersky, Ali Khodaei, Gowoon Chen, Chris Larkin, Mario Pinto, Geng Yan, Adria Puigdomenech Badia, Piyush Patil, Steven Hansen, Dave Orr, Sebastien M. R. Arnold, Jordan Grimstad, Andrew Dai, Sholto Douglas, Rishika Sinha, Vikas Yadav, Xi Chen, Elena Gribovskaya, Jacob Austin, Jeffrey Zhao, Kaushal Patel, Paul Komarek, Sophia Austin, Sebastian Borgeaud, Linda Friso, Abhimanyu Goyal, Ben Caine, Kris Cao, Da-Woon Chung, Matthew Lamm, Gabe Barth-Maron, Thais Kagohara, Kate Olszewska, Mia Chen, Kaushik Shivakumar, Rishabh Agarwal, Harshal Godhia, Ravi Rajwar, Javier Snaider, Xerxes Dotiwalla, Yuan Liu, Aditya Barua, Victor Ungureanu, Yuan Zhang, Bat-Orgil Batsaikhan, Mateo Wirth, James Qin, Ivo Danihelka, Tulsee Doshi, Martin Chadwick, Jilin Chen, Sanil Jain, Quoc Le, Arjun Kar, Madhu Gurumurthy, Cheng Li, Ruoxin Sang, Fangyu Liu, Lampros Lamprou, Rich Munoz, Nathan Lintz, Harsh Mehta, Heidi Howard, Malcolm Reynolds, Lora Aroyo, Quan Wang, Lorenzo Blanco, Albin Cassirer, Jordan Griffith, Dipanjan Das, Stephan Lee, Jakub Sygnowski, Zach Fisher, James Besley, Richard Powell, Zafarali Ahmed, Dominik Paulus, David Reitter, Zalan Borsos, Rishabh Joshi, Aedan Pope, Steven Hand, Vittorio Selo, Vihan Jain, Nikhil Sethi, Megha Goel, Takaki Makino, Rhys May, Zhen Yang, Johan Schalkwyk, Christina Butterfield, Anja Hauth, Alex Goldin, Will Hawkins, Evan Senter, Sergey Brin, Oliver Woodman, Marvin Ritter, Eric Noland, Minh Giang, Vijay Bolina, Lisa Lee, Tim Blyth, Ian Mackinnon, Machel Reid, Obaid Sarvana, David Silver, Alexander Chen, Lily Wang, Loren Maggiore, Oscar Chang, Nithya Attaluri, Gregory Thornton, Chung-Cheng Chiu, Oskar Bunyan, Nir Levine, Timothy Chung, Evgenii Eltyshev, Xiance Si, Timothy Lillicrap, Demetra Brady, Vaibhav Aggarwal, Boxi Wu, Yuanzhong Xu, Ross McIlroy, Kartikeya Badola, Paramjit Sandhu, Erica Moreira, Wojciech Stokowiec, Ross Hemsley, Dong Li, Alex Tudor, Pranav Shyam, Elahe Rahimtoroghi, Salem Haykal, Pablo Sprechmann, Xiang Zhou, Diana Mincu, Yujia Li, Ravi Addanki, Kalpesh Krishna, Xiao Wu, Alexandre Frechette, Matan Eyal, Allan Dafoe, Dave Lacey, Jay Whang, Thi Avrahami, Ye Zhang, Emanuel Taropa, Hanzhao Lin, Daniel Toyama, Eliza Rutherford, Motoki Sano, HyunJeong Choe, Alex Tomala, Chalence Safranek-Shrader, Nora Kassner, Mantas Pajarskas, Matt Harvey, Sean Sechrist, Meire Fortunato, Christina Lyu, Gamaleldin Elsayed, Chenkai Kuang, James Lottes, Eric Chu, Chao Jia, Chih-Wei Chen, Peter Humphreys, Kate Baumli, Connie Tao, Rajkumar Samuel, Cicero Nogueira dos Santos, Anders Andreassen, Nemanja Rakićević, Dominik Grewe, Aviral Kumar, Stephanie Winkler, Jonathan Caton, Andrew Brock, Sid Dalmia, Hannah Sheahan, Iain Barr, Yingjie Miao, Paul Natsev, Jacob Devlin, Feryal Behbahani, Flavien Prost, Yanhua Sun, Artiom Myaskovsky, Thanumalayan Sankaranarayana Pillai, Dan Hurt, Angeliki Lazaridou, Xi Xiong, Ce Zheng, Fabio Pardo, Xiaowei Li, Dan Horgan, Joe Stanton, Moran Ambar, Fei Xia, Alejandro Lince, Mingqiu Wang, Basil Mustafa, Albert Webson, Hyo Lee, Rohan Anil, Martin Wicke, Timothy Dozat, Abhishek Sinha, Enrique Piqueras, Elahe Dabir, Shyam Upadhyay, Anudhyan Boral, Lisa Anne Hendricks, Corey Fry, Josip Djolonga, Yi Su, Jake Walker, Jane Labanowski, Ronny Huang, Vedant Misra, Jeremy Chen, RJ Skerry-Ryan, Avi Singh, Shruti Rijhwani, Dian Yu, Alex Castro-Ros, Beer Changpinyo, Romina Datta, Sumit Bagri, Arnar Mar Hrafnkelsson, Marcello Maggioni, Daniel Zheng, Yury Sulsky, Shaobo Hou, Tom Le Paine, Antoine Yang, Jason Riesa, Dominika Rogozinska, Dror Marcus, Dalia El Badawy, Qiao Zhang, Luyu Wang, Helen Miller, Jeremy Greer, Lars Lowe Sjos, Azade Nova, Heiga Zen, Rahma Chaabouni, Mihaela Rosca, Jiepu Jiang, Charlie Chen, Ruibo Liu, Tara Sainath, Maxim Krikun, Alex Polozov, Jean-Baptiste Lespiau,

