

Synthesizing Optimal Parallelism Placement and Reduction Strategies on Hierarchical Systems For Deep Learning

Ningning Xie¹ Tamara Norman² Dominik Grewe² Dimitrios Vytiniotis²

¹University of Cambridge ²DeepMind

Presenter: Shiwei Zhang

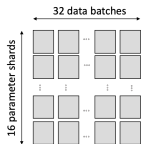
Content

- ▶ Introduction
- ▶ Design Overview
- ▶ Synthesis Algorithm
- ▶ Experiments
- ▶ Summary

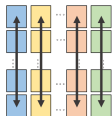
Introduction

Parallelism and Communication

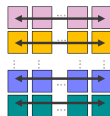
- ▶ Recent studies combine data parallelism and model parallelism (parameter sharding) to maximize training throughput.
- ▶ How we map parallelism over devices decides the communication overhead.
- ▶ Each form of parallelism is referred to as a *parallelism axis*.



(a) Combining
parameter sharding
and data parallelism

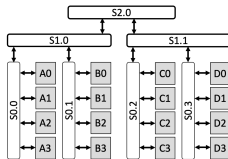


(b) Reduction
along the axis of
parameter sharding

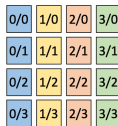


(c) Reduction
along the axis of
data parallelism

Parallelism and Communication



(a) [(rack, 1), (server, 2), (CPU, 2), (GPU, 4)]



(b) $\begin{bmatrix} 1 & 2 & 2 & 1 \\ 1 & 1 & 1 & 4 \end{bmatrix}$



(c) $\begin{bmatrix} 1 & 2 & 1 & 2 \\ 1 & 1 & 2 & 2 \end{bmatrix}$



(d) $\begin{bmatrix} 1 & 1 & 2 & 2 \\ 1 & 2 & 1 & 2 \end{bmatrix}$

Figure 2: (a): A system. (b), (c), (d): Possible (non-exhaustive) parallelism placements for (a) under data parallelism of size 4 and 4 parameter shards. For clarity, we show only the 16 GPUs but omit interconnects. Device marker n/m indicates data batch n and parameter shard m .

P^2 : a tool for parallelism placement and placement-aware synthesis of reduction strategies

- ▶ Parallelism placement synthesis: mapping parallelism axes to the system hierarchy.
- ▶ Reduction strategy synthesis: synthesize a wide variety of reduction strategies to implement reductions using common collective operations.

Content
○

Introduction
○○○○

Design Overview
●○○○○

Synthesis Algorithm
○○○○○○

Experiments
○○○○○○○○

Summary
○○○

Design Overview

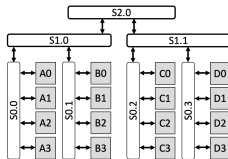
Parallelism Placement

Objective: Deciding which parts of a partitioned program will execute on which parts of a system.

Challenge: Synthesizing all arbitrary device mappings can be extremely expensive.

Solution: Partition parallelism axes over the system hierarchy to generate topology-aware parallelism placements.

Parallelism Matrix



(a) [(rack, 1), (server, 2), (CPU, 2), (GPU, 4)]

0/0	1/0	2/0	3/0
0/1	1/1	2/1	3/1
0/2	1/2	2/2	3/2
0/3	1/3	2/3	3/3

$$(b) \begin{bmatrix} 1 & 2 & 2 & 1 \\ 1 & 1 & 1 & 4 \end{bmatrix}$$

0/0	0/2	2/0	2/2
0/1	0/3	2/1	2/3
1/0	1/2	3/0	3/2
1/1	1/3	3/1	3/3

$$(c) \begin{bmatrix} 1 & 2 & 1 & 2 \\ 1 & 1 & 2 & 2 \end{bmatrix}$$

0/0	2/0	0/2	2/2
0/1	2/1	0/3	2/3
1/0	3/0	1/2	3/2
1/1	3/1	1/3	3/3

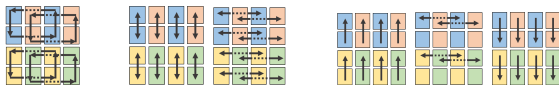
$$(d) \begin{bmatrix} 1 & 1 & 2 & 2 \\ 1 & 2 & 1 & 2 \end{bmatrix}$$

