Alpa: Automating Inter- and Intra-Operator Parallelism for Distributed Deep learning

Lianmin Zheng, Zhuohan Li, and Hao Zhang, *UC Berkeley*; Yonghao Zhuang, *Shanghai Jiao Tong University*; Zhifeng Chen and Yanping Huang, *Google*; Yida Wang, *Amazon Web Services*; Yuanzhong Xu, *Google*; Danyang Zhuo, *Duke University*; Eric P. Xing, *MBZUAI and Carnegie Mellon University*; Joseph E. Gonzalez and Ion Stoica, *UC Berkeley*

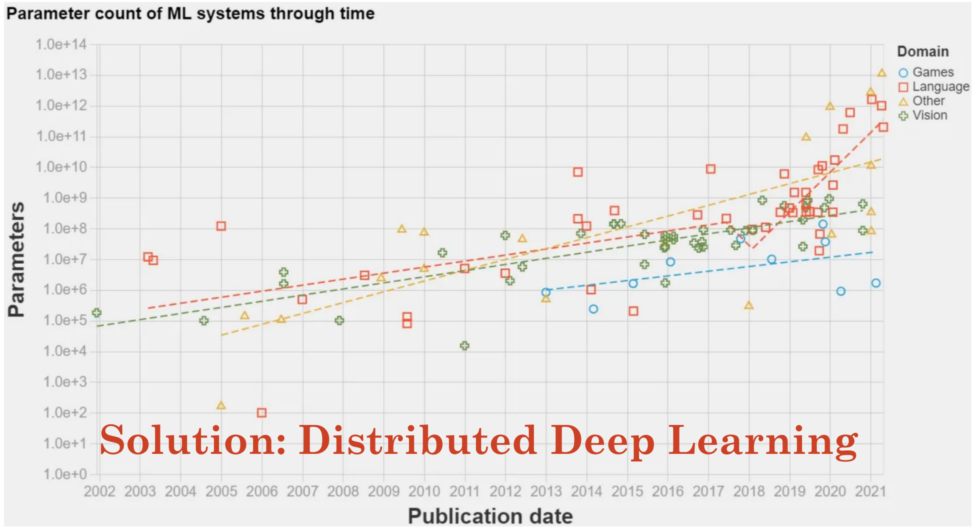
16th USENIX Symposium on Operating Systems Design and Implementation (OSDI 2022)

Presenter
Dipak Acharya
University of North Texas, Denton, Texas

Deep Learning Models

Deep learning models' size is continuously increasing

Parameter size in Billions and Trillions



Solution: Distributed Deep Learning

Alpa: Automating Inter- and Intra-Operator Parallelism for Distributed Deep learning

GPT-3 - 175B parameters

~325 GB(assuming FP16 parameters)

GPT-4 - 1.76T parameters (speculated)

~3300 GB(assuming FP16 parameters)

Intermediate activation values, gradient values, Optimizer states etc. are not included

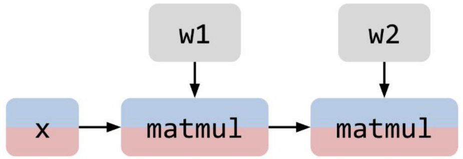
Reality: 10B parameters requires ~200GB

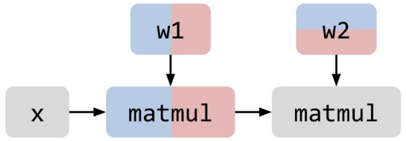
One of the popular accelerators for deep learning (Nvidia A100) has 80GB Memory

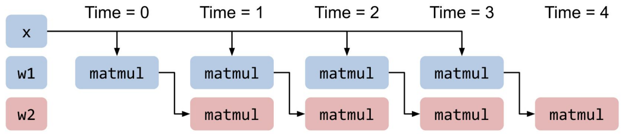
Source: https://towardsdatascience.com/parameter-counts-in-machine-learning-a312dc4753d0