Josh Newlan, Zeyncep Cankara, Soo Kwak, Yunhan Xu, Phil Chen, Andy Coenen, Clemens Meyer, Katerina Tsihlas, Ada Ma, Juraj Gottweis, Jinwei Xing, Chenjie Gu, Jin Miao, Christian Frank, Zeynep Cankara, Sanjay Ganapathy, Ishita Dasgupta, Steph Hughes-Fitt, Heng Chen, David Reid, Keran Rong, Hongmin Fan, Joost van Amersfoort, Vincent Zhuang, Aaron Cohen, Shixiang Shane Gu, Anhad Mohananey, Anastasija Ilic, Taylor Tobin, John Wieting, Anna Bortsova, Phoebe Thacker, Emma Wang, Emily Caveness, Justin Chiu, Eren Sezener, Alex Kaskasoli, Steven Baker, Katie Millican, Mohamed Elhawaty, Kostas Aisopos, Carl Lebsack, Nathan Byrd, Hanjun Dai, Wenhao Jia, Matthew Wiethoff, Elnaz Davoodi, Albert Weston, Lakshman Yagati, Arun Ahuja, Isabel Gao, Golan Pundak, Susan Zhang, Michael Azzam, Khe Chai Sim, Sergi Caelles, James Keeling, Abhanshu Sharma, Andy Swing, YaGuang Li, Chenxi Liu, Carrie Grimes Bostock, Yamini Bansal, Zachary Nado, Ankesh Anand, Josh Lipschultz, Abhijit Karmarkar, Lev Proleev, Abe Ittycheriah, Soheil Hassas Yeganeh, George Polovets, Aleksandra Faust, Jiao Sun, Alban Rrustemi, Pen Li, Rakesh Shivanna, Jeremiah Liu, Chris Welty, Federico Lebron, Anirudh Baddepudi, Sebastian Krause, Emilio Parisotto, Radu Soricut, Zheng Xu, Dawn Bloxwich, Melvin Johnson, Behnam Neyshabur, Justin Mao-Jones, Renshen Wang, Vinay Ramasesh, Zaheer Abbas, Arthur Guez, Constant Segal, Duc Dung Nguyen, James Svensson, Le Hou, Sarah York, Kieran Milan, Sophie Bridgers, Wiktor Gworek, Marco Tagliasacchi, James Lee-Thorp, Michael Chang, Alexey Guseynov, Ale Jakse Hartman, Michael Kwong, Ruizhe Zhao, Sheleem Kashem, Elizabeth Cole, Antoine Miech, Richard Tanburn, Mary Phuong, Filip Pavetic, Sebastien Cevey, Ramona Comanescu, Richard Ives, Sherry Yang, Cosmo Du, Bo Li, Zizhao Zhang, Mariko Iinuma, Clara Huiyi Hu, Aurko Roy, Shaan Bijwadia, Zhenkai Zhu, Danilo Martins, Rachel Saputro, Anita Gergely, Steven Zheng, Dawei Jia, Ioannis Antonoglou, Adam Sadovsky, Shane Gu, Yingying Bi, Alek Andreev, Sina Samangooei, Mina Khan, Tomas Kocisky, Angelos Filos, Chintu Kumar, Colton Bishop, Adams Yu, Sarah Hodkinson, Sid Mittal, Premal Shah, Alexandre Moufarek, Yong Cheng, Adam Bloniarz, Jaehoon Lee, Pedram Pejman, Paul Michel, Stephen Spencer, Vladimir Feinberg, Xuehan Xiong, Nikolay Savinov, Charlotte Smith, Siamak Shakeri, Dustin Tran, Mary Chesus, Bernd Bohnet, George Tucker, Tamara von Glehn, Carrie Muir, Yiran Mao, Hideto Kazawa, Ambrose Slone, Kedar Soparkar, Disha Shrivastava, James Cobon-Kerr, Michael Sharman, Jay Pavagadhi, Carlos Araya, Karolis Misiunas, Nimesh Ghelani, Michael Laskin, David Barker, Qiujia Li, Anton Briukhov, Neil Houlsby, Mia Glaese, Balaji Lakshminarayanan, Nathan Schucher, Yunhao Tang, Eli Collins, Hyeontaek Lim, Fangxiaoyu Feng, Adria Recasens, Guangda Lai, Alberto Magni, Nicola De Cao, Aditya Siddhant, Zoe Ashwood, Jordi Orbay, Mostafa Dehghani, Jenny Brennan, Yifan He, Kelvin Xu, Yang Gao, Carl Saroufim, James Molloy, Xinyi Wu, Seb Arnold, Solomon Chang, Julian Schrittwieser, Elena Buchatskaya, Soroush Radpour, Martin Polacek, Skye Giordano, Ankur Bapna, Simon Tokumine, Vincent Hellendoorn, Thibault Sottiaux, Sarah Cogan, Aliaksei Severyn, Mohammad Saleh, Shantanu Thakoor, Laurent Shefey, Siyuan Qiao, Meenu Gaba, Shuo yiin Chang, Craig Swanson, Biao Zhang, Benjamin Lee, Paul Kishan Rubenstein, Gan Song, Tom Kwiatkowski, Anna Koop, Ajay Kannan, David Kao, Parker Schuh, Axel Stjerngren, Golnaz Ghiasi, Gena Gibson, Luke Vilnis, Ye Yuan, Felipe Tiengo Ferreira, Aishwarya Kamath, Ted Klimenko, Ken Franko, Kefan Xiao, Indro Bhattacharya, Miteyan Patel, Rui Wang, Alex Morris, Robin Strudel, Vivek Sharma, Peter Choy, Sayed Hadi Hashemi, Jessica Landon, Mara Finkelstein, Priya Jhakra, Justin Frye, Megan Barnes, Matthew Mauger, Dennis Daun, Khuslen Baatarsukh, Matthew Tung, Wael Farhan, Henryk Michalewski, Fabio Viola, Felix de Chaumont Quitry, Charline Le Lan, Tom Hudson, Qingze Wang, Felix Fischer, Ivy Zheng, Elspeth White, Anca Dragan, Jean baptiste Alayrac, Eric Ni, Alexander Pritzel, Adam Iwanicki, Michael Isard, Anna Bulanova, Lukas Zilka, Ethan Dyer, Devendra Sachan, Srivatsan Srinivasan, Hannah Muckenhirn, Honglong Cai, Amol Mandhane, Mukarram Tariq, Jack W. Rae, Gary Wang, Kareem Ayoub, Nicholas FitzGerald, Yao Zhao, Woohyun Han, Chris Alberti, Dan Garrette, Kashyap Krishnakumar, Mai Gimenez, Anselm Levskaya, Daniel Sohn, Josip Matak, Inaki Iturrate, Michael B. Chang, Jackie Xiang, Yuan Cao, Nishant Ranka, Geoff Brown, Adrian Hutter, Vahab Mirrokni, Nanxin Chen, Kaisheng Yao, Zoltan Egyed, Francois Galilee, Tyler Liechty, Praveen Kallakuri, Evan Palmer, Sanjay Ghemawat, Jasmine Liu, David Tao, Chloe Thornton, Tim Green, Mimi Jasarevic, Sharon Lin, Victor Cotruta, Yi-Xuan Tan, Noah Fiedel, Hongkun Yu, Ed Chi, Alexander Neitz, Jens Heitkaemper, Anu Sinha, Denny Zhou, Yi Sun, Charbel Kaed, Brice Hulse, Swaroop Mishra, Maria Georgaki, Sneha Kudugunta, Clement Farabet, Izhak Shafran, Daniel Vlasic, Anton Tsitsulin, Rajagopal Ananthanarayanan, Alen Carin, Guolong Su, Pei Sun, Shashank V, Gabriel Carvajal, Josef Broder, Iulia Comsa, Alena Repina, William Wong, Warren Weilun Chen, Peter Hawkins, Egor Filonov, Lucia Loher, Christoph Hirnschall, Weiyi Wang, Jingchen Ye, Andrea Burns, Hardie Cate, Diana Gage Wright, Federico Piccinini, Lei Zhang, Chu-Cheng Lin, Ionel

