Appendix

A. Notation table

Table 5 summarizes the main notations used throughout the paper.

Symbol	Description
m_a	the activation memory cost
m_s	the memory cost of model states
ps	parameter size
ts	TP size
fs	FSDP size
c_{dtype}	a constant dependent on mixed precision choice
m_c	context memory cost
deg	the number of computation stages
c	the number of micro-batches
tpi_{gpipe}	TPI in GPipe
m	the memory limit for each device
B	mini-batch size
n	the number of GPUs
fp_i	forward computation time for computation stage i
bp_i	backward computation time for computation stage i
fo_j	forward communication time for communication stage j
bo_j	backward communication time for communication stage j
p_i	the cost for computation stage i
o_j	the cost for communication stage j
$\boldsymbol{M}_{uk}$	the memory cost for the k-th intra-layer strategy of layer u on a single device
$\boldsymbol{A}_{uk}$	execution cost for the k-th intra-layer strategy of layer u
$\boldsymbol{S}_{uk}$	whether the k-th parallel strategy is selected for the layer u
$\boldsymbol{P}_{ui}$	whether layer u is placed on the i-th computation stage
$\boldsymbol{R}_{uv}$	resharding cost between layers u and v when they are located within the same pipeline stage
$\boldsymbol{R}'_{uv}$	resharding cost between layers u and v when they are located across consecutive stages
$\boldsymbol{Z}$	the auxiliary variable used for order-preserving constraint
$\mathbb{P}$	the set of cost for computation stages, which is $\{p_1, p_2, \ldots, p_{deg}\}$
$\mathbb{O}$	the set of cost for communication stages, which is $\{o_1, o_2, \ldots, o_{deg-1}\}$
$\mathbb{S}_u$	the set of intra-layer parallel strategies for layer u
$\mathcal{G}(\mathbb{V}, \mathbb{E})$	the computation graph for the model

Table 5. Summary of the main notations.

B. Proof of the linear form for the contiguous set

To facilitate our discussion, we adopt the linear form of the order-preserving constraint as presented in the main paper. We denote $\boldsymbol{P}_{ui}$ as a 0-1 variable indicating whether layer u is to be placed on the i-th computation stage, pp_size as the number of computation stages in the pipeline. Besides, $\mathcal{G}(\mathbb{V}, \mathbb{E})$ represents the computation graph for the model. Then, we formalize the theorem as follows:

Theorem B.1. *A subgraph with node set $\mathbb{V}_i = \{\forall u \in \mathbb{V} : \boldsymbol{P}_{ui} = 1\}$ is contiguous if and only if there exists $\boldsymbol{Z}_{vi}$ such that Equation (6a), (6b), and (6c) are satisfied.*

Previous work [38] has proven this theorem. Our proof draws on the process of this work. The details of the proof are as follows:

Proof. "If": Assume that there exists nodes $u, w \in \mathbb{V}_i$ and $v \notin \mathbb{V}_i$ such that v and w are reachable from u and v, respectively. Hence, $\boldsymbol{P}_{ui} = 1$, $\boldsymbol{P}_{wi} = 1$, and $\boldsymbol{P}_{vi} = 0$. Without losing generality, we assume $\langle u, v\rangle \in \mathbb{E}$. Thus, according to Equation (6c), we have $\boldsymbol{Z}_{vi} \leqslant \boldsymbol{P}_{vi} - \boldsymbol{P}_{ui} + 1 = 0$. By applying Equation (6b) repeatedly following the path from v to w, we have $\boldsymbol{Z}_{wi} \leqslant \boldsymbol{Z}_{vi}$. Thus, $\boldsymbol{Z}_{wi} \leqslant 0$. However, we also have $\boldsymbol{Z}_{wi} \geqslant \boldsymbol{P}_{wi} = 1$ according to Equation (6a). A contradiction.