Figure 2: (a): A system. (b), (c), (d): Possible (non-exhaustive) parallelism placements for (a) under data parallelism of size 4 and 4 parameter shards. For clarity, we show only the 16 GPUs but omit interconnects. Device marker n/m indicates data batch n and parameter shard m .

Reduction Strategy

P^2 synthesizes topology-aware reduction strategies using common collective operations.

- ▶ (a) is commonly used but it does not utilize the topology of the system.
- ▶ (b) and (c) are strategies synthesized by P^2 . Their first steps are within S0.
- ▶ (c) has fewer data to transfer over S1/S2 than (b), but it has more steps.



(a) AllReduce (b) AllReduce-AllReduce (c) Reduce-AllReduce-Broadcast

Formalism of Collective Operations

Synthesizing all sequences of collective operations is not necessary. Some sequences of the operations lead to *semantically invalid states* that can never reach the final desired state.

P^2 formalize common collective operations using Hoare triples. A Hoare triple $\{\mathcal{G}_1\}\mathcal{C}\{\mathcal{G}_2\}$ means when the precondition $\{\mathcal{G}_1\}$ is met, executing the command \mathcal{C} establishes the postcondition $\{\mathcal{G}_2\}$.

Content
○

Introduction
○○○○

Design Overview
○○○○○

Synthesis Algorithm
●○○○○○

Experiments
○○○○○○○○

Summary
○○○

Synthesis Algorithm

Parallelism Placement

The Parallelism placement is defined by the parallelism matrix.

$\mathbf{H} = [h_0 \cdots h_n]$ is the system hierarchy (e.g., $[1 \ 2 \ 2 \ 4]$),
 $\mathbf{P} = [p_0 \cdots p_m]$ is the parallelism axes (e.g., $[4 \ 4]$),
then a parallelism matrix is

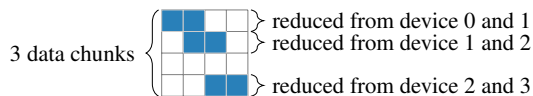
$$\begin{bmatrix} x_{0,0} & x_{0,1} & \cdots & x_{0,n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{m,0} & x_{m,1} & \cdots & x_{m,n} \end{bmatrix} \text{ subject to:}$$
$$\prod_{i=0}^m x_{i,j} = h_j, \quad j = 0, \dots, n \quad (1)$$
$$\prod_{j=0}^n x_{i,j} = p_i, \quad i = 0, \dots, m \quad (2)$$

Collective Operations Notations and States

Notations We first define the notations.

d		device
s	$\in \mathbb{B}^{k \times k}$	device state
\mathcal{G}	$:= \overline{d_i : s_i}$	state context
\mathcal{C}	$:=$ AllReduce ReduceScatter AllGather Reduce Broadcast	

The state of a device is a $k \times k$ boolean matrix where $s[i][j] = 1$ means that device j has contributed its original i th chunk to the reduction result.



Collective Operations Semantics

$\boxed{\{ \mathcal{G}_1 \} \mathcal{C} \{ \mathcal{G}_2 \}}$ (Reduction: from the pre-condition state \mathcal{G}_1 , \mathcal{C} yields to the post-condition state \mathcal{G}_2) before after

R-ALLREDUCE

$$\frac{\forall i j, s_i.\text{rows} = s_j.\text{rows} \quad \forall i j k, i \neq j \implies s_i[k] \otimes s_j[k] \quad s = \uplus \overline{s_i}}{\{ \overline{d_i : s_i} \} \text{AllReduce} \{ \overline{d_i : s} \}}$$



