SymTensor: Symbolic and Adaptive Tensor Partitioning by Unified Parallelism for Deep Learning

Hongxing Wang, Zhengdao Yu, Chong Li, Serge Petiton

HLPP 2025 @ Innsbruck

Background: Rapid Evolution of Large Language Models (LLM)

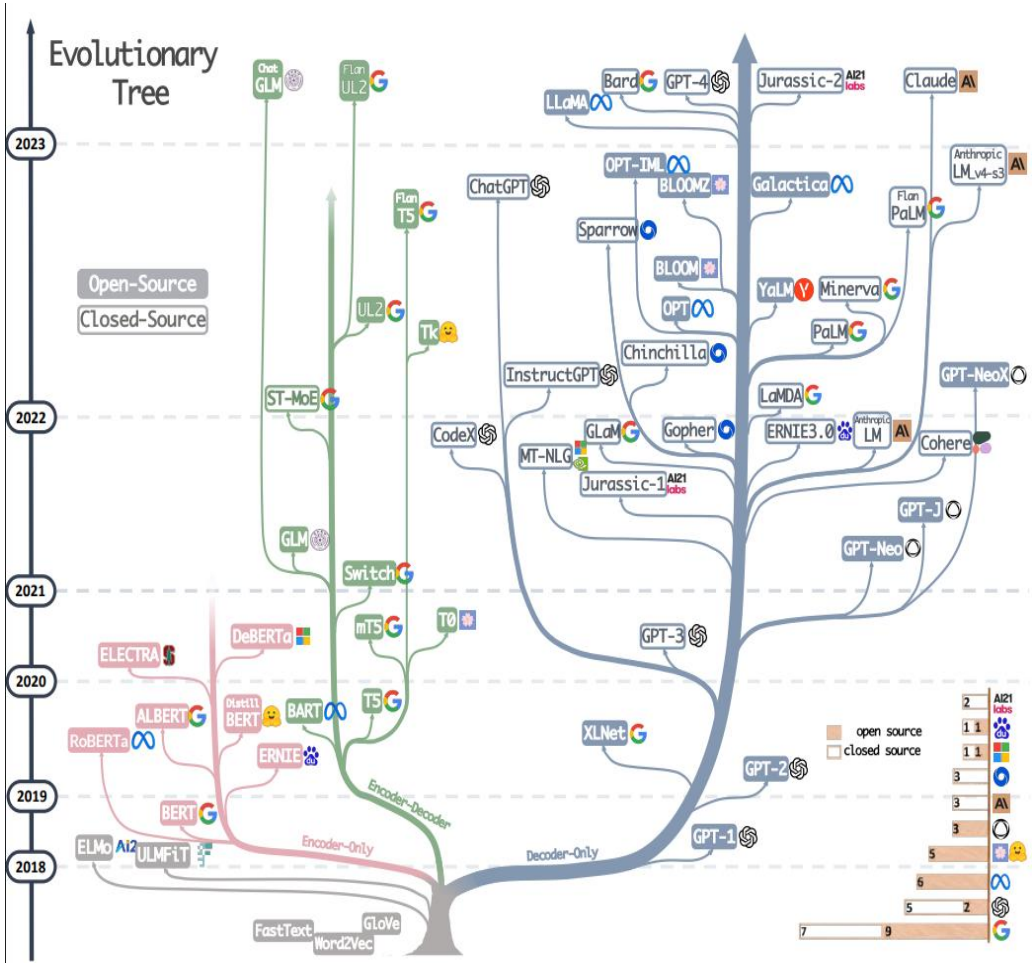


Image source:Harnessing the Power of LLMs in Practice: A Survey on ChatGPT and Beyond,
J JIANG et al., AMAZON USA, 2023

LLM evolution remains extraordinarily rapid:

- **Expansion of parameters**:
  - From millions to hundreds of billions

- **Structural diversity**:
  - BERT -> Encoder-Decoder architecture
  - GPT -> Decoder-only architecture
  - Mixtral -> MoE architecture
  - DeepSeek -> Dense + MoE architecture

  and more LLM...

- **Operator innovations**:
  - Self-Attention -> FlashAttention
  - Operator Fusion: Mul + Add -> Mul_Add ...

LLM designers want a software framework that could:

1. help them to focus on DL design
2. simplifying performance tuning (on distributed clusters)

Deep Learning (DL) Frameworks: Designed for User-Friendly

1. Simplifying for Training

DL frameworks:

PyTorch, MindSpore, TensorFlow

DL designer

can focus on DL network without worrying low-level details

DL Framework

Input: DL Network

DL training:

Forward Pass → Loss → Backward Pass → Update → Forward Pass

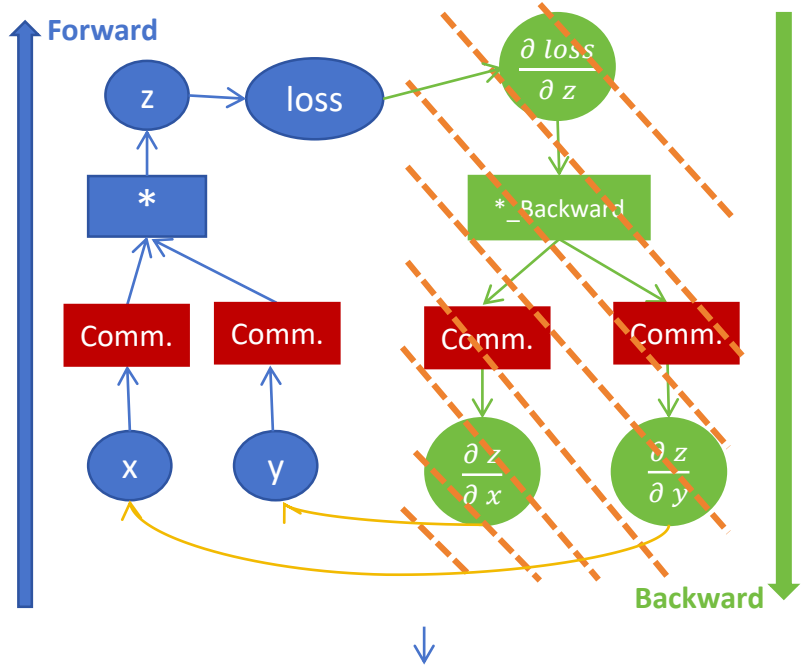
Backward auto generation for reducing network development complexity