Gog, Yana Kulizhskaya, Ashwin Sreevatsa, Shuang Song, Luis C. Cobo, Anand Iyer, Chetan Tekur, Guillermo Garrido, Zhuyun Xiao, Rupert Kemp, Huaixiu Steven Zheng, Hui Li, Ananth Agarwal, Christel Ngani, Kati Goshvadi, Rebeca Santamaria-Fernandez, Wojciech Fica, Xinyun Chen, Chris Gorgolewski, Sean Sun, Roopal Garg, Xinyu Ye, S. M. Ali Eslami, Nan Hua, Jon Simon, Pratik Joshi, Yelin Kim, Ian Tenney, Sahitya Potluri, Lam Nguyen Thiet, Quan Yuan, Florian Luisier, Alexandra Chronopoulou, Salvatore Scellato, Praveen Srinivasan, Minmin Chen, Vinod Koverkathu, Valentin Dalibard, Yaming Xu, Brennan Saeta, Keith Anderson, Thibault Sellam, Nick Fernando, Fantine Huot, Junehyuk Jung, Mani Varadarajan, Michael Quinn, Amit Raul, Maigo Le, Ruslan Habalov, Jon Clark, Komal Jalan, Kalesha Bullard, Achintya Singhal, Thang Luong, Boyu Wang, Sujeevan Rajayogam, Julian Eisenschlos, Johnson Jia, Daniel Finchelstein, Alex Yakubovich, Daniel Balle, Michael Fink, Sameer Agarwal, Jing Li, Dj Dvijotham, Shalini Pal, Kai Kang, Jaclyn Konzelmann, Jennifer Beattie, Olivier Dousse, Diane Wu, Remi Crocker, Chen Elkind, Siddhartha Reddy Jonnalagadda, Jong Lee, Dan Holtmann-Rice, Krystal Kallarackal, Rosanne Liu, Denis Vnukov, Neera Vats, Luca Invernizzi, Mohsen Jafari, Huanjie Zhou, Lilly Taylor, Jennifer Prendki, Marcus Wu, Tom Eccles, Tianqi Liu, Kavya Kopparapu, Francoise Beaufays, Christof Angermueller, Andreea Marzoca, Shourya Sarcar, Hilal Dib, Jeff Stanway, Frank Perbet, Nejc Trdin, Rachel Sterneck, Andrey Khorlin, Dinghua Li, Xihui Wu, Sonam Goenka, David Madras, Sasha Goldshtein, Willi Gierke, Tong Zhou, Yaxin Liu, Yannie Liang, Anais White, Yunjie Li, Shreya Singh, Sanaz Bahargam, Mark Epstein, Sujoy Basu, Li Lao, Adnan Ozturel, Carl Crous, Alex Zhai, Han Lu, Zora Tung, Neeraj Gaur, Alanna Walton, Lucas Dixon, Ming Zhang, Amir Globerson, Grant Uy, Andrew Bolt, Olivia Wiles, Milad Nasr, Ilia Shumailov, Marco Selvi, Francesco Piccinno, Ricardo Aguilar, Sara McCarthy, Misha Khalman, Mrinal Shukla, Vlado Galic, John Carpenter, Kevin Villela, Haibin Zhang, Harry Richardson, James Martens, Matko Bosnjak, Shreyas Rammohan Belle, Jeff Seibert, Mahmoud Alnahlawi, Brian McWilliams, Sankalp Singh, Annie Louis, Wen Ding, Dan Popovici, Lenin Simicich, Laura Knight, Pulkit Mehta, Nishesh Gupta, Chongyang Shi, Saaber Fatehi, Jovana Mitrovic, Alex Grills, Joseph Pagadora, Tsendsuren Munkhdalai, Dessie Petrova, Danielle Eisenbud, Zhishuai Zhang, Damion Yates, Bhavishya Mittal, Nilesh Tripuraneni, Yannis Assael, Thomas Brovelli, Prateek Jain, Mihajlo Velimirovic, Canfer Akbulut, Jiaqi Mu, Wolfgang Macherey, Ravin Kumar, Jun Xu, Haroon Qureshi, Gheorghe Comanici, Jeremy Wiesner, Zhitao Gong, Anton Ruddock, Matthias Bauer, Nick Felt, Anirudh GP, Anurag Arnab, Dustin Zelle, Jonas Rothfuss, Bill Rosgen, Ashish Shenoy, Bryan Seybold, Xinjian Li, Jayaram Mudigonda, Goker Erdogan, Jiawei Xia, Jiri Simsa, Andrea Michi, Yi Yao, Christopher Yew, Steven Kan, Isaac Caswell, Carey Radebaugh, Andre Elisseeff, Pedro Valenzuela, Kay McKinney, Kim Paterson, Albert Cui, Eri Latorre-Chimoto, Solomon Kim, William Zeng, Ken Durden, Priya Ponnapalli, Tiberiu Sosea, Christopher A. Choquette-Choo, James Manyika, Brona Robenek, Harsha Vashisht, Sebastien Pereira, Hoi Lam, Marko Velic, Denese Owusu-Afriyie, Katherine Lee, Tolga Bolukbasi, Alicia Parrish, Shawn Lu, Jane Park, Balaji Venkatraman, Alice Talbert, Lambert Rosique, Yuchung Cheng, Andrei Sozanschi, Adam Paszke, Praveen Kumar, Jessica Austin, Lu Li, Khalid Salama, Bartek Perz, Wooyeol Kim, Nandita Dukkipati, Anthony Baryshnikov, Christos Kaplanis, XiangHai Sheng, Yuri Chervonyi, Caglar Unlu, Diego de Las Casas, Harry Askham, Kathryn Tunyasuvunakool, Felix Gimeno, Siim Poder, Chester Kwak, Matt Miecnikowski, Vahab Mirrokni, Alek Dimitriev, Aaron Parisi, Dangyi Liu, Tomy Tsai, Toby Shevlane, Christina Kouridi, Drew Garmon, Adrian Goedeckemeyer, Adam R. Brown, Anitha Vijayakumar, Ali Elqursh, Sadegh Jazayeri, Jin Huang, Sara Mc Carthy, Jay Hoover, Lucy Kim, Sandeep Kumar, Wei Chen, Courtney Biles, Garrett Bingham, Evan Rosen, Lisa Wang, Qijun Tan, David Engel, Francesco Pongetti, Dario de Cesare, Dongseong Hwang, Lily Yu, Jennifer Pullman, Srini Narayanan, Kyle Levin, Siddharth Gopal, Megan Li, Asaf Aharoni, Trieu Trinh, Jessica Lo, Norman Casagrande, Roopali Vij, Loic Matthey, Bramandia Ramadhana, Austin Matthews, CJ Carey, Matthew Johnson, Kremena Goranova, Rohin Shah, Shereen Ashraf, Kingshuk Dasgupta, Rasmus Larsen, Yicheng Wang, Manish Reddy Vuyyuru, Chong Jiang, Joana Ijazi, Kazuki Osawa, Celine Smith, Ramya Sree Boppana, Taylan Bilal, Yuma Koizumi, Ying Xu, Yasemin Altun, Nir Shabat, Ben Bariach, Alex Korchemniy, Kiam Choo, Olaf Ronneberger, Chimezie Iwuanyanwu, Shubin Zhao, David Soergel, Cho-Jui Hsieh, Irene Cai, Shariq Iqbal, Martin Sundermeyer, Zhe Chen, Elie Bursztein, Chaitanya Malaviya, Fadi Biadsy, Prakash Shroff, Inderjit Dhillon, Tejasi Latkar, Chris Dyer, Hannah Forbes, Massimo Nicosia, Vitaly Nikolaev, Somer Greene, Marin Georgiev, Pidong Wang, Nina Martin, Hanie Sedghi, John Zhang, Praseem Banzal, Doug Fritz, Vikram Rao, Xuezhi Wang, Jiageng Zhang, Viorica Patraucean, Dayou Du, Igor Mordatch, Ivan Jurin, Lewis Liu, Ayush Dubey, Abhi Mohan, Janek Nowakowski, Vlad-Doru Ion, Nan Wei, Reiko Tojo, Maria Abi Raad, Drew A. Hudson, Vaishakh Keshava, Shubham

