Main project Report

Name: SoonHo KIM
Student ID: 20200703

Overview

This report explains the implementation of an automatic search algorithm for selecting the optimal 3D parallelism strategy—data, tensor, and pipeline parallelism—for deep learning model training. The core of the implementation lies in the Searcher class, which uses a cost model to estimate per-iteration time and selects the most efficient configuration based on the results.

Constants Used

The following constants are used for performance modeling:

BASE_EFF = 0.35: Base single-GPU efficiency

DP_EFF_PENALTY_FACTOR = 0.20: Efficiency loss per increase in DP degree

TP_EFF_PENALTY_FACTOR = 0.25: Efficiency loss per increase in TP degree

OVERLAP_RATIO_PP = 0.60: Overlap factor for 1F1B pipeline schedule

Function Descriptions

1. estimate_compute_time(dp, tp, pp, total_tflops, gpu_flops)

Purpose:
Estimates the compute time (in milliseconds) required for one training iteration using a given (dp, tp, pp) configuration.

Key Logic:

Adjusts compute efficiency based on the DP/TP values and their imbalance.

Calculates total system FLOP throughput: $gpu\_flops \times dp \times tp$.

Applies overlap ratio for pipeline parallelism when pp > 1.

2. estimate_dp_time(dp_sum_gradient_MB, dp, n_gpu_per_node, bw_gpu_to_gpu, bw_inter_node)

Purpose:
Estimates the communication time needed to synchronize gradients across data parallel replicas.

Key Logic:

Uses ring AllReduce formula:

$$2 \times \frac{dp - 1}{dp} \times gradient\ size$$

Chooses bandwidth based on whether all DP GPUs are within a single node.

3. estimate_tp_time(tp_comm_sum_MB, tp, bw_gpu_to_gpu)

Purpose:
Estimates the communication time for synchronizing intermediate results in tensor parallelism.

Key Logic:

Also uses a ring AllReduce communication model.

Communication is scaled by the number of tensor partitions.

4. estimate_pp_time(layer_activation_MB, pp, bw_gpu_to_gpu)

Purpose:
Estimates communication cost due to pipeline parallelism between stages.

Key Logic:

Assumes only (pp - 1) stage boundaries.

Uses the maximum activation size and assumes two-way communication per boundary.

5. evaluate_config((dp, tp, pp, cluster_metadata, model_metadata))

Purpose:
Calculates total estimated time (in ms) for one iteration with the given (dp, tp, pp) configuration.

Steps:

Estimates compute, DP, TP, and PP times individually.

Returns the total sum.

6. search(cluster_metadata, model_metadata, output_file="output.json")

Purpose:
Searches the configuration space for the optimal (dp, tp, pp) setting that minimizes iteration time.

Steps:

1. Generates all valid combinations such that dp × tp × pp = total # of GPUs.
2. Uses ProcessPoolExecutor for parallel evaluation of configurations.
3. Selects and returns the configuration with the lowest total time.
4. Writes all configuration results to a JSON file for review.

Conclusion

This implementation allows efficient exploration of 3D parallelism strategies for deep learning model training. By modeling both compute and communication costs, and balancing across DP, TP, and PP dimensions, it identifies configurations that are likely to minimize training time. The modular structure also allows easy extension for new parallelism models or cost functions.