2. Simplify for Parallelization

DL Framework	Parallelization Planner
Pytorch	**Megatron-LM**
Jax	**Alpa**
MindSpore	**SAPP**

Planner: decide & manage Comm. Operator

designer's perspective:

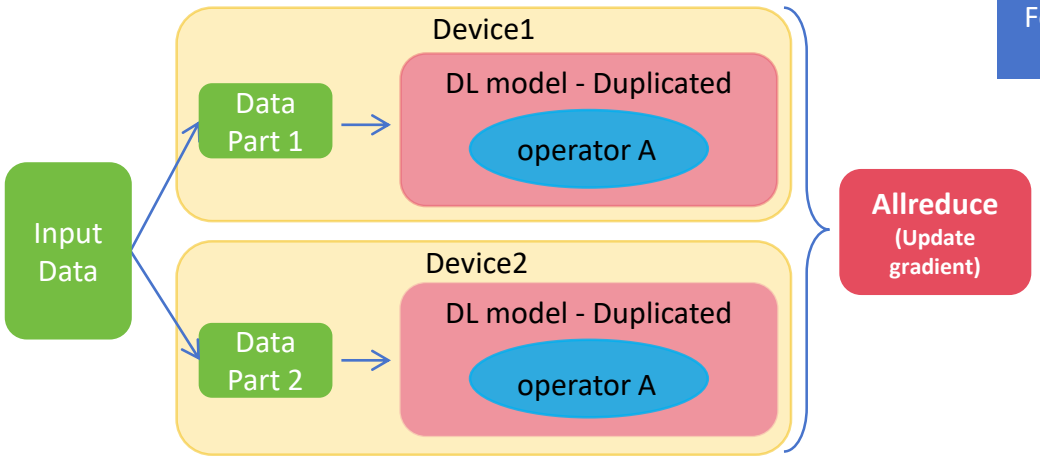


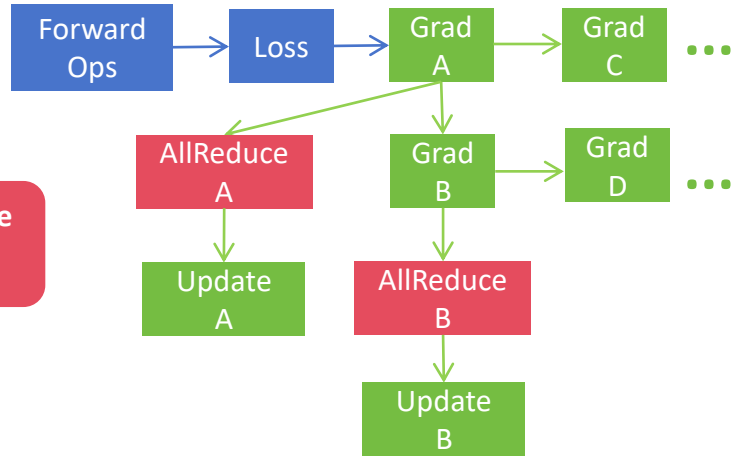
Indirect control necessitates systematic approaches

Parallelisms in DL: Predefined Patterns with Performance Guarantee

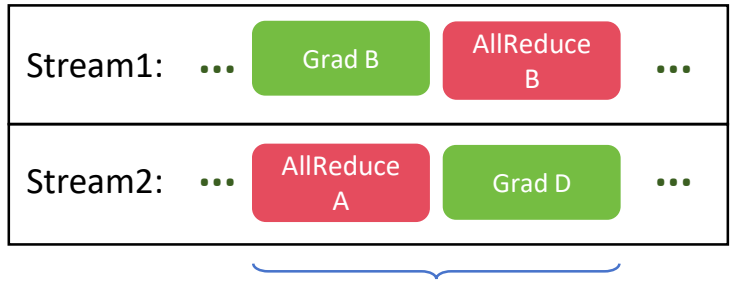
- **Data Parallelism (DP):**

Replicate model on each worker, split the input data





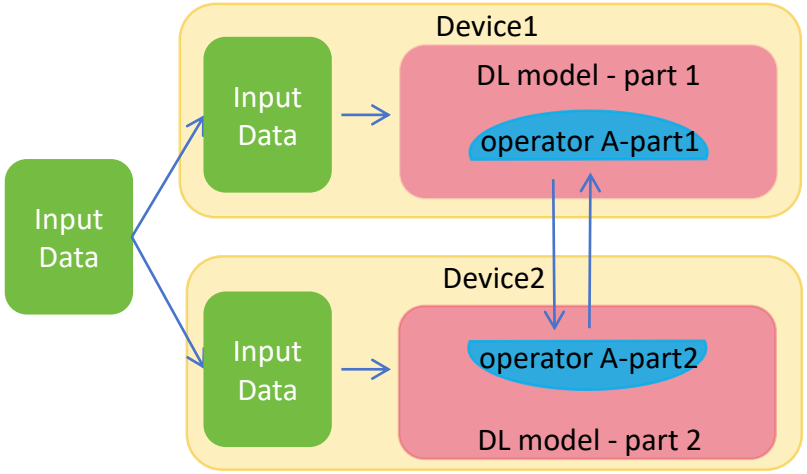
Hide comm. cost:



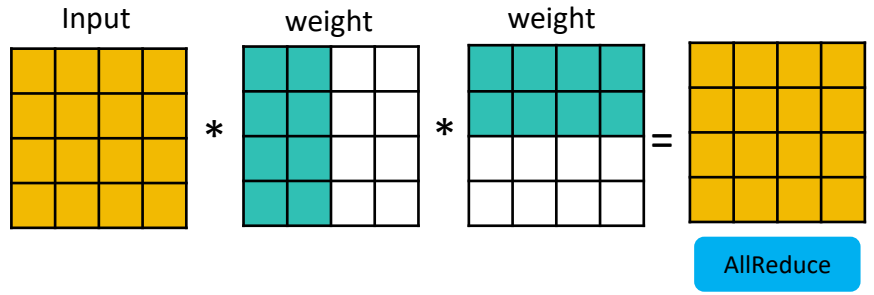
Communication Overlaped
Better Performance