Agrawal, Kevin Ramirez, Zhichun Wu, Hoang Nguyen, Ji Liu, Madhavi Sewak, Bryce Petrini, DongHyun Choi, Ivan Philips, Ziyue Wang, Ioana Bica, Ankush Garg, Jarek Wilkiewicz, Priyanka Agrawal, Xiaowei Li, Danhao Guo, Emily Xue, Naseer Shaik, Andrew Leach, Sadh MNM Khan, Julia Wiesinger, Sammy Jerome, Abhishek Chakladar, Alek Wenjiao Wang, Tina Ornduff, Folake Abu, Alireza Ghaffarkhah, Marcus Wainwright, Mario Cortes, Frederick Liu, Joshua Maynez, Andreas Terzis, Pouya Samangouei, Riham Mansour, Tomasz Kępa, François-Xavier Aubet, Anton Algymr, Dan Banica, Agoston Weisz, Andras Orban, Alexandre Senges, Ewa Andrejczuk, Mark Geller, Niccolo Dal Santo, Valentin Anklin, Majd Al Merey, Martin Baeuml, Trevor Strohman, Junwen Bai, Slav Petrov, Yonghui Wu, Demis Hassabis, Koray Kavukcuoglu, Jeff Dean, and Oriol Vinyals. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context, 2024. URL https://doi.org/10.48550/arXiv.2403.05530.