"Only if": First, we define $\boldsymbol{Z}_{vi} = 1$ if a node $w \in \mathbb{V}_i$ is reachable from v $(v \in \mathbb{V})$. Otherwise, $\boldsymbol{Z}_{vi} = 0$. Thus, Equation (6a) and (6b) are satisfied according to this kind of definition. For Equation (6c), if $\boldsymbol{P}_{vi} = 1$, the constraint will hold true regardless of whether $\boldsymbol{P}_{ui}$ is 1 or 0. If $\boldsymbol{P}_{vi} = 0$ and $\boldsymbol{P}_{ui} = 0$, $\boldsymbol{Z}_{vi} \leqslant \boldsymbol{P}_{vi} - \boldsymbol{P}_{ui} + 1 = 1$ will also hold true because $\boldsymbol{Z}_{vi}$ could be either 0 or 1. Finally, if $\boldsymbol{P}_{vi} = 0$ and $\boldsymbol{P}_{ui} = 1$, $\boldsymbol{Z}_{vi} = 0$ will hold true because $\mathbb{V}_i$ is a contiguous set and we cannot find any $w \in \mathbb{V}_i$, such that w is reachable from v. □

C. QIP formulation for intra-layer-only parallelism

Here we present the QIP formulation for intra-layer-only parallelism with explanations.

Objective function In terms of intra-layer-only parallelism, there is only one computation stage involved. As a result, the objective function takes into account only the value of p_1. We hereby formalize the equation as

$$\min \quad tpi_{gpipe} = p_1. \tag{9}$$

Computation-stage constraint With only one computation stage in intra-layer-only parallelism, the communication-stage constraint can be omitted, and the computation and communication cost can be modeled for p_1. Thus, we could formalize the constraint as

$$\sum_{u \in \mathbb{V}} \boldsymbol{S}_u^\mathsf{T} \boldsymbol{A}_u + \sum_{\langle u,v \rangle \in \mathbb{E}} \boldsymbol{S}_u^\mathsf{T} \boldsymbol{R}_{uv} \boldsymbol{S}_v = p_1. \tag{10}$$

In the equation, the first summation term for any $u \in \mathbb{V}$ represents the cost of choosing intra-layer strategies for all layers, while the second term represents the summation of resharding costs on all edges.

Memory constraint Similar to the memory constraint in MIQP, it is necessary to ensure that the memory usage on a single device does not exceed its device memory bound m in QIP. This restriction gives

$$\sum_{u \in V} \boldsymbol{S}_u^\mathsf{T} \boldsymbol{M}_u \leqslant m. \tag{11}$$

It is worth noting that m should be an identical constant across multiple devices if these devices are homogeneous. Otherwise, the value of m varies.

Strategy-selection constraint For intra-layer-only parallelism, the layer-placement constraint can be safely omitted because it is designed for PP. However, the strategy-selection constraint is necessary because each layer can only select one intra-layer strategy. Therefore, the strategy-selection constraint for QIP is identical to Equation (8a) and (8b) for MIQP.

By combining objective function (9) and constraints (8a), (8b), (10), and (11), we have the QIP expression for optimizing the intra-layer-only AP. Like MIQP expression for optimizing the inter- and intra-layer AP, UniAP will eventually get the minimum TPI and corresponding parallel strategies by invoking the off-the-shelf solver.

D. Visualization for the candidate solution

In this section, we proceed to visually represent a potential solution for UOP. Given a deep learning model $\mathcal{G}$, pipeline parallel size pp_size, and number of micro-batches c, UniAP will determine layer placement $\boldsymbol{P}$ for inter-layer parallelism and parallel strategy $\boldsymbol{S}$ for intra-layer parallelism using an off-the-shelf solver. As Figure 7 shows, the solver is optimizing a three-layer model with two pipeline stages, each assigned four GPUs. At this time, a candidate solution could be

$$\boldsymbol{P} = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}, \ \boldsymbol{S} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 0 \\ \vdots & \vdots & \vdots \\ 0 & 0 & 0 \end{bmatrix}. \tag{12}$$