- **Tensor Parallelism (TP):**

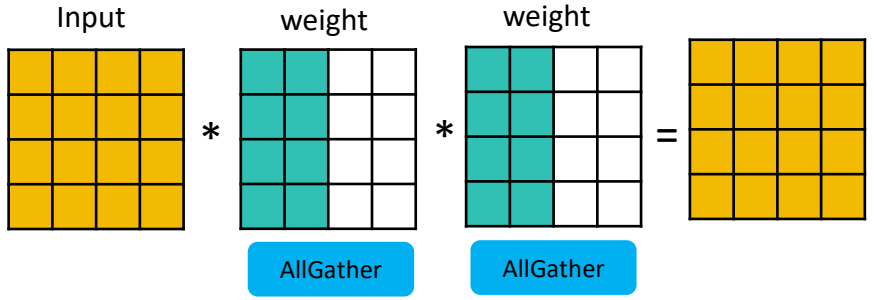
Split weight of operators (MatMul) across workers



Skip bad choices, e.g. for X * Y * Z:



Only 1 single
Allreduce is needed



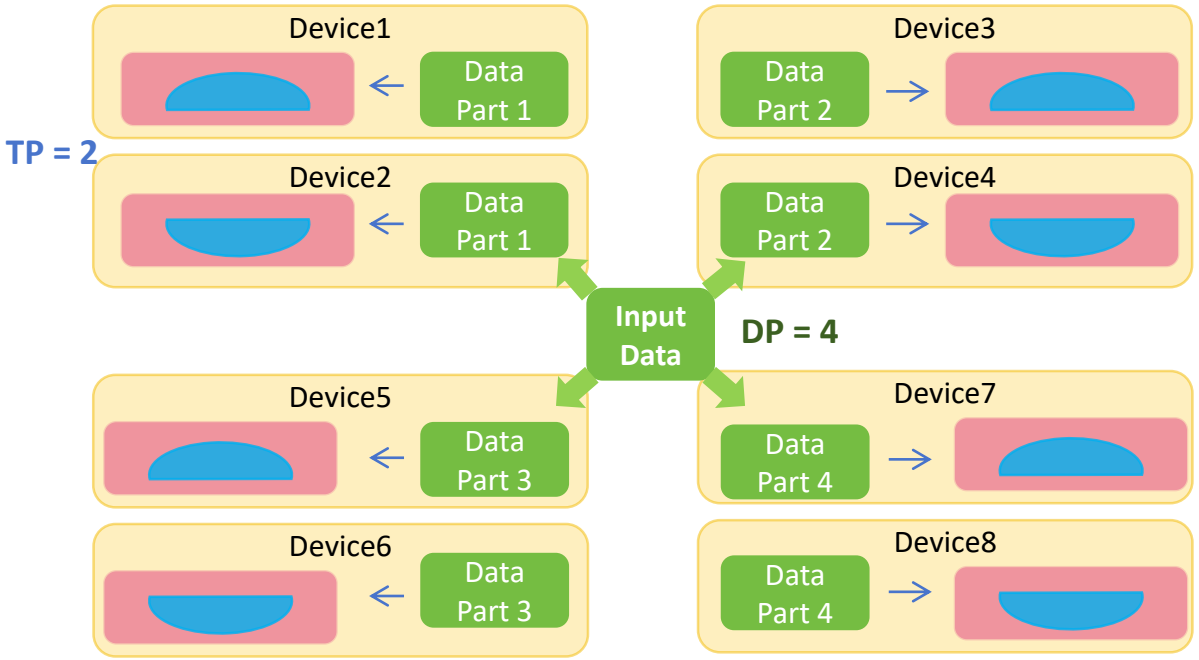
2 AllGathers are needed

....

Strategy of Parallelisms: Finding Best Performance for End-User

	Definition	Example
Strategy	Configure degrees of parallelism over the AI model	(DP = 4, TP = 2) over 8 devices

(DP = 4, TP = 2) over 8 devices:



Experiment on Deepseek V3 671B (64 devices)

MFU - Model FLOPs Utilization

Config	DP	TP	MFU	Insights
Option 1	8	8	26.84%	Balanced DP/TP
Option 2	64	1	Out-of-Memory	DP-only parallelism → OOM
Option 3	32	2	31.42%	High-DP within memory limits -> Better performance

Parallelism Strategy is important:

- Same 64 devices, different DP/TP configs -> vastly different outcomes
- From OOM to 26.84% to 31.42% MFU -> strategy choice makes the difference

Challenges

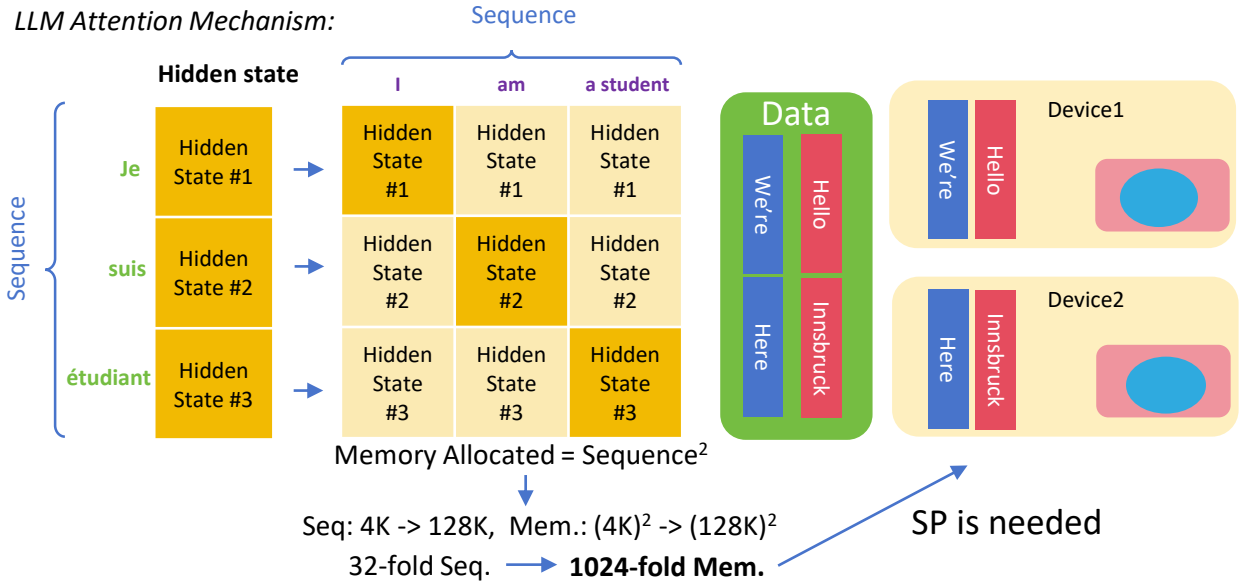
Challenge 1: Existing frameworks lack adaptability for evolving models & parallelisms

Relying on predefined rules and paradigms: DP, TP, ...