Zheng Wang, Anna Cai, Xinfeng Xie, Zaifeng Pan, Yue Guan, Weiwei Chu, Jie Wang, Shikai Li, Jianyu Huang, Chris Cai, et al. Wlb-llm: Workload-balanced 4d parallelism for large language model training. In *19th USENIX Symposium on Operating Systems Design and Implementation (OSDI 25)*, pp. 785–801, 2025. URL https://dl.acm.org/doi/10.5555/3767901.3767944.

Hongtao Xu, Wenting Shen, Yuanxin Wei, Ang Wang, Guo Runfan, Tianxing Wang, Yong Li, Mingzhen Li, and Weile Jia. Skrull: Towards efficient long context fine-tuning through dynamic data scheduling. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025a. URL https://openreview.net/forum?id=WBEknRZBpT.

Peng Xu, Wei Ping, Xianchao Wu, Chejian Xu, Zihan Liu, Mohammad Shoeybi, and Bryan Catanzaro. Chatqa 2: Bridging the gap to proprietary LLMs in long context and RAG capabilities. In *The Thirteenth International Conference on Learning Representations*, 2025b. URL https://openreview.net/forum?id=cPD2hU35x3.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 Technical Report, 2025. URL https://doi.org/10.48550/arXiv.2505.09388. _eprint: 2505.09388.

Jingyang Yuan, Huazuo Gao, Damai Dai, Junyu Luo, Liang Zhao, Zhengyan Zhang, Zhenda Xie, Yuxing Wei, Lean Wang, Zhiping Xiao, Yuqing Wang, Chong Ruan, Ming Zhang, Wenfeng Liang, and Wangding Zeng. Native sparse attention: Hardware-aligned and natively trainable sparse attention. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 23078–23097, Vienna, Austria, July 2025a. Association for Computational Linguistics. ISBN 979-8-89176-251-0. URL https://doi.org/10.18653/v1/2025.acl-long.1126.

Xiulong Yuan, Hongtao Xu, Wenting Shen, Ang Wang, Xiafei Qiu, Jie Zhang, Yuqiong Liu, Bowen Yu, Junyang Lin, Mingzhen Li, Weile Jia, Yong Li, and Wei Lin. Efficient long context fine-tuning with chunk flow. In *Forty-second International Conference on Machine Learning*, 2025b. URL https://openreview.net/forum?id=rzn2OgflOK.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, et al. H2o: Heavy-hitter oracle for efficient generative inference of large language models. *Advances in Neural Information Processing Systems*, 36:34661–34710, 2023. URL https://dl.acm.org/doi/10.5555/3666122.3667628.

Yuze Zhao, Jintao Huang, Jinghan Hu, Xingjun Wang, Yunlin Mao, Daoze Zhang, Zeyinzi Jiang, Zhikai Wu, Baole Ai, Ang Wang, Wenmeng Zhou, and Yingda Chen. Swift: a scal-

able lightweight infrastructure for fine-tuning. In *Proceedings of the Thirty-Ninth AAAI Conference on Artificial Intelligence and Thirty-Seventh Conference on Innovative Applications of Artificial Intelligence and Fifteenth Symposium on Educational Advances in Artificial Intelligence*, AAAI'25/IAAI'25/EAAI'25. AAAI Press, 2025. ISBN 978-1-57735-897-8. doi: 10.1609/aaai.v39i28.35383. URL https://doi.org/10.1609/aaai.v39i28.35383.

A Background

A.1 Distributed Training Strategies

Data Parallelism (DP) Li et al. (2020) partitions the global training batch across multiple workers. In each iteration, every worker independently processes its assigned subset of data and computes the local gradients. At the end of each iteration, DP enforces a rigid synchronization barrier, requiring all workers to synchronize their gradients before updating the model weights. This inherent synchronization mechanism forces all workers to wait for the slowest worker (i.e., the straggler) to reach the barrier, making DP highly sensitive to workload imbalances.

Pipeline Parallelism (PP) Huang et al. (2019); Narayanan et al. (2019; 2021); Qi et al. (2024) splits the model layers into sequential stages, with each stage assigned to a different PP worker. To maximize hardware utilization, existing methods orchestrate model execution into a pipeline by dividing the input data into multiple micro-batches. These micro-batches then sequentially traverse all PP ranks via point-to-point communication to deliver activations and gradients between stages. However, PP inherently suffers from the pipeline bubble problem. This issue is severely exacerbated under highly heterogeneous workloads, as any execution delay in a single stage cascades throughout the entire pipeline.

Tensor Parallelism (TP) Shoeybi et al. (2020) divides the tensor operations across devices, and each device handles a slice of the tensor operations. With TP, each GPU only has part of the input and parameters, resulting in intensive communication during training. Therefore, TP is typically applied within a single node, while other levels of parallelism are applied across nodes.

Sequence Parallelism (SP) partitions the input tensor along the sequence length dimension Korthikanti et al. (2023); Jacobs et al. (2024); Liu et al. (2024); Grattafiori et al. (2024). There are three types of SP according to the communication schemes: (i) ring-based point-to-point communications Liu et al. (2024), (ii) AllToAll-based communications Jacobs et al. (2024), and (iii) AllGather-based communications Grattafiori et al. (2024). Additionally, there is another type of SP, which is proposed by Megatron-LM and splits the dropout and normalization module activation Korthikanti et al. (2023). In this paper, we use this Megatron-style SP as default.