Distributed Deep learning: Conventional View

🟦 Device 1 🟥 Device 2 ⬜ Replicated ▤ Row-partitioned ▥ Column-partitioned







Data Parallelism

Split the data among devices

Replicate the model

Aggregate the update from all devices

Operator Parallelism

Split the operator (non-batch axis)

Compute parts of operator on different device

Portion of tensor may be in different device; requires communication to fetch

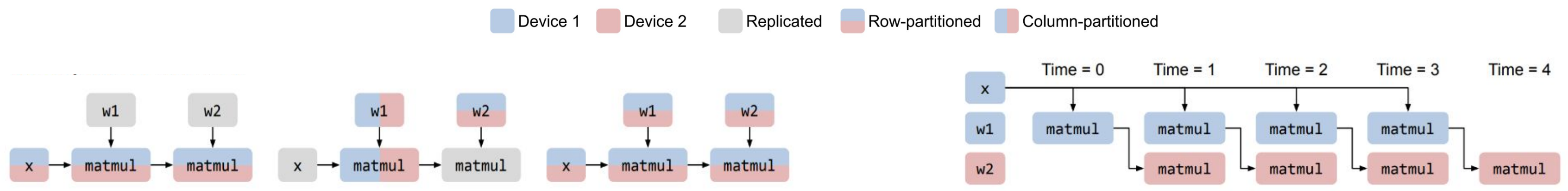
Pipeline Parallelism

Group operator into *stages*

Split training batch into microbatches

Pipeline forward/backward pass of microbatches on distributed workers

Distributed Deep learning: A Different View



Intra Operator Parallelism

Split a single operator along some axis

Perform partitioned computations on different devices

Collective communication for split and merge

Inter Operator Parallelism

Do not partition operators

Assign different operators to execute on different devices

Communication between pipeline stages using point to point communication

Intra and Inter - Operator Parallelism

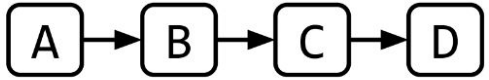
Trade off	Intra-Operator	Inter-Operator
Communication	High	Low
Device Idle time	Low	High

Intra operator parallelism requires a lot of collective communication

Inter operator parallelism requires less communication (uses point to point communication between stage meshes)

Distributed Deep Learning

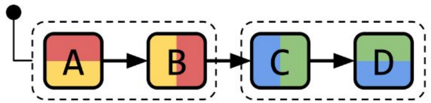
A single type of parallelism doesn't guarantees best performance



(a) Computational graph

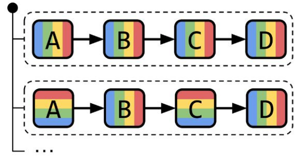
One approach is to create a manual execution plan including different parallelism

Problem: Manual plan cannot encompass all possible plans



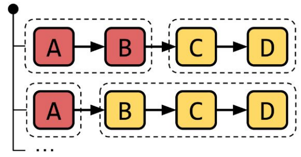
(b) Manual plan (e.g., Megatron-LM [40])

The search space for both intra and inter operator parallelism is very large

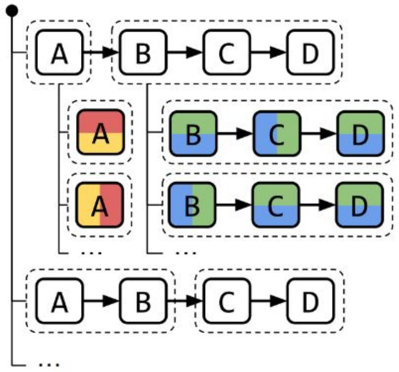


(c) The space of intra-operator parallelism (e.g., Tofu [55])

Combining the both will make it even larger



(d) The space of inter-operator parallelism (e.g., DAPPLE [17])