Here, the u-th row of matrix $\boldsymbol{P}$ denotes the placement strategy for layer u, where $\boldsymbol{P}_{ui} = 1$ signifies the placement of layer u on stage i, while 0 indicates otherwise. For example, $\boldsymbol{P}_{l_0} = [1,\ 0]$ denotes the placement of layer l_0 on pipeline stage 1. Additionally, the u-th column of matrix $\boldsymbol{S}$ denotes the selected intra-layer parallel strategy for layer u, where $\boldsymbol{S}_{uj} = 1$ denotes the selection of the j-th strategy from the intra-layer parallel strategy set. For example, $\boldsymbol{S}_{l_0} = [1,\ 0,\ 0,\ \cdots,\ 0]^\mathsf{T}$ indicates that layer l_0 will adopt only the DP strategy, while $\boldsymbol{S}_{l_1} = [0,\ 1,\ 0,\ \cdots,\ 0]^\mathsf{T}$ indicates that layer l_1 will employ a strategy where DP is performed on GPU 0, 1 and GPU 2, 3, and TP is performed across these two GPU groups.

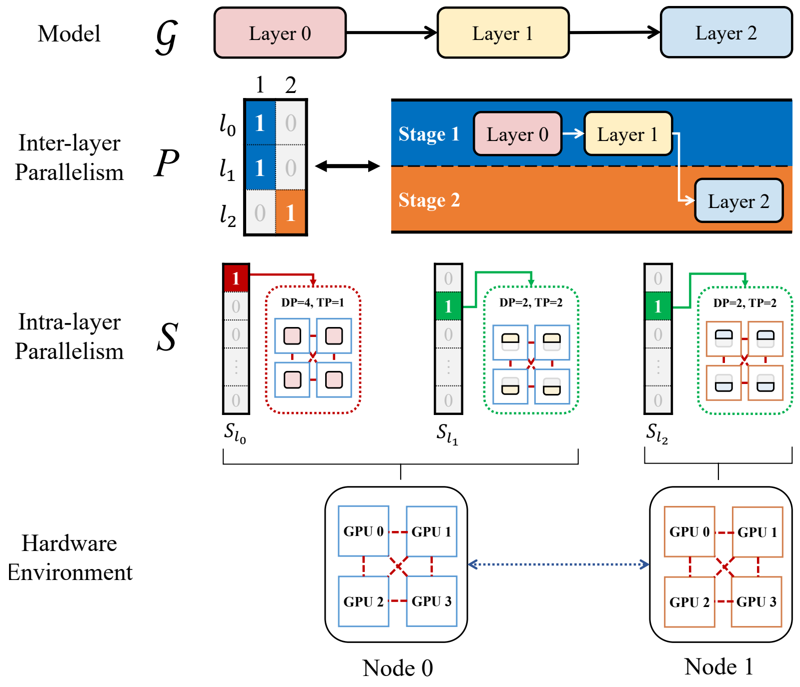

Figure 7. A candidate solution for UOP.

There exist numerous combinations of $\boldsymbol{P}$ and $\boldsymbol{S}$. The off-the-shelf solver will automatically search for the optimal solution given pipeline parallel size pp_size and the number of micro-batches c. By solving the MIQP expression and enumerating every possible pp_size and c in the UOP process, UniAP will ultimately derive an optimal parallel strategy for the deep learning model within the current hardware environment.

E. Experiment detail

Gurobi configuration When tackling the MIQP problem, UniAP employs several configurations for the Gurobi Optimizer 10.1 [13]. In particular, we set *TimeLimit* to 60 seconds, *MIPFocus* to 1, *NumericFocus* to 1, and remain other configurations to default. For instance, we establish the *MIPGap* parameter as the default value of 1e-4 to serve as a strict termination criterion. Furthermore, we have implemented an early stopping mechanism to terminate the optimization process as early as possible. There are two conditions that can activate the mechanism. Firstly, if the current runtime exceeds 15 seconds and the relative MIP optimality gap is less than 4%, we will terminate the optimization. Secondly, if the current runtime exceeds 5 seconds and the best objective bound is worse than the optimal solution obtained in the previous optimization process, we will terminate the optimization.