A.2 Self-Attention and Trainable Sparse Attention

Modern LLMs are typically built by stacking many Transformer layers, each of which contains a self-attention module as the core mechanism to capture complex dependencies between elements within a sequence. In long-context training, the self-attention module becomes the dominant computational bottleneck because its cost scales quadratically with the sequence length. Given the query (Q), key (K), and value (V) matrices, where $Q, K, V \in \mathbb{R}^{N \times d}$ with N denoting the sequence length and d representing the hidden dimension. The self-attention computation is formulated as follows:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d}}\right)V \tag{5}$$

In this formulation, the dot product $QK^\top$ calculates the raw affinity scores, representing the alignment between sequence elements. These scores are subsequently scaled by $\frac{1}{\sqrt{d}}$. Then, the softmax operation is applied row-wise to yield a normalized distribution of attention weights. Finally, these weights are multiplied by the value matrix V to aggregate the relevant contextual information into the final output representation.

To alleviate the quadratic computational bottleneck of standard attention in long-context modeling, various trainable sparse attention methods have been proposed Lu et al. (2025); DeepSeek-AI et al.

(2025); Yuan et al. (2025a). These methods leverage the inherent sparsity of the attention mechanism by selecting only a subset of highly relevant tokens, referred to as *critical tokens*, to approximate the full attention computation.

Existing trainable sparse methods typically adopt a block-sparse paradigm, which partition the key and value (KV) sequences into discrete blocks. They compute the correlation between each query token and these blocks using specific routing metrics to determine the critical tokens. This type of sparse attention computation is formulated as follows:

$$\text{SparseAttn}(Q, K, V) = \text{softmax}\left(\frac{QK[I]^\top}{\sqrt{d}}\right)V[I] \tag{6}$$

where $I \in \mathbb{R}^N$ serves as the indexer indicating the set of selected critical tokens.

The primary distinction among existing algorithms lies in the design of this routing mechanism, also referred to as the indexer. For instance, MoBA Lu et al. (2025) identifies critical tokens by calculating the relevance between each query and the mean representation of the key blocks. Similarly, DSA DeepSeek-AI et al. (2025) employs a lightweight, low-precision indexer based on multi-head latent attention. Despite differing indexer designs, these block-sparse algorithms uniformly select a fixed Top-K most relevant token blocks for computation across all samples. Notably, our method reuses the inherent indexer of these sparse algorithms to achieve workload-aware dynamic sparsity tuning with negligible additional overhead. Consequently, our approach can be seamlessly integrated into existing sparse attention algorithms. In this paper, we also use the *attention budget* k (e.g., the Top-K selected blocks) to characterize the sparsity level. A smaller attention budget k indicates higher sparsity, while a larger k indicates lower sparsity.

B Further design details

B.1 Feasible budget selection

For bottleneck micro-batches, DST further restricts the candidate budgets to those that simultaneously satisfy the latency target and the quality bound. We define the feasible set as

$$\mathcal{F}_i = \left\{ k \in \mathcal{K} \;\middle|\; \hat{T}_i(k) \le T_{\text{anchor}},\ C_i(k_i^{base}) - C_i(k) \le p \right\} \tag{7}$$

Accordingly, the final accepted budget is given by

$$k_i^{final} = \begin{cases} \max \mathcal{F}_i, & \hat{T}_i(k_i^{base}) > T_{\text{anchor}} \\ k_i^{anchor}, & \hat{T}_i(k_i^{base}) \le T_{\text{anchor}} \end{cases} \tag{8}$$

B.2 Latency predictor interfaces

The predictor `Perf` exposes two runtime interfaces:

$$\hat{T}(x, k) = \texttt{Perf.predict}(x, k) \tag{9}$$

$$k^{\text{target}} = \texttt{Perf.align}(x, T_{\text{target}}) \tag{10}$$

where x denotes the length descriptor of a micro-batch and k denotes the attention budget. The first interface estimates the practical latency under budget k, while the second returns the largest budget whose predicted latency does not exceed the target latency T_{target}.

To materialize these interfaces efficiently, we adopt a ***profiling-based latency modeling*** approach. Specifically, we use a single Transformer layer as a performance proxy, discretize sequence lengths into bins $\mathcal{X}$ and attention budgets into a small candidate set $\mathcal{K}$, and profile the latency of every pair $(x, k) \in \mathcal{X} \times \mathcal{K}$. The resulting lookup table is

$$\mathcal{M}[x, k] = \texttt{profile}(x, k) \tag{11}$$

where $\mathcal{M}$ stores the measured latency on the target system. At runtime, `Perf.predict` queries $\mathcal{M}$ directly, optionally with interpolation when the input falls between neighboring buckets, while

`Perf.align` searches over $\mathcal{K}$ and returns the budget whose predicted latency best matches the target.

This design keeps the online overhead negligible, because runtime only requires lightweight table lookup and search over a small candidate set. Moreover, the profiled table remains reusable as long as the hardware environment, parallelization strategy, and sparse attention implementation remain unchanged. Based on this predictor, Algorithm 1 summarizes the runtime workflow of DST. In addition, we later reuse the same latency predictor in SAB to guide latency-based batching.

Algorithm 1 Workload-Aware Dynamic Sparsity Tuning (DST)

Require: Threshold p, anchor strategy $anchor$, current micro-batch length x_i, global-batch length list $\{x_j\}_{j=1}^{N}$, base budget list $\{k_j^{base}\}_{j=1}^{N}$, routing logits $logits_i$, latency predictor $Perf$

Ensure: Final attention budget k_i^{final} for current micro-batch

$T_{list} \leftarrow [\,]$
for each (x_j, k_j^{base}) in $\{(x_j, k_j^{base})\}_{j=1}^{N}$ **do**
    T_{list}.`append`(`Perf.predict`(x_j, k_j^{base}))
end for
$T_{\text{anchor}} \leftarrow$ `SelectAnchor`$(T_{list}, anchor)$
$k_i^{anchor} \leftarrow$ `Perf.align`(x_i, T_{anchor})
$k_i^{final} \leftarrow k_i^{anchor}$
$C_i \leftarrow$ `AccumulateNormScores`$(logits_i)$
if $C_i[k_i^{base}] - C_i[k_i^{anchor}] > p$ **then**
    $k_i^{final} \leftarrow$ `FindFeasibleK`(C_i, k_i^{base}, p)
end if
return k_i^{final}