R-REDUCESCATTER

$$\frac{\forall i j, s_i.\text{rows} = s_j.\text{rows} \quad \forall i j k, i \neq j \implies s_i[k] \otimes s_j[k] \quad s = \uplus \overline{s_i} \quad s'_i = \text{scatter}(s, \bar{i})[i]}{\{ \overline{d_i : s_i} \} \text{ReduceScatter} \{ \overline{d_i : s'_i} \}}$$



R-ALLGATHER

$$\frac{\forall i j, i \neq j \implies s_i.\text{rows} \otimes s_j.\text{rows} \quad \forall i j, |s_i.\text{rows}| = |s_j.\text{rows}| \quad s = \uplus \overline{s_i}}{\{ \overline{d_i : s_i} \} \text{AllGather} \{ \overline{d_i : s} \}}$$



R-REDUCE

$$\frac{\forall i j, s_i.\text{rows} = s_j.\text{rows} \quad \forall i j k, i \neq j \implies s_i[k] \otimes s_j[k] \quad s = \uplus \overline{s_i}}{\{ \overline{d_i : s_i} \} \text{Reduce} \{ \overline{d_0 : s, d_i : \{ \}^{i \neq 0}} \}}$$



\otimes disjoint rows non-empty rows
 \uplus addition $|\cdot|$ length
 $\text{scatter}(s, \bar{i})$ scatters non-empty rows in s over devices \bar{i}

R-BROADCAST

$$\frac{\forall i, s_i \leq s_0 \quad \exists i, s_i < s_0}{\{ \overline{d_i : s_i} \} \text{Broadcast} \{ \overline{d_i : s_0} \}}$$

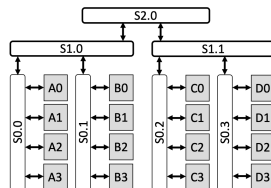


Reduction Program

A reduction strategy is represented as a *program*, a list of reduction instructions.

$$\begin{aligned}
 \text{program} &\in [\text{reduction}] \\
 \text{reduction} &\in \text{slice} \times \text{form} \times \mathcal{C} \\
 \text{slice} &:= e \\
 \text{form} &:= \text{InsideGroup} \mid \text{Parallel}(e) \mid \text{Master}(e)
 \end{aligned}$$

<i>slice</i>	<i>form</i>	<i>groups(slice, form)</i>
CPU	InsideGroup	$\{A_0, A_1, A_2, A_3\}, \{B_0, B_1, B_2, B_3\},$ $\{C_0, C_1, C_2, C_3\}, \{D_0, D_1, D_2, D_3\}$
	Parallel(server)	$\{A_0, B_0\}, \{A_1, B_1\}, \{A_2, B_2\}, \{A_3, B_3\}$ $\{C_0, D_0\}, \{C_1, D_1\}, \{C_2, D_2\}, \{C_3, D_3\}$
	Parallel(rack)	$\{A_0, B_0, C_0, D_0\}, \{A_1, B_1, C_1, D_1\},$ $\{A_2, B_2, C_2, D_2\}, \{A_3, B_3, C_3, D_3\}$
	Master(rack)	$\{A_0, B_0, C_0, D_0\}$
server	InsideGroup	$\{A_0, A_1, A_2, A_3, B_0, B_1, B_2, B_3\},$ $\{C_0, C_1, C_2, C_3, D_0, D_1, D_2, D_3\}$
	Parallel(rack)	$\{A_0, C_0\}, \{A_1, C_1\}, \{A_2, C_2\}, \{A_3, C_3\}$ $\{B_0, D_0\}, \{B_1, D_1\}, \{B_2, D_2\}, \{B_3, D_3\}$
rack	InsideGroup	$\{A_0, A_1, A_2, A_3, B_0, B_1, B_2, B_3,$ $C_0, C_1, C_2, C_3, D_0, D_1, D_2, D_3\}$