Problem:

New parallelisms continue to emerge to meet evolving training demands

- Sequence Parallelism (SP) for long sequence scenario (4K tokens -> 128K tokens)
- Expert Parallelism (EP) for MoE architecture
- ...



Lack of unified abstraction of parallelisms

Challenge 2: Difficult to determine best parallelism strategy

Parallelisms were designed independently

However parallelism strategies require combining them

Problem:

Parallelism combinations yield undetermined performance impacts

Experiment on Deepseek V3 671B (64 devices)

Config	DP	TP	MFU	Insights
Option 1	8	8	26.84%	Balanced DP/TP
Option 2	64	1	Out-of-Memory	DP-only parallelism → OOM
Option 3	32	2	31.42%	High-DP within memory limits -> Better performance

Lack of comprehensive cost model

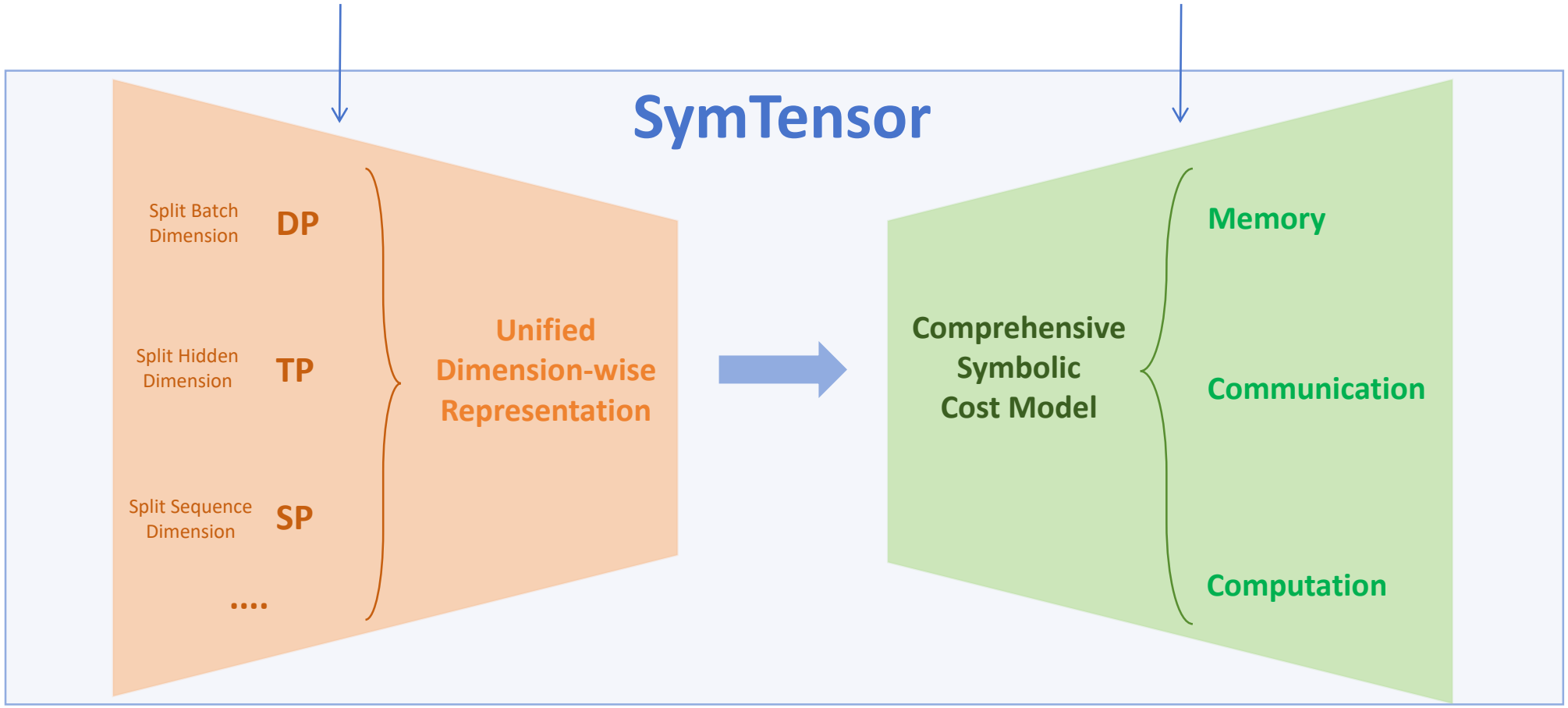
Our Solution

For challenge 1

Lack of unified abstraction of parallelisms

For challenge 2

Lack of comprehensive cost model

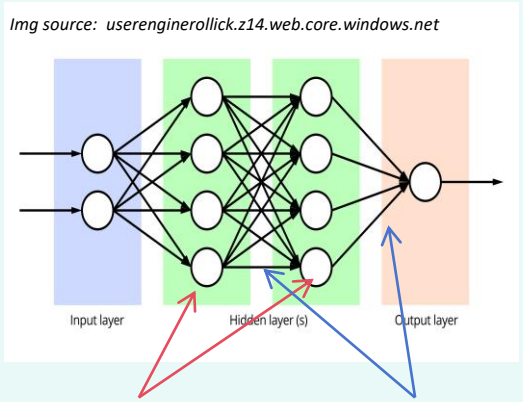


Solution in Detail

Input

DL model is described as a computational graph: G = (V, E)

- each node v ∈ V -> **Operator**
- each edge e ∈ E -> **Tensor Flow**



Unified Dimension-wise Representation

We define:

$shape(\text{Tensor}) = (d_1, d_2, \ldots, d_k)$

d_i = size of dimension i

Map parallelism to **Tensor-level**, dimension-wise partitioning abstraction

$strategy(\text{Tensor}) = (s_1, s_2, \ldots, s_k)$

s_i = number of partitions along d_i

Symbolic Cost Model

- Communication Cost of redistribution between tensor T

1.AllReduce (TP - Part. Result Sum)
2.AllGather (DP - Gradient Aggregation)
3.ReduceScatter (TP - Part. Result distribution)
4.AllToAll (EP - Expert Tokens Routing)

$$Cost_{comm}(T) = \alpha + DataVolume * \beta$$

Hockney Model:
α : Comm. Latency
β : data transfer time

1.Ring-based AllReduce can be modeled as:

$$(P-1)\alpha + \frac{2(P-1)}{P}\beta * \text{DataVolume(T)}$$

2.Ring-based AllGather can be modeled as:

$$(P-1)\alpha + \frac{(P-1)}{P}\beta * \text{DataVolume(T)}$$

P : Number of devices

- Memory Cost of tensor T

$$Costmem(T) = \prod_{i=1}^{k} \frac{d_i}{s_i}, \text{ where } \prod_{i=1}^{k} s_i = P$$

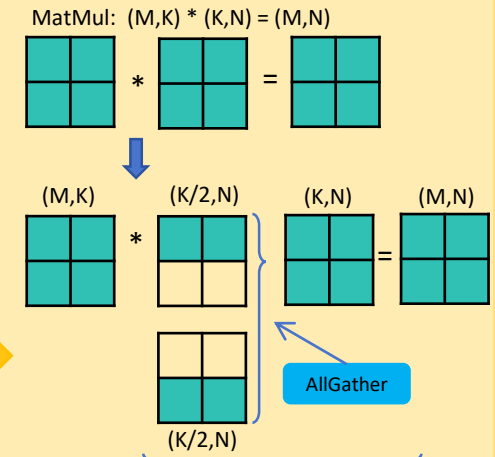
Operator Matmul: (M,K) * (K,N) = (M, N)

- s_M=**X**, s_K=1, s_N=1: $Cost_{mem}$(Matmul) = ($\frac{M}{X}$*K) + (K*N) + ($\frac{M}{X}$*N)
- s_M=1, s_K=**X**, s_N=1: $Cost_{mem}$(Matmul) = (M* $\frac{K}{X}$) + ($\frac{K}{X}$*N) + (M*N)
- s_M=1, s_K=1, s_N=**X** : $Cost_{mem}$(Matmul) = (M*K) + (K* $\frac{N}{X}$) + (M* $\frac{N}{X}$)

- Computation Cost

Partitioning does not affect the total FLOPs of an operator -> we thus exclude computation cost to simplify the cost model

Redistribution Comm. :



Total Cost of operator v

$$Cost_{total}(v) = \sum_{\text{tensors } T \text{ of } v} \left(\frac{Cost_{mem}(T)}{\gamma} \oplus Costcom_m(T)\right)$$

γ = Memory Copy Bandwidth

SymTensor workflow:

8 Devices →

Input:

OpV(Tensor1, Tensor2) = Tensor3
shape(Tensor1) = (1, 4096, 6144)
shape(Tensor2) = (1, 12288,32000)
shape(Tensor3) = (1, 4096, 32000)

Initial Strategy:

strategy(Tensor) = (1,1,1)

→ Recursive Binary Search →

$Cost_{mem}$(Tensor1)= 12,582,912 Byte (FP32)
$Cost_{mem}$(Tensor2)=196,608,000 Byte (FP32)
$Cost_{mem}$(Tensor3)=65,536,000 Byte (FP32)
$Cost_{comm}$(Tensor1)=10.0 ms
$Cost_{comm}$(Tensor2)=0.0 ms

$$Cost_{total}(OpV) = \sum_{T \text{ of } v} \left(\frac{274{,}726{,}912Byte}{10GB/s} + 10.0ms\right)$$

= 35.5 ms (lowest cost obtained)

Output: *strategy*(Tensor) = (1,2,4)

Experiment Environment

MindSpore

Open-Source DL framework:
https://gitee.com/mindspore/mindspore

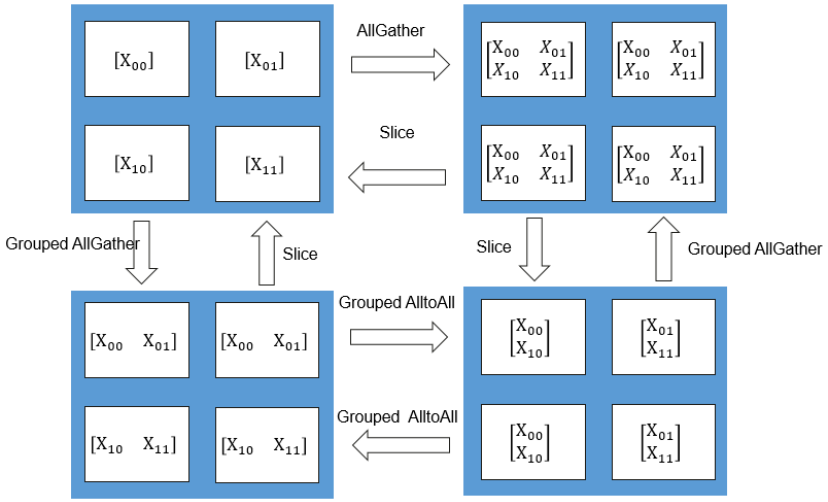
```
class Layer(nn.Cell):
    def __init__(self):
        super(Layer, self).__init__()
        self.matmul1 = ops.MatMul.shard(((2, 1), (1, 2)))
            self.relu = ops.Relu()
        self.matmul2 = ops.MatMul.shard(((2, 2), (2, 1)))
    def construct(self, x, w, v):
        y = self.matmul1(x, w)
            y = self.relu(y, w)
        z = self.matmul2(y, v)
        return s
```

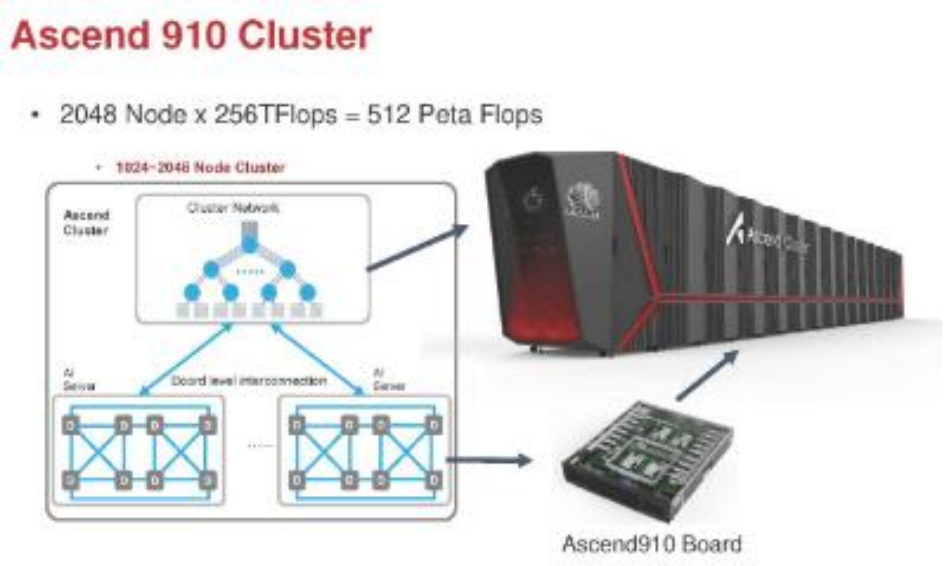
Parallel Strategy

shard(([d0, d1, d2], [d2, d1, d0]))