B.3 Sparsity-aware batching procedure

Algorithm 2 Sparsity-Aware Batching (SAB)

Require: Input batch B, global batch size gbs, micro-batch size mbs, data parallel size N, current DP rank dp_i, sequence lengths L, sparsity predictor $Spar$, latency predictor $Perf$

Ensure: Local micro-batch index bins for the current DP rank

$W \leftarrow \emptyset$
for each sample index $i \in B$ **do**
    $\tilde{k}_i \leftarrow$ `Spar.predict`$(L[i])$
    $W[i] \leftarrow$ `Perf.predict`$(L[i], \tilde{k}_i)$
end for
$DP_Bins \leftarrow$ `bin_packing`(W, N)
$I_{dp} \leftarrow DP_Bins[dp_i]$
$MB_Bins \leftarrow$ `bin_packing`$(W[I_{dp}],\ gbs/(mbs \cdot N))$
return MB_Bins

C Additional experiments

C.1 Dataset and model effects

First, we analyze the performance variances between the two datasets. In terms of end-to-end speedup, LongAlign achieves higher overall gains compared to ChatQA2-Long-SFT. However, when breaking down the contributions, LongAlign benefits significantly more from the DST module, whereas its gains from the SAB module are noticeably lower. We attribute this discrepancy to the distinct length distributions and intrinsic characteristics of the datasets. As discussed in Section 2, sequence lengths in LongAlign are relatively concentrated between 8K and 16K tokens. Conversely, ChatQA2-Long-SFT exhibits a clear bimodal distribution, with the majority of sequences being either shorter than 4K or longer than 16K tokens. Because of this concentrated distribution, LongAlign is less sensitive to

batching strategies. In contrast, the extreme length variance in ChatQA2-Long-SFT allows the SAB module to deliver larger optimizations. Furthermore, as the optimization potential of the DST module is highly correlated with data complexity, its varying performance indicates the distinct characteristics of these two datasets.

Next, we examine the impact of different training configurations. We observe that the 3B model achieves a higher speedup ratio than the 0.5B model. This is because the attention mechanism constitutes a more severe computational bottleneck in the 3B model, amplifying the load imbalance issues. Consequently, optimizing the 3B model yields more efficiency benefits. Additionally, the 3B LoRA configuration exhibits speedups similar to the standard 3B model, demonstrating the versatility and robustness of SparseBalance across different training paradigms.

C.2 Ablation Study

To evaluate the impact of the key hyper-parameters in SparseBalance, we conduct an ablation study on the sparsity threshold p, the execution anchor choice, and the micro-batch size using the Qwen2.5-3B model. We test the speedup ratio under a configuration of $\text{DP} = 2, \text{PP} = 4, \text{TP} = 1, \text{SP} = 1$ in this section.

First, we study the impact of p and anchor choice. With the anchor fixed to MEAN, increasing p from 0.05 to 0.30 steadily improves the end-to-end speedup from $1.074\times$ to $1.451\times$, indicating that a looser threshold enables more aggressive optimization. Second, we evaluate the two extreme anchor choices, MIN and MAX, which align the workload to the predicted shortest and longest micro-batch latencies, respectively. The range of bars delineates the optimization potential within this dimension. Last, Figure 10 further shows that SparseBalance consistently achieves notable speedups across all tested micro-batch sizes, demonstrating its robustness to different batching sizes.

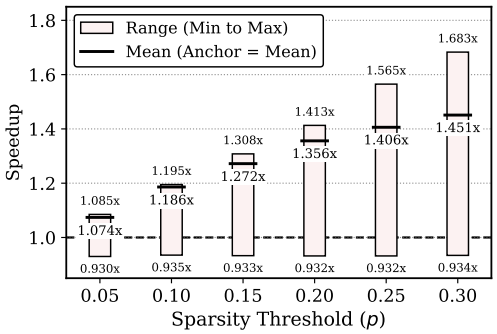

Figure 9: Sensitivity of end-to-end speedup with respect to the sparsity threshold p and execution anchor choice. The upper end of each bar corresponds to MIN, the lower end corresponds to MAX, and the horizontal line denotes MEAN.

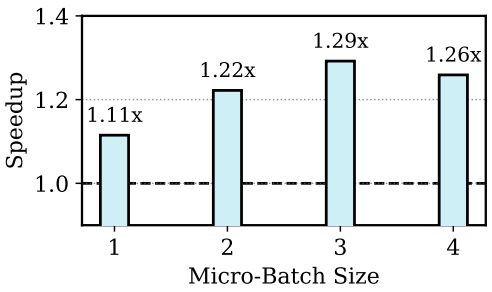

Figure 10: Sensitivity of end-to-end speedup to different micro-batch sizes ($mbs = 1$ to 4, corresponding to global batch sizes $gbs = 8$ to 32). SparseBalance consistently achieves clear speedups across all tested settings.

C.3 Training Loss

To validate the model convergence and accuracy, we conduct long-context fine-tuning on the Qwen-2.5-3B model using the ChatQA2 dataset. We employ MoBA with a block size of 256 and top-k 32 as our baseline. For SparseBalance, we evaluate configurations with $\text{MIN}_{0.1}$, $\text{MIN}_{0.2}$, $\text{MEAN}_{0.1}$, and $\text{MEAN}_{0.2}$. We exclude the MAX configurations, as they severely degrade training efficiency without practical benefits.