Program Synthesis for Reduction Programs

The goal is to find a program \mathcal{L} that

$$\left\{ \overline{d_i : \begin{bmatrix} & i & & & \\ 0 & \dots & 1 & \dots & 0 \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & \dots & 1 & \dots & 0 \end{bmatrix}} \right\} \mathcal{L} \left\{ \overline{d_i : \begin{bmatrix} & i & & & \bar{j} & & \\ 0 & \dots & 1 & \dots & 0 & \dots & 1 & \dots & 0 \\ \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & \dots & 1 & \dots & 0 & \dots & 1 & \dots & 0 \end{bmatrix}} \right\}$$

supposing d_i reduces with devices \bar{j} .

P^2 uses a method called *syntax-guided program synthesis* for this purpose.

Content
○

Introduction
○○○○

Design Overview
○○○○○

Synthesis Algorithm
○○○○○○

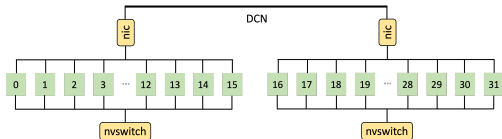
Experiments
●○○○○○○○

Summary
○○○

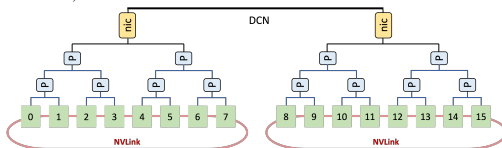
Experiments

Experimental Setup

- ▶ 2 and 4 nodes on Google Cloud Platform.
- ▶ 2 system topologies.



(a) 2 nodes, each with 16 A100 GPUs sharing one NVSwitch and one NIC, and all NICs are connected in a data center



(b) 2 nodes, each with 8 V100 GPUs forming a ring via NVLink and connected via PCIe switches. Each node consists of two CPUs (each owning 4 GPUs) with one NIC to the DCN. A shared NIC connecting the two CPUs is a modeling simplification – in reality cross-domain communication is through shared memory.

Result 1

The performance of AllReduce differs significantly among parallelism matrices, up to $448.5\times$.

Parallelism axes		Parallelism matrix		Reduction on the 0th axis		Reduction on the 1st axis	
				Ring	Tree	Ring	Tree
4 nodes, each with 16 A100							
A1	[2 32]	[1 2]	[4 8]	0.12	0.17	8.74	9.89
A2		[2 1]	[2 16]	37.16	36.94	4.81	3.41
B1	[4 16]	[1 4]	[4 4]	0.15	0.20	17.70	19.03
B2		[2 2]	[2 8]	28.77	19.81	8.39	4.99
B3		[4 1]	[1 16]	56.13	89.70	0.18	0.22
C1	[8 8]	[1 8]	[4 2]	0.17	0.21	33.92	41.06
C2		[2 4]	[2 4]	16.52	9.18	15.68	9.43
C3		[4 2]	[1 8]	34.05	41.23	0.17	0.21
4 nodes, each with 8 V100							
E1	[8 4]	[1 8]	[4 1]	0.28	0.39	21.74	30.42
E2		[2 4]	[2 2]	14.25	15.48	10.98	7.34
E3		[4 2]	[1 4]	14.84	19.90	2.96	0.43

Result 2

The pruning techniques are effective for the synthesizer to achieve fast synthesis time.

In the experiments, the program size limit is set to 5 for the synthesizer, which turns out to be sufficient to generate interesting reduction patterns. With this setup, the longest synthesis time is under 2 seconds (for up to 235 programs). Increasing the size limit makes the synthesis slightly slower, but, for most cases, does not generate new programs.

Result 3

If the reduction axes can be put within one node, then a single step AllReduce inside that node is the most performant reduction due to fast local bandwidth.

Result 4

Synthesized programs can mitigate the impact of parallelism placement.

