A Momery-balanced Micro-batch Chunking in Sequence Blaster

We illustrate the memory-balanced micro-batch chunking algorithm in Sequence Blaster based on dynamic programming. Specifically, given a batch of sequences $\mathcal{B} = \{\mathcal{S}_k\}$ that has already been sorted according to takeaway #2, we split them into consecutive M micro-batches, and micro-batch $\mathcal{M}_i$ contains all sequences k satisfying $j_{i-1} \le k < j_i$, where j_i is the ending indices for $\mathcal{M}_i$ ($j_0 = 0$). To balance the token amount of each micro-batch, we aim to minimize the maximum total token number of each micro-batch as follows:

$$\arg\min_{\{j_i\}} \max_{i\in[1,M]} \{ \sum_{k\in[j_{i-1},j_i)} s_k \}. \tag{23}$$

Again, we solve the problem via dynamic programming. Denote $DP[k][i]$ as the optimal value when blasting the first k sequences into i micro-batches. Starting with $DP[0][0] = 0$, we can solve the problem via the following state transition formula:

$$DP[k][i] = \min_{j\in[i-1,k-1]} \{\max\{DP[j][i-1], \sum_{l\in[j+1,k]} s_l\}\}, \tag{24}$$

where $\sum_{l\in[j+1,k]} s_l$ represents the total token number of the i^{th} micro-batch when splitting micro-batch at the $\mathcal{S}_j$. $DP[K][M]$ denotes the optimal solution, and the optimized values of j_i splits the global batch data into M micro-batches with balanced memory consumption.

B Details of Experimental Setups

B.1 Model Configuration

Tab. 5 presents the specific configurations of the GPT-7B, 13B and 30B models used in our experiments. The parameter number of each model in Tab. 5 is under maximum context length of 384K, where the positional embedding contains 1-2 billion parameters.

Table 5. Model configuration (384K max context length).

Model	# Layers	# Param	Hidden Dim
GPT-7B	32	7.85B	4096
GPT-13B	40	14.03B	5120
GPT-30B	60	32.72B	6656

B.2 Protocols

For fair comparison, we manually tune the most efficient parallelism strategies for all baseline systems under different workloads. For DeepSpeed, the optimal strategy is usually among SP=64 or SP=32 with ZeRO-3, and for Megatron-LM, the optimal strategy is usually among TP=8, CP=8, or TP=16, CP=4, or TP=8, CP=4, DP=2 with ZeRO-1.

We also apply activation checkpointing strategies for each system to make sure all systems can fit the models without out-of-memory issues. For GPT-7B, activation checkpointing is unnecessary to support a 384K context length on 64 GPUs. For GPT-13B, we only checkpoint MLP layers, while for GPT-30B, almost all layers are checkpointed to support 384K context length.

C Estimation Accuracy of Cost Models

We evaluate the accuracy of the cost estimator (§4.1.2) utilized in FlexSP across diverse configurations (as shown in Tab. 1), including sequence parallelism degree, batch size, and sequence length. Fig. 9 compares the deviation of the estimated cost and the empirical execution time. As can be seen, our overhead estimator adeptly approximates the execution overhead, with discrepancies consistently remaining below 5%. The accurate estimations rendered by the estimator facilitates performance of our system.

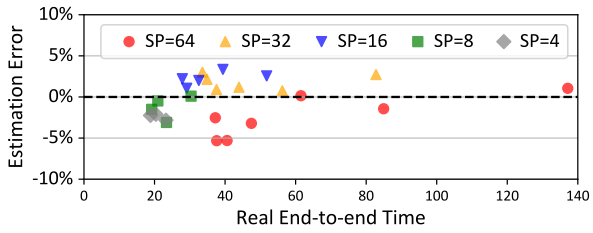

Figure 9. Estimation accuracy.

D Performance Analysis of SOTA Systems

Here we analyze the performance of SOTA systems, i.e., DeepSpeed and Megatron-LM. As shown in Fig. 4, in most cases, DeepSpeed has similar or better performance than Megatron-LM. This is attributed to the different mechanism of CP in Megatron-LM and SP in DeepSpeed, as well as the skewness of datasets. CP usually has much more communication volumn than the All-to-All in SP. Although CP leverages the overlap between attention computation and KV transmission to hide the communication overhead, in scenarios with limited inter-node bandwidth and a majority of short sequences in datasets, the attention computation often fails to hide the communication. Therefore, Megatron-LM's performance is usually constrained by its higher communication volume compared to DeepSpeed and FlexSP.

E Integrating Context Parallelism

Context parallelism [5, 23, 26, 28] is also an important technique for long context training, as illustrated in §2.1.3. As a different paradigm to scatter sequences across devices, CP is orthogonal to our work, and can be integrated into FlexSP. Specifically, similar to FlexSP utilizing ZeRO and flexible SP, we can employ TP, ZeRO and CP, fix the parallelism degree of TP, and employ the flexible sequence parallelism strategy of FlexSP to achieve flexible CP, which adaptively adjusts the CP group size according to the varied-length sequences. We'll integrate CP into our system as the future work.