Model detail Table 6 summarizes six Transformer-based models selected for our evaluations. Four of these models, namely BERT-Huge [5], T5-Large [31], Llama-7B, and Llama-13B [40, 41], belong to the domain of natural language processing (NLP). At the same time, the remaining two, ViT-Huge [6] and Swin-Huge [24], are associated with computer vision (CV). It is noteworthy that BERT, ViT, and Llama maintain consistent types of hidden layers respectively, whereas T5 and Swin have different types of hidden layers. Numbers separated by slashes represent the statistical information for different layer types. For instance, Swin-Huge comprises four types of layers, each with 2, 2, 42, and 2 layers, respectively.

Training detail UniAP is based on the PyTorch framework and integrates models from HuggingFace Transformers. It employs various types of parallelism, including Pipeline Parallelism (PP), Data Parallelism (DP), Tensor Parallelism (TP), and Fully Sharded Data Parallelism (FSDP), utilizing GPipe [15], PyTorch DDP [22], Megatron-LM [30], and FairScale [8], respectively. For NLP models, we use the English Wikipedia dataset [43], while the ImageNet-1K dataset [35] is used for CV models. We train these models using the Adam optimizer [18]. We omit hyperparameters here such as learning rate and weight decay as these have minimal impact on training throughput. The model parameters in the HuggingFace

Model	Task[1]	Statistics				Precision
		#hidden layers	Hidden size	Sequence length	#params	
BERT-Huge	PT	32	1280	512	672M	FP32
T5-Large	CG	24/24	1024	512	737M	FP32
ViT-Huge	IC	32	1280	196	632M	FP32
Swin-Huge	IC	2/2/42/2	320	49×64	1.02B	FP32
Llama-7B	CLM	32	4096	2048	7B	FP16
Llama-13B	CLM	40	5120	2048	13B	FP16

[1] PT: Pretraining; CG: Conditional Generation; IC: Image Classification; CLM: Causal Language Modeling.

Table 6. Summary of the evaluated models.

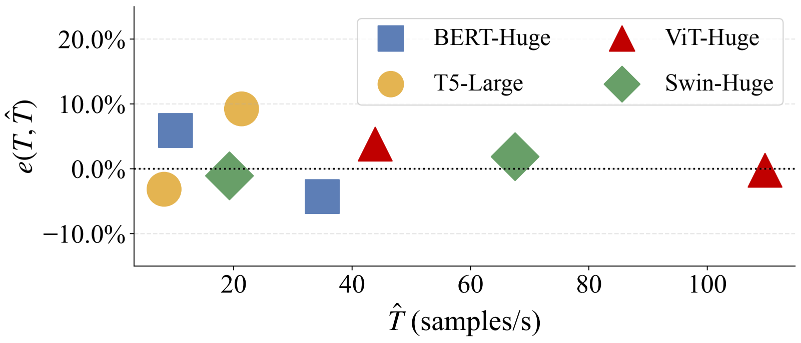

Figure 8. Relative estimation error.

Transformers are configured to align with the specifications of each individual model. For instance, we set *hidden_size* to 1280, *num_hidden_layers* to 32, *num_attention_heads* to 16, and *seq_length* to 512 for BERT-Huge. Regarding other hyperparameters in the HuggingFace configurations, we set *hidden_dropout_prob* and *attention_probs_dropout_prob* to 0.0 for ViT-Huge. For Swin-Huge, we set *drop_path_rate* to 0.2. We remain other configurations to default. It should be noted that the training batch sizes for each experiment are outlined in the main paper.