Data Parallelism
shard(([dev_num, 1, 1], [1, 1, 1]))

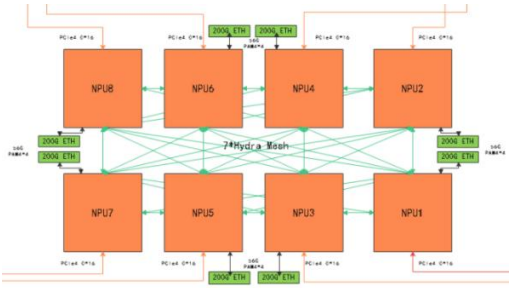
Model Parallelism
shard(([1, 1, 1], [1, 1, dev_num]))





Scalability with Hierarchical Comm has been discussed in:
H. Wang, C. Li, T. Tachon, et al., "Efficient and systematic partitioning of large and deep neural networks for parallelization," in Euro-Par 2021

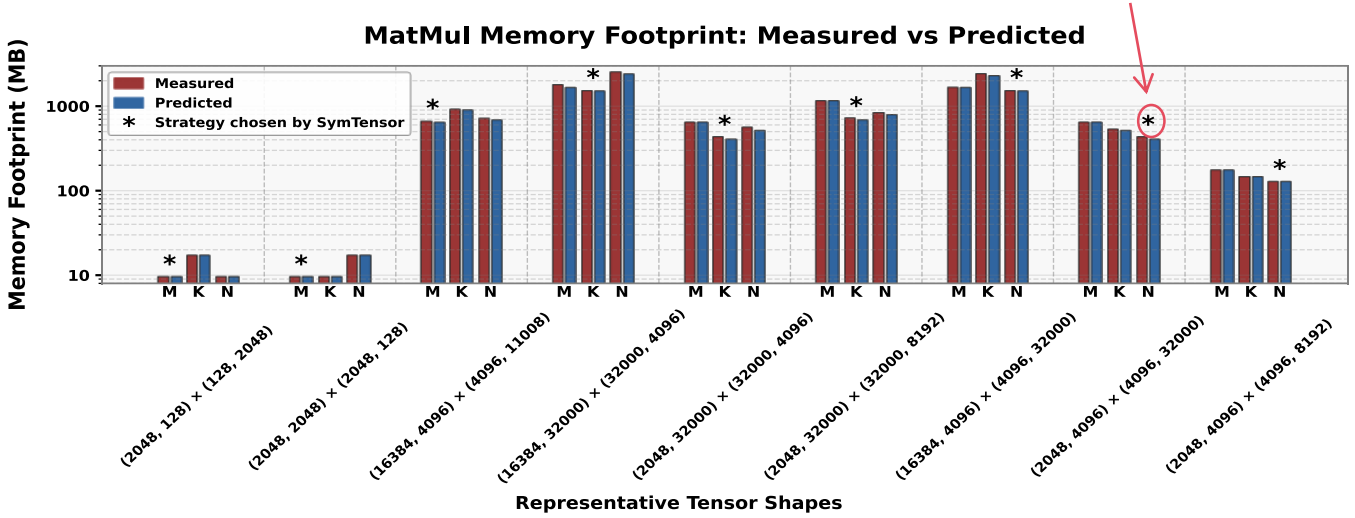
This paper focuses on Comm- & Mem-aware Exp str on a 910 A2 node



8x Ascend NPU per node
Each NPU:
- To CPU/Mem: PCIe 4.0 x16
- To neighbors:
  - 7x 56 GB/s full mesh HCCS
  - 1x 200G Eth

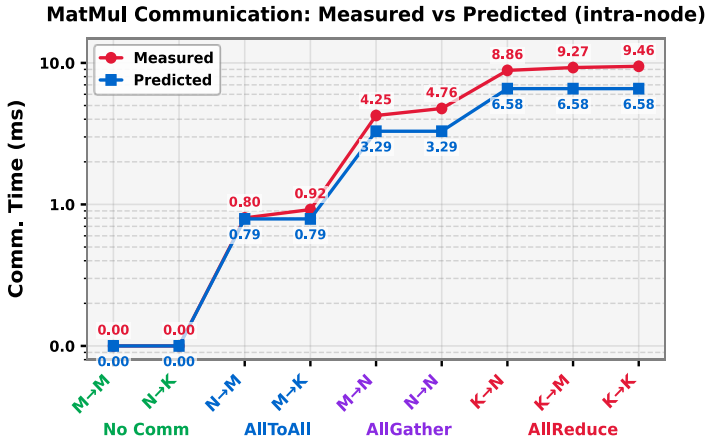
Cost Model Validation

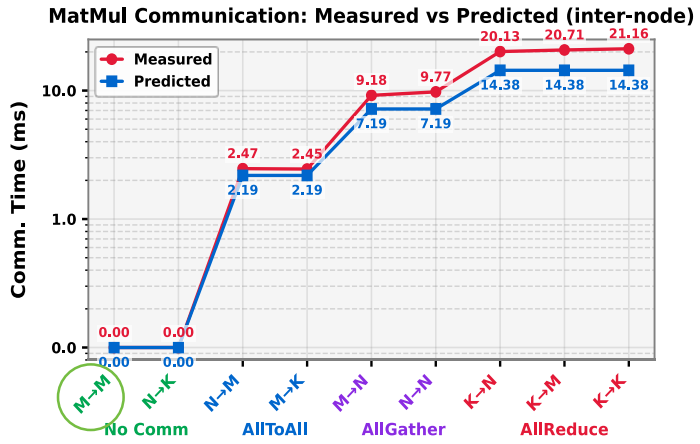
Best choice of strategy (Memory)