As illustrated in Figure 11, the training loss curves of SparseBalance closely track the baseline. Although minor deviations appear during the early training stages—likely stemming from the instability of transitioning from dense pre-training to sparse fine-tuning—the gap rapidly diminishes. Notably, the configuration with $p = 0.2$ exhibits a slightly higher loss than $p = 0.1$. This underscores the necessity of the performance-bound mechanism in our DST module, which effectively prevents excessive critical information dropping in sensitive sequences. Furthermore, setting the anchor MEAN

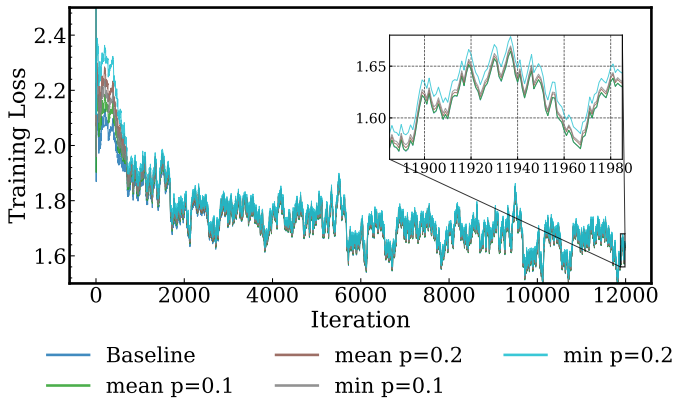

Figure 11: Training loss comparison between the MoBA baseline and SparseBalance with different dynamic tuning configurations.

tends to yield lower loss compared to targeting the minimum latency MIN. This is consistent with the design intuition of bidirectional tuning in SparseBalance.

C.4 DETAILED BENCHMARK RESULTS

Table 2: Average accuracy (%) of Needle-in-a-Haystack (NIAH) tasks evaluated on Qwen-2.5-3B model. Results are averaged across the *niah_single_1*, *niah_single_2*, and *niah_single_3* evaluations for each model.

Model	**≤8K**	**16K**	**32K**	**64K**
Baseline SFT	100.00	100.00	100.00	**97.60**
$\text{MIN}_{0.1}$	100.00	100.00	99.93	96.33
$\text{MIN}_{0.2}$	100.00	100.00	99.93	95.20
$\text{MEAN}_{0.1}$	100.00	100.00	100.00	97.53
$\text{MEAN}_{0.2}$	100.00	100.00	99.93	96.47

Table 3: General benchmark results on Qwen2.5-3B. **Bold** indicates the best performance.

Task	**Base-SFT**	$\textbf{MIN}_{0.1}$	$\textbf{MIN}_{0.2}$	$\textbf{MEAN}_{0.1}$	$\textbf{MEAN}_{0.2}$
ARC-Chal.	0.4727	0.4735	0.4684	**0.4744**	0.4659
ARC-Easy	0.7332	**0.7348**	0.7277	0.7315	0.7260
BoolQ	0.7936	0.7994	0.7951	0.8003	**0.8031**
HellaSwag	0.7356	0.7354	0.7355	0.7349	**0.7366**
Lambada	0.6598	0.6583	0.6594	**0.6604**	0.6592
Piqa	0.7884	0.7867	0.7894	**0.7911**	0.7900
Winogrande	**0.6875**	0.6851	0.6835	0.6843	0.6859
Average	0.6958	0.6962	0.6941	**0.6967**	0.6952

Table 4: Category-level average results on LongBench for Qwen2.5-3B.

Category	Base-SFT	MIN$_{0.1}$	MIN$_{0.2}$	MEAN$_{0.1}$	MEAN$_{0.2}$
Single-Doc QA	32.24	**33.86**	33.68	32.93	32.99
Multi-Doc QA	34.18	35.56	35.04	**35.84**	35.41
Summarization	**26.14**	24.71	24.92	25.63	24.38
Retrieval	**49.08**	47.10	48.30	48.34	49.00
Code	**67.41**	66.44	66.52	66.54	66.24
Overall	39.28	39.19	39.28	**39.46**	39.14

D ADDITIONAL RELATED WORK

System Optimization in Long-Context Training. To enhance the training efficiency of long-context LLMs, existing systems primarily rely on hybrid parallelism strategies Shoeybi et al. (2020); Narayanan et al. (2021); Korthikanti et al. (2023); Smith et al. (2022), which combine data, pipeline, tensor, and sequence parallelism. However, the highly heterogeneous sequence length distribution in real-world long-context data inherently leads to severe load imbalance across workers. To mitigate this issue, various batching and scheduling techniques have been proposed Wang et al. (2025); Bai et al. (2024); Xu et al. (2025a); Yuan et al. (2025b). For example, LongAlign adopts sorted batching to group sequences by length within a global batch Bai et al. (2024), while WLB-LLM employs a heuristic packing approach that adaptively delays the execution of long sequences Wang et al. (2025). Despite their effectiveness in improving system throughput, these methods mainly optimize data organization and may compromise data randomness, potentially raising the risk of changing the convergence behavior. Furthermore, existing works are largely designed for dense attention training and fall short in sparse scenarios. As a result, they do not address the additional workload uncertainty introduced by dynamic sparse attention.

Sparse Attention Training. Sparse attention has become a promising direction for reducing the quadratic cost of long-context modeling. Early work mainly focused on training-free sparse methods for inference acceleration Zhang et al. (2023); Tang et al. (2024), while recent efforts have shifted toward trainable sparse attention mechanisms. Representative methods such as NSA Yuan et al. (2025a), MoBA Lu et al. (2025), and DSA DeepSeek-AI et al. (2025) accelerate training by selecting critical tokens or blocks with Top-K-style mechanisms. However, these methods typically use a fixed attention budget across sequences and layers, overlooking the substantial heterogeneity in sparsity sensitivity during training. More recent work has begun to explore dynamic sparse attention. For instance, Twilight Lin et al. (2025) introduces hierarchical Top-P selection for adaptive budget allocation, while MTraining Li et al. (2025) combines dynamic sparsity guided by a vertical-slash structure with context-parallel execution optimization. Nevertheless, these methods primarily treat dynamic sparsity as an algorithmic mechanism, while SparseBalance further makes sparsity control system-aware by coupling runtime tuning with profiling-based latency modeling and workload balancing.