F. Estimation accuracy

Some variables in UniAP and other AP methods are estimated values rather than actual running values. The TPI (inverse of training throughput) returned by UniAP and other AP methods is one of them. Accurate estimation for TPI or training throughput is crucial for evaluating candidate parallel strategies and ensuring the optimality of the solution. To quantify the accuracy of the estimated training throughput, we introduce a metric called *relative estimation error (REE)* e for training throughput:

$$e(T,\hat{T}) = \frac{|T-\hat{T}|}{T} \times 100\%, \tag{13}$$

where T is the actual training throughput and $\hat{T}$ is the estimated training throughput.

We evaluate the optimal parallel strategies obtained from EnvA and EnvB and visualize the REE of UniAP in Figure 8. The results show that UniAP achieves an average REE of 3.59%, which is relatively small. In contrast, the average REE for Galvatron [25] in our experiments is 11.17%, which is larger than that of UniAP.

G. Case study: BERT-Huge

In this section, we present a visualization of the optimal parallel strategy discovered by UniAP. As represented in Figure 9, the strategy pertains to training BERT-Huge with 32 hidden layers in a 2-node environment *EnvB* with a mini-batch size of 16. Each node was equipped with 2 Xeon E5-2620 v4 CPUs, 4 TITAN Xp 12GB GPUs, and 125GB memory. These nodes are

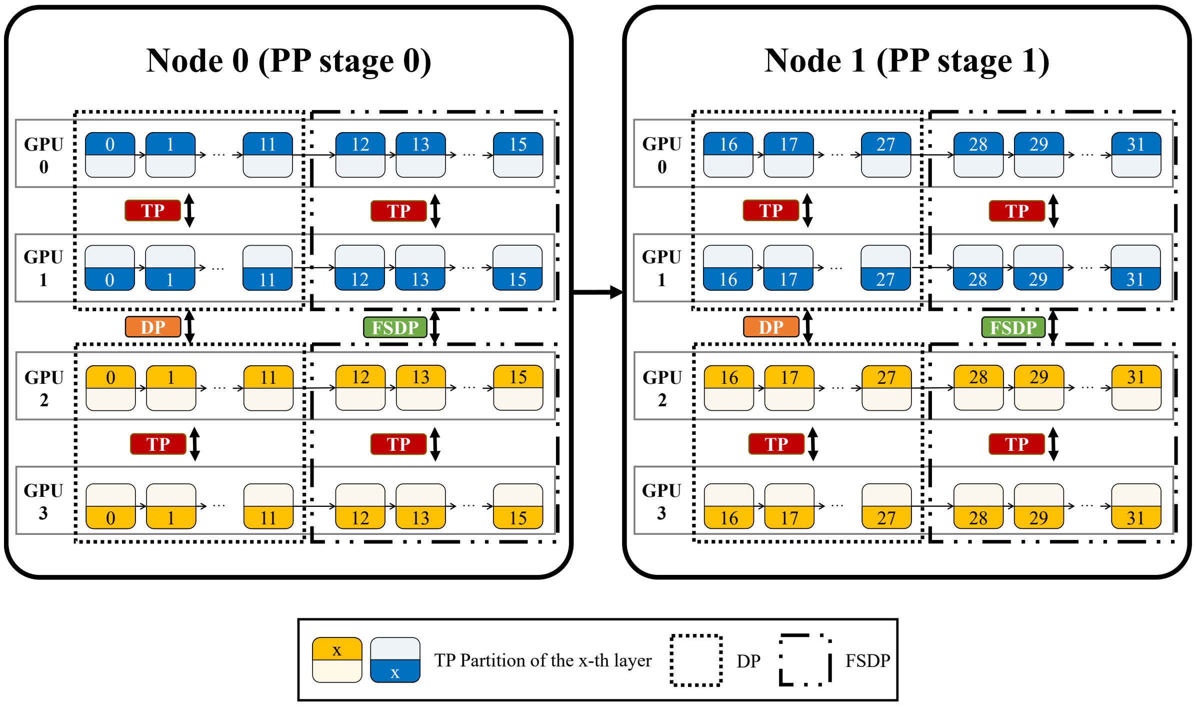

Figure 9. The optimal parallel strategy for all hidden layers of BERT-Huge on *EnvB*. Different colors represent different input samples in a micro-batch.