Memory

- Memory-aware cost model
  - Precisely predicts the lowest relative cost strategy (“*”)





Best choice of strategy (Communication)

Communication

- SymTensor captures the relative costs between different strategies
  - Lowest predicted cost matches the optimal strategy choice

SymTensor focuses on choices
-> tolerated in a good level of precision

Real-case Validation

Generality

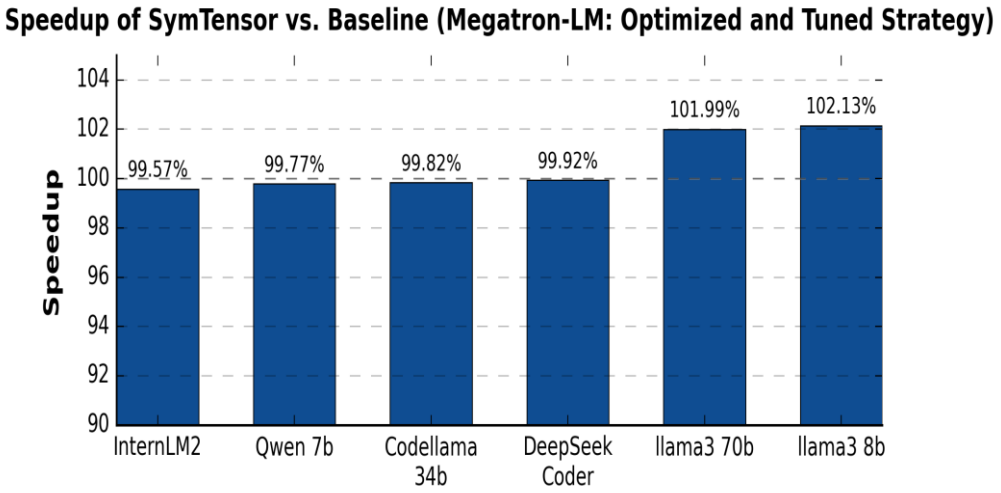
Experiment objective:

- Evaluate SymTensor's effectiveness

Experimental setup:

we chose 6 typical DL models for:

- **Dense autoregressive language models:**
  LLaMA3-8B, LLaMA3-70B
- **Instruction-tuned model for code generation:**
  Deepseek-Coder-7B
- **LLM with standard transformer backbones:**
  InternLM2-20B, CodeLLaMA-34B
- **Bilingual Model using Lora:**
  Qwen 7b



Adaptability

Experiment objective:

- Demonstrate adaptability to common real-world model variations

Experimental setup:

we chose widely used foundation models for:

- **Operator substitution** (LLama2-13B):
  Replace self-attention -> FlashAttention
- **New MoE architecture** (Mixtral 8x7B):
  Novel MoE design
- **High Memory scenario** (Qwen-7B-LoRA):
  Increase the batch size from 8 to 32

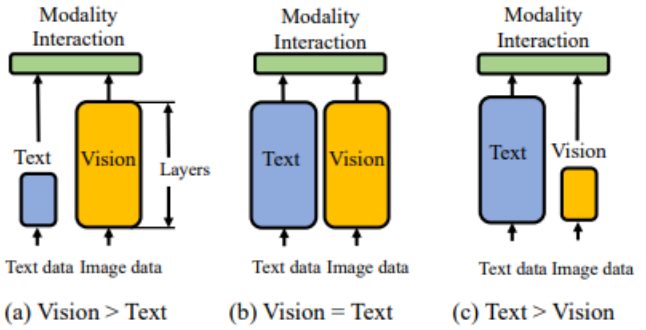
Table: Training Throughput Comparison (in tokens/sec)

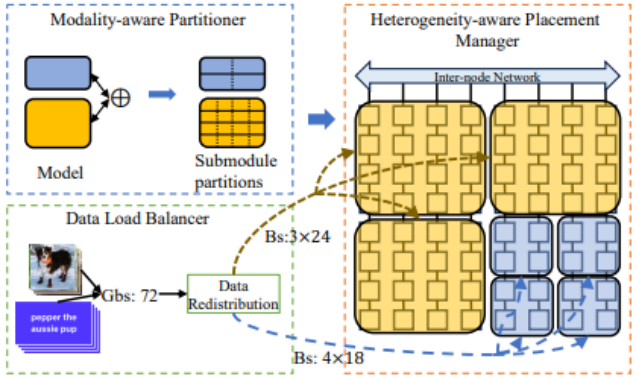
Model	Megatron-LM	SymTensor	SpeedUp
LLaMA2 13B	10961	15845	144.56%
Mixtral 8x7B	2936	6506	221.59%
Qwen 7B LoRA	OOM	17626	∞

Conclusion & More...

We demonstrated how to systematically optimize:
- Training
- for LLMs
- over typical AI-accelerator Clusters

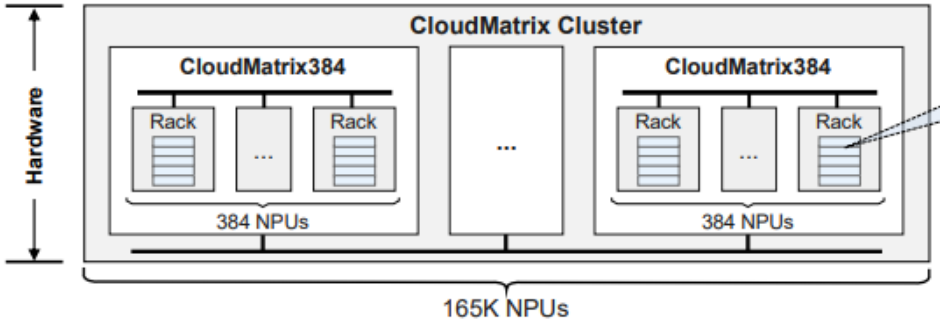
We are open to further research challenges on :
- New architectures (superPod, …)
- New AI model (multi-modal, …)
- Other distributed DL scenario (inference, …)
- And all topics related to parallel and AI (for perf, …)