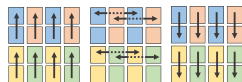
NCCL algo	Parallelism axes	Synthesis time (s)	Programs outperforming AllReduce / total programs	Parallelism matrix	AllReduce (bold if the optimal AllReduce)	Optimal (bold if overall optimal)	Speedup	
2 nodes, each with 16 A100								
F1	Ring	[8 4]	0.03	14/47	$\begin{bmatrix} 1 & 8 \\ 2 & 2 \end{bmatrix}$	0.17	0.17	1×
F2					$\begin{bmatrix} 2 & 4 \\ 1 & 4 \end{bmatrix}$	16.84	9.19	1.83×
4 nodes, each with 16 A100								
G1	Tree	[4 16]	0.04	10/53	$\begin{bmatrix} 1 & 4 \\ 4 & 4 \end{bmatrix}$	0.20	0.17	1.17×
G2					$\begin{bmatrix} 4 & 1 \\ 1 & 16 \end{bmatrix}$	89.70	56.13	1.60×
H1	Ring	[16 2 2]	0.97	25/235	$\begin{bmatrix} 1 & 16 \\ 2 & 1 \\ 2 & 1 \end{bmatrix}$	4.79	4.63	1.03×
H2					$\begin{bmatrix} 2 & 8 \\ 2 & 1 \\ 1 & 2 \end{bmatrix}$	4.91	3.10	1.58×
I1	Ring	[2 2 16]	0.93	29/235	$\begin{bmatrix} 2 & 1 \\ 2 & 1 \\ 1 & 16 \end{bmatrix}$	4.82	2.99	1.61×
I2					$\begin{bmatrix} 1 & 2 \\ 2 & 1 \\ 2 & 8 \end{bmatrix}$	5.28	4.77	1.11×
J1	Tree	[64]	1.16	5/47	$\begin{bmatrix} 4 & 16 \end{bmatrix}$	5.75	4.74	1.21×
4 nodes, each with 8 V100								
K1	Ring	[8 2 2]	0.24	17/188	$\begin{bmatrix} 2 & 4 \\ 2 & 1 \\ 1 & 2 \end{bmatrix}$	4.80	2.35	2.04×
K2					$\begin{bmatrix} 1 & 8 \\ 2 & 1 \\ 2 & 1 \end{bmatrix}$	4.40	4.40	1×
L1	Ring	[32]	0.06	11/47	$\begin{bmatrix} 4 & 8 \end{bmatrix}$	4.83	3.45	1.4×

Result 5

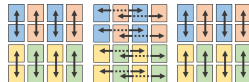
For reduction across nodes, a topology-aware reduction program tends to outperform a single step AllReduce, with speedup on average $1.28\times$, upto $2.04\times$.

Optimal strategies found by P^2

For ResNet-50 model, P^2 found the optimal strategy (ii) that achieves 15% overall training time speedup compared to the baseline (Haiku).



(i) Reduce-AllReduce-Broadcast



(ii) ReduceScatter-AllReduce-AllGather

Content
○

Introduction
○○○○

Design Overview
○○○○○

Synthesis Algorithm
○○○○○○

Experiments
○○○○○○○○

Summary
●○○

Summary

Conclusion

Strength

- ▶ Jointly optimize the parallelism placement and reduction strategy for hierarchical topologies.
- ▶ Formalize the collective semantics to automatically search for valid programs.

Limitation

- ▶ Only strictly symmetric and hierarchical topologies are considered.
- ▶ The optimal reduction strategy is simple and has already been studied.
- ▶ Why not take a step further and also consider the parallelism strategy?

Takeaways

- ▶ Operation synthesis
 - ▶ Communication synthesis: transform a single collective operation into multiple smaller operations. (P^2 , BlueConnect, SCCL, etc.)
 - ▶ Computation synthesis: transform a computation operation into multiple smaller operations. (TASO, DietCode, etc.)
 - ▶ Parallelism strategy synthesis: transform a computation operation into a series of communication and computation operations.
- ▶ Define the state of the system and treat operations as directed links (with costs) that connect states.

Thank you!