(e) Our hierarchical space

A Hierarchical Search Space with both Inter and Intra-Operator Parallelism

Existing Approaches

Existing models allow optimizations such as

- Intra - Operator (Megatron-LM, Mesh-Tensorflow, Tofu, Flexflow)
- Inter - Operator (GPipe, PipeDream, Megatron-LM V2)
- Automatic (Dapple, PipeDream, ZeRO, Tofu, Flexflow)

Alpa Automatically finds and executes best Inter-op and Intra-op parallelism for large deep learning models

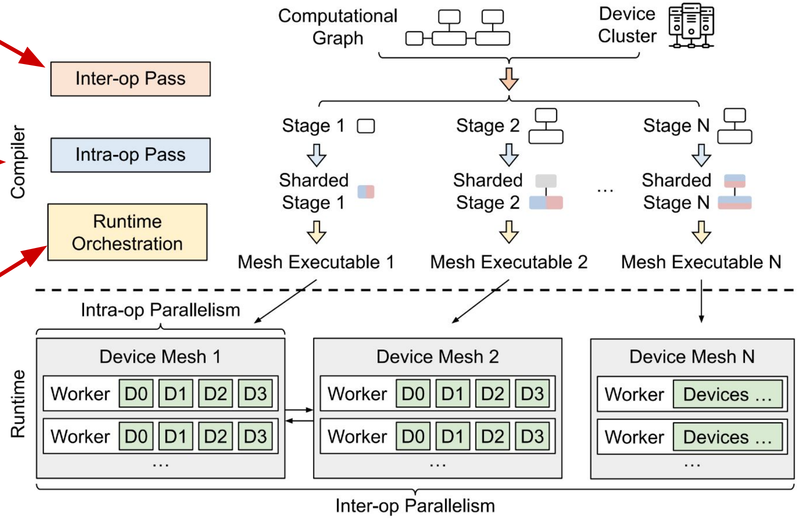
Alpa: Overview

Alpa optimizes execution of DL model hierarchically

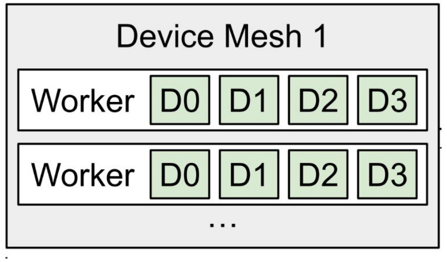
Slice model into stages and cluster into device meshes

Optimizes the intra op execution cost for given stage-mesh pair

Generates static instructions to fulfill the communication requirements



Intra-Operator Parallelism



Device Mesh:

- 2D Logical View of devices
- It has same bandwidth along axes
- One way is nodes in one axis, devices on same node in other axis

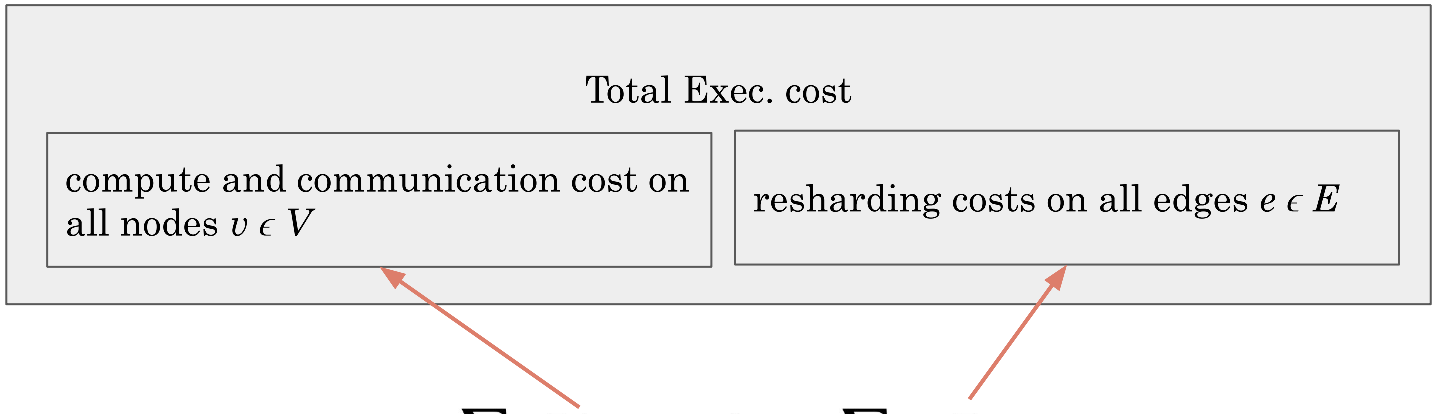
Sharding spec: Defines the layout of tensor, weather a tensor is partitioned or replicated along each axis