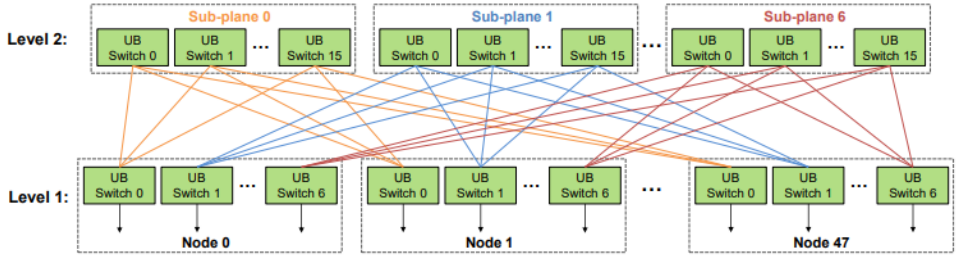


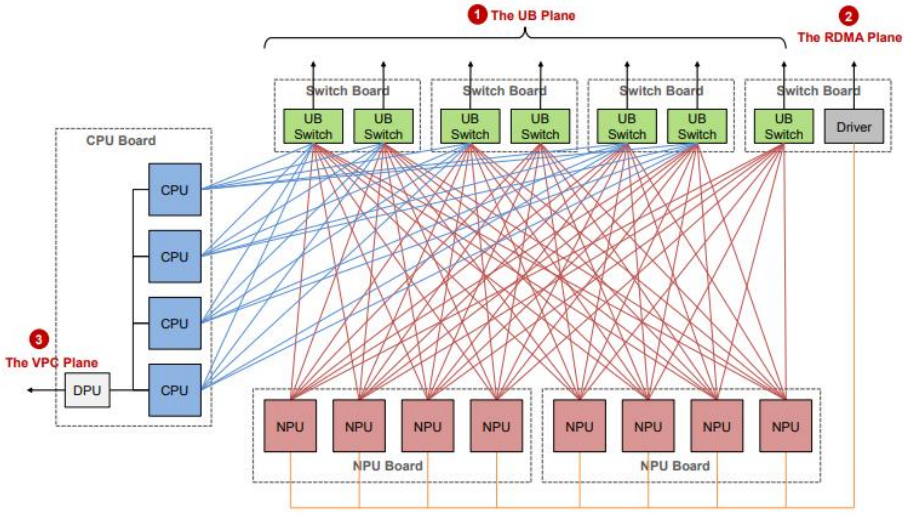


https://www.usenix.org/system/files/nsdi24-huang.pdf

Huawei CloudMatrix384





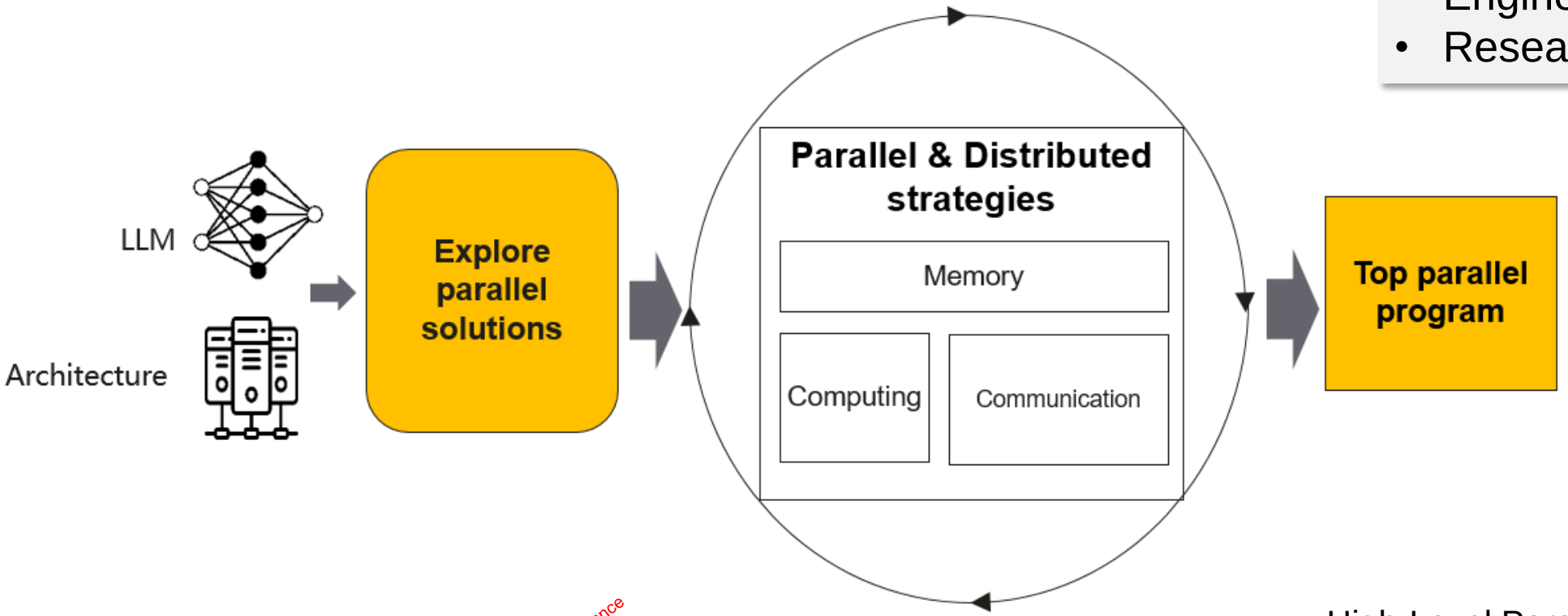


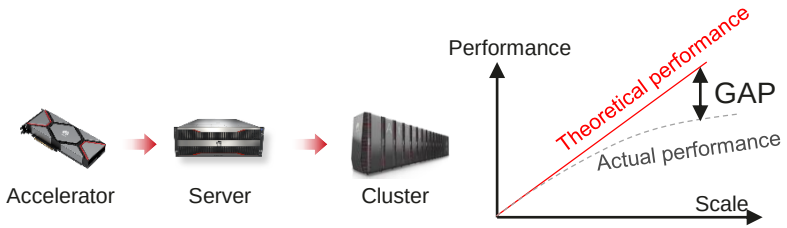
https://arxiv.org/pdf/2506.12708

Thank you

We are hiring:

- Interns
- Industrial PhDs
- Postdocs
- Engineers
- Researchers





High-Level Parallel Programming for AI
Algorithmic Performance on Distributed Systemes

Paris Distributed and Parallel Technologies Lab
Huawei France & Central Software Institute