Enabling Parallelism Hot Switching for Efficient Training of Large Language Models

Hao Ge*, Fangcheng Fu*, Haoyang Li, Xuanyu Wang, Sheng Lin, Yujie Wang, Xiaonan Nie, Hailin Zhang, Xupeng Miao(Purdue University), Bin Cui

PKU, Purdue University

December, 12th

1. Background
2. HotSPa
3. Experiments
4. Conclusions

Background-Transformers and Large Language Models

- Padding and Packing
  - In order to train sequences with di erent lengths together, techniques such as padding or packing are needed to preprocess the sequences

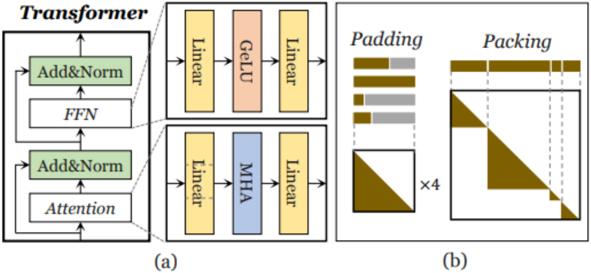


Figure 1. Illustration of (a) the architecture of Transformer layer; (b) sequence padding and packing.

Background- Deep Learning Training

- Computation Graph
- Mixed-precision Training
- Gradient Accumulation

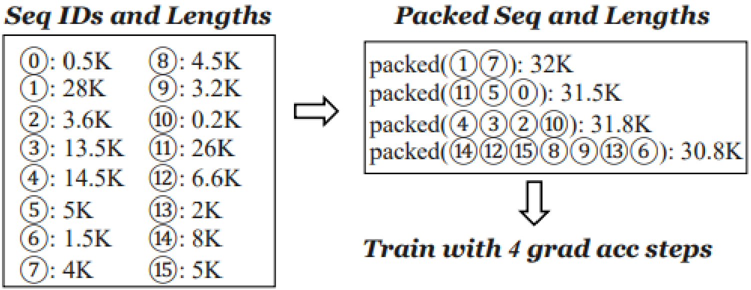


Figure 2. An example of training with gradient accumulation and sequence packing when the maximal supported length is 32K.

Background-Gradient Accumulation

- Data Parallelism (DP)
- Model Parallelism (MP)
- Sequence Parallelism (SP)
- Hybrid Parallelism

Data Parallelism (DP)

- Introduction
  - Each device maintains a full copy of model states execute locally and synchronize the model gradients globally.
- Need an extra round of all-gather communication
  - For example, when there are p micro-batches, the communication cost of ZeRO-3(FSDP) would be $p + \frac{1}{2}$ times of that of conventional DP.

Model Parallelism (MP)

- Tensor parallelism (TP)
  - Two parameters within one self-attention or FFN block are split in the column- and row-wise.
  - 4 all-reduce communication for one Transformer layer.
- Pipeline parallelism (PP)
  - A model is reckoned as a sequence of layers, and divided into multiple stages across devices.
  - P2P communication operations are needed to transfer the intermediate results.

Sequence Parallelism (SP)

A special form of DP

- SP splits the training samples (sequences) in the sequence dimension

Hybrid Parallelism

Combine DP, TP, and PP and researchers can tune the parallelism degree of each strategy for a given task to achieve better efficiency.

Motivation

- Two popular datasets for LLMs, CommonCrawl and Github, show significant skewness in the sequence lengths.
- This skewness leads to inefficiencies when applying static parallelism strategies in training.

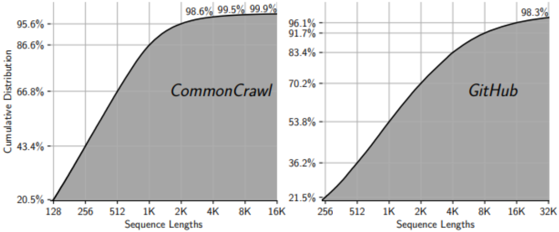
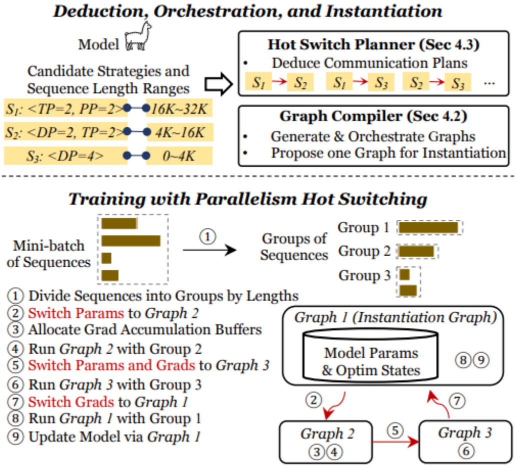


Figure 3. Cumulative distributions of sequence lengths.