Resharding: If sharding spec doesn’t match selected algorithm, needs resharding
Might require cross device communication

Parallel Algorithms: Possible parallel algorithms for every primitive operator are manually enumerated

Intra-Operator Parallelism

For graph $G = (V, E)$


Total Exec. cost
compute and communication cost on all nodes $v \in V$
resharding costs on all edges $e \in E$

$$\min_{s} \sum_{v \in V} s_v^\intercal (c_v + d_v) + \sum_{(v,u) \in E} s_v^\intercal R_{vu} s_u,$$

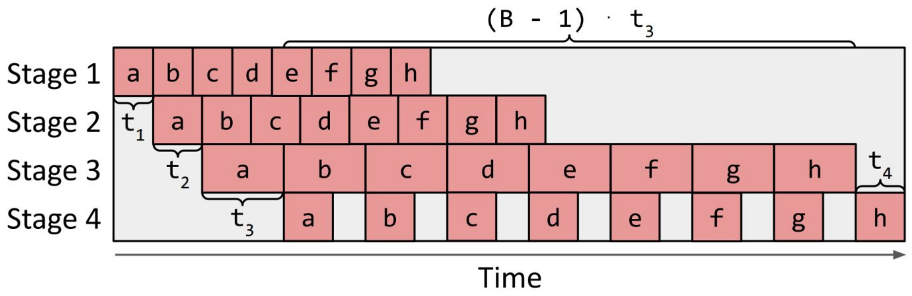
This problem is formalized as an **ILP(Integer Linear Problem)** and solved using an off-the-shelf ILP solver.

Inter-Operator Parallelism

Generate Stage-submesh pair with best for end to end latency

Latency$_{e2e}$ = Sum of all stage latencies + (B-1) * latency of longest stage

$$T^* = \min_{\substack{s_1,\ldots,s_S;\\(n_1,m_1),\ldots,(n_S,m_S)}} \left\{ \sum_{i=1}^{S} t_i + (B-1) \cdot \max_{1 \le j \le S} \{t_j\} \right\}.$$



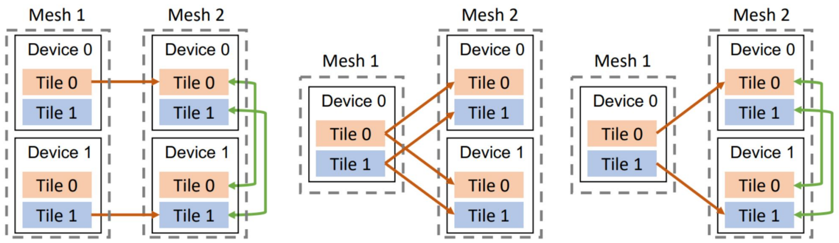
- Enumerates all possible stage-mesh combination where stage latency is obtained from Intra-Operator pass
- End to end latency is optimized by a new **DP algorithm**

Parallelism Orchestration

Cross mesh resharding

2 adjacent stages may require many-to-many multicast communication

Alpa solves this by introducing *local all-gather* on the destination mesh using high bandwidth communication



(a) Megatron-LM (b) Naïve send/recv (c) Local all-gather

Pipeline execution instruction generation

Alpa generates distinct static execution instruction for each mesh

Compared to many SPMD pipeline-parallel systems alpa adopts MPMD approach

Instructions include allocation/deallocating tensors, communication instructions, cross-mesh resharding plans, synchronization etc.