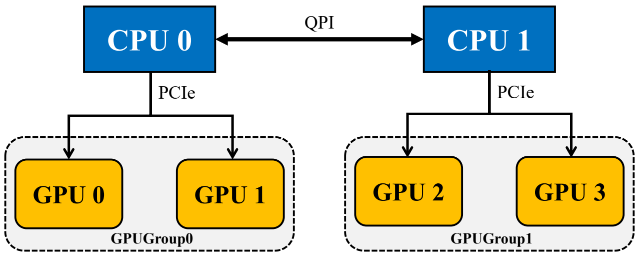

Figure 10. Topology of a node in *EnvB*.

interconnected via a 10Gbps network. It should be noted that we only showcase the parallel strategy for the hidden layers here for simplicity but without losing generality.

Here, we provide further topological information for a node within *EnvB*. As illustrated in Figure 10, we categorize the GPUs numbered 0 and 1 in each node and refer to them collectively as *GPUGroup0*. Similarly, we label the GPUs numbered 2 and 3 as *GPUGroup1*. In *EnvB*, the interconnects within each GPU group (i.e., PCIe) have superior bandwidth than that between different groups (i.e., QPI). We collectively designate these two connection bandwidths as intra-node bandwidth, which is higher than inter-node bandwidth.

In this example, UniAP has identified a parallel strategy for inter-layer parallelism that involves a two-stage pipeline. This strategy utilizes parallelism in a manner that is both efficient and effective. Specifically, the communication cost of point-to-point (P2P) between two nodes is less than that of all-reduce. Given that the inter-node bandwidth is lower than the intra-node bandwidth, the two-stage PP becomes a reasonable choice. Moreover, the pipeline has been designed such that each stage comprises an equal number of layers. This design leverages the homogeneity of the nodes and ensures load balancing across the cluster.

Within each PP stage, UniAP employs an intra-layer parallel strategy. It utilizes a 2-way DP for the initial 12 hidden layers in each stage between *GPUGroup0* and *GPUGroup1*. For the remaining four hidden layers, a 2-way FSDP is utilized between

GPUGroup0 and *GPUGroup1* to reduce memory footprint and meet memory constraints. Within each GPU group, UniAP employs a 2-way TP for each layer. In general, TP incurs more significant communication volumes than DP and FSDP. In order to achieve maximum training throughput on *EnvB*, it is necessary to implement parallel strategies that prioritize higher communication volumes within each group and lower volumes between groups. Therefore, the strategy for BERT-Huge with 32 hidden layers combines the best elements of PP, DP, TP, and FSDP to maximize training throughput.

In addition, we have conducted calculations for the model FLOPs utilizatio (MFU) [4] for Galvatron, Alpa, and UniAP in this scenario to validate our analysis. MFU is independent of hardware, frameworks, or implementations. Therefore, it allows us to examine the performance of different parallel strategies solely from a strategic perspective. For BERT-Huge, the resulting MFUs for UniAP, Galvatron, and Alpa are 58.44%, 58.44%, and 55.10% on EnvA, while 23.6%, 13.7%, and 19.6% on EnvB, respectively. These results validate that UniAP's optimization of inter- and intra-layer AP results in superior performance compared to Galvatron and Alpa.

H. Limitation

UniAP is currently designed and tested on homogeneous clusters, but incorporating automatic parallelism for training deep models on heterogeneous clusters (e.g., a cluster equipped with both NVIDIA GPUs and DCUs) is another important research topic. Given that current parallel techniques primarily target homogeneous clusters with limited emphasis on heterogeneous clusters, we have chosen to leave this topic for future exploration.