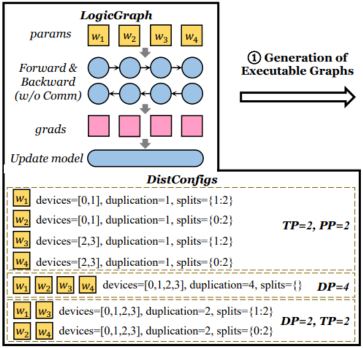
Overview


Deduction, Orchestration, and Instantiation
Model
Candidate Strategies and Sequence Length Ranges
S1: <TP=2, PP=2> 16K~32K
S2: <DP=2, TP=2> 4K~16K
S3: <DP=4> 0~4K
Hot Switch Planner (Sec 4.3)
Deduce Communication Plans
S1 → S2 S1 → S3 S2 → S3 ...
Graph Compiler (Sec 4.2)
Generate & Orchestrate Graphs
Propose one Graph for Instantiation
Training with Parallelism Hot Switching
Mini-batch of Sequences
1
Groups of Sequences
Group 1
Group 2
Group 3
1 Divide Sequences into Groups by Lengths
2 Switch Params to Graph 2
3 Allocate Grad Accumulation Buffers
4 Run Graph 2 with Group 2
5 Switch Params and Grads to Graph 3
6 Run Graph 3 with Group 3
7 Switch Grads to Graph 1
8 Run Graph 1 with Group 1
9 Update Model via Graph 1
Graph 1 (Instantiation Graph)
Model Params & Optim States
8 9
2
7
Graph 2
3 4
5
Graph 3
6

Graph Compilation

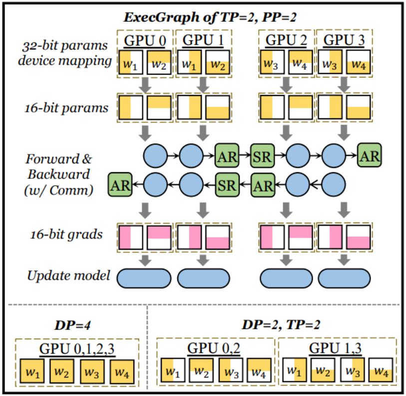
- Generation of Logical Graph
- Generation of Executable Graphs
- Orchestration of Executable Graphs

Logical Graph



Generate *DistConfigs* based on candidate parralelism strategies,the assigned devices, the number of duplications, and a map to indicate how a multi-dimensional parameter is split.

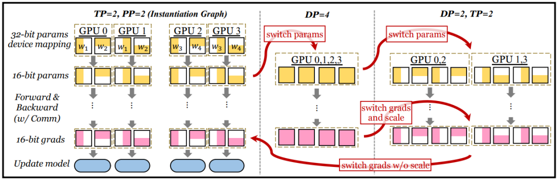
Generation of Executable Graphs



Generate *ExecGraphs* considering three types of insertions:

- For model parameters, the type casting operators are inserted.
- Inside the forward and backward propagation, communication operators.
- For model gradients, accumulation operators are inserted.

Orchestration of Executable Graphs



- Firstly, we need to identify the candidate ExecGraphs that minimize the memory occupation.
- Secondly, the ExecGraphs are re-ordered for better efficiency.
- Thirdly, prune the type casting and model update operators of the other ExecGraphs like cast model parameter to the 16-bit version.

Model Hot Switching

Two key characteristics:

- Intra-node communication is preferable than inter-node: In typical GPU clusters, GPUs within a node are connected by NVLink, which has a higher communication bandwidth than Infini-band or Ethernet.
- GPU connectivities are full-duplex:it is reasonable to minimize the maximum sending volume of all devices.

Algorithm 1: Our heuristic hot switching planner.

Initialize hot switching plan $\mathcal{P} = \{\}$;
Initialize intra- and inter-node communication volume $V_i^{(inter)}, V_i^{(intra)}$ as 0 for each device i;
foreach *model parameter/gradient slice* **do**
    Determine the owner (source) devices S;
    Determine the target (destination) devices D;
    foreach *dst in D* **do**
        if *$dst \notin S$* **then**
            Partition S into $S^{(intra)}, S^{(inter)}$;
            if *$S_i^{(intra)}$ is not empty* **then**
                $src \leftarrow \arg\min_i \{V_i^{(intra)} \mid i \in S^{(intra)}\}$;
                $V_{src}^{(intra)} \leftarrow V_{src}^{(intra)} + \text{sizeof}(slice)$;
            else
                $src \leftarrow \arg\min_i \{V_i^{(inter)} \mid i \in S^{(inter)}\}$;
                $V_{src}^{(inter)} \leftarrow V_{src}^{(inter)} + \text{sizeof}(slice)$;
            $\mathcal{P} \leftarrow \mathcal{P} \cup (slice, src, dst)$;
return $\mathcal{P}$;

1. Background
2. HotSPa
3. **Experiments**
4. Conclusions

Case Studies

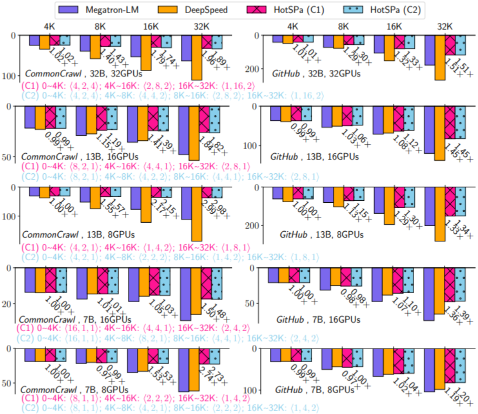


Figure 7. End-to-end evaluation (measured in seconds per mini-batch). We present two hot switching combinations for each experiments and the corresponding speedups compared with baselines (format: "seq_len_range: ⟨*DP*, *TP*, *PP*⟩").