Evaluation

Baselines:

GPT-3: Megatron-LM v2

GShard MoE: DeepSpeed

Wide ResNet: Alpa PP-DP (Use alpa but search space is only data parallelism and pipeline parallelism)

Also trained only using Alpa's inter and intra operator parallelism

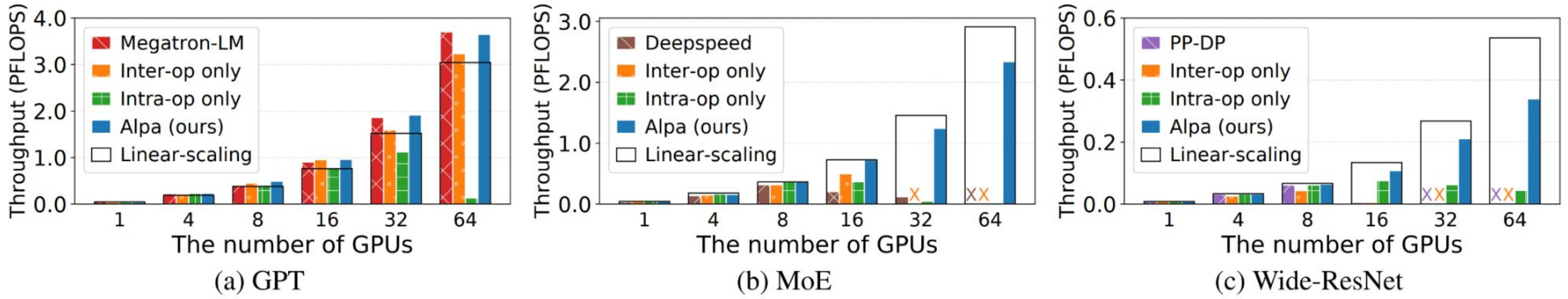
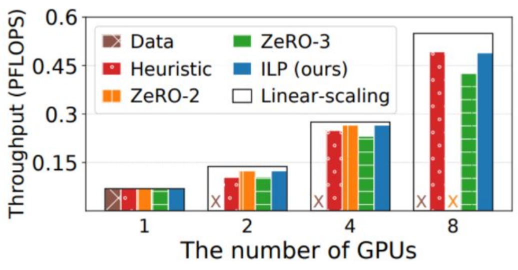
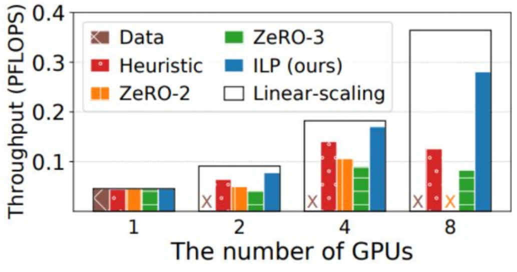


Figure 7: End-to-end evaluation results. "×" denotes out-of-memory. Black boxes represent linear scaling.

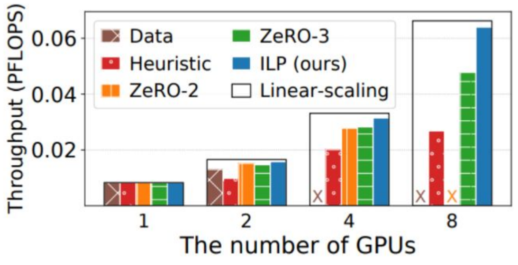
Evaluation: Ablation Study



(a) GPT



(b) MoE



(c) Wide-ResNet

Comparing automatic solutions for intra-operator parallelism

Other approaches either run out of memory or fail to scale as well compared to ILP

Evaluation: Ablation Study

Alpas inter-operator parallelism(DP) can cluster stages based on compute and communication costs

Clustering stages equally works well on homogeneous models(eg. GPT) but Alpa outperforms others on models such as Wide-ResNet