Case Studies

Table 3. Case studies of training with different combinations of parallelism strategies (32B, 32GPUs, *s*=32K). We dissect the running time for different groups of sequence lengths. "Others" includes the time cost of hot switching in one mini-batch.

<table>
<tr><th rowspan="2">Breakdown</th><th colspan="4">CommonCrawl (time in seconds)</th><th colspan="4">GitHub (time in seconds)</th></tr>
<tr><th>Static</th><th>C1</th><th>C2</th><th>C3</th><th>Static</th><th>C1</th><th>C2</th><th>C3</th></tr>
<tr><td>0~1K</td><td>42.1</td><td rowspan="2">22.7 (2.40×)</td><td rowspan="2">22.7 (2.40×)</td><td>12.8 (3.28×)</td><td>34.9</td><td rowspan="2">26.3 (2.78×)</td><td rowspan="2">26.3 (2.78×)</td><td>8.6 (4.05×)</td></tr>
<tr><td>1K~4K</td><td>12.5</td><td>8.2 (1.52×)</td><td>38.4</td><td>17.1 (2.24×)</td></tr>
<tr><td>4K~8K</td><td>3.5</td><td rowspan="2">5.4 (1.18×)</td><td>2.7 (1.29×)</td><td>2.7 (1.29×)</td><td>25.0</td><td rowspan="2">37.4 (1.40×)</td><td>15.4 (1.62×)</td><td>15.4 (1.62×)</td></tr>
<tr><td>8K~16K</td><td>2.9</td><td>2.2 (1.31×)</td><td>2.2 (1.31×)</td><td>27.3</td><td>20.2 (1.35×)</td><td>20.2 (1.35×)</td></tr>
<tr><td>16K~32K</td><td>2.4</td><td>2.4 (1.00×)</td><td>2.4 (1.00×)</td><td>2.4 (1.00×)</td><td>53.0</td><td>53.0 (1.00×)</td><td>53.0 (1.00×)</td><td>53.0 (1.00×)</td></tr>
<tr><td>Others</td><td>-</td><td>1.8</td><td>3.5</td><td>4.9</td><td>-</td><td>1.8</td><td>3.5</td><td>4.9</td></tr>
<tr><td>Total</td><td>63.4</td><td>32.3 (1.96×)</td><td>33.5 (1.89×)</td><td>33.2 (1.90×)</td><td>178.6</td><td>118.5 (1.50×)</td><td>118.4 (1.50×)</td><td>119.2 (1.49×)</td></tr>
</table>

(Static) 0~32K: ⟨1, 16, 2⟩ (C1) 0~4K: ⟨4, 2, 4⟩; 4K~16K: ⟨2, 8, 2⟩; 16K~32K: ⟨1, 16, 2⟩

(C2) 0~4K: ⟨4, 2, 4⟩; 4K~8K: ⟨4, 4, 2⟩; 8K~16K: ⟨2, 8, 2⟩; 16K~32K: ⟨1, 16, 2⟩

(C3) 0~1K: ⟨8, 4, 1⟩; 1K~4K: ⟨4, 2, 4⟩; 4K~8K: ⟨4, 4, 2⟩; 8K~16K: ⟨2, 8, 2⟩; 16K~32K: ⟨1, 16, 2⟩

Model Hot Switching

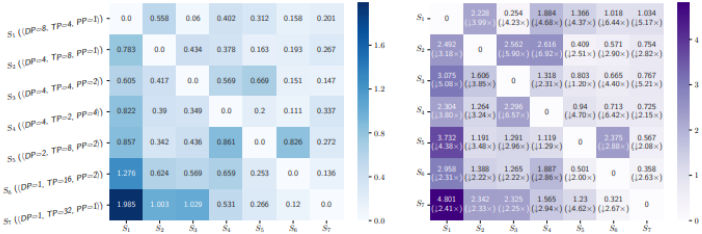


Figure 8. Time cost (in seconds) of switching between parallelism strategies (LLaMA2-32B, 32GPUs). The value in the i-th row and j-th column indicates switching from S_i to S_j. (Left: time cost of our work. Right: time cost without the optimizations in Section 4.3.)

1. Background
2. HotSPa
3. Experiments
4. **Conclusions**

Future Work

Limitations:

- Reliance on user-provided combinations of strategies
- When the number of gradient accumulation steps is large, the hot switching overhead can be amortize
- Many parallelism strategies that are not incorporated

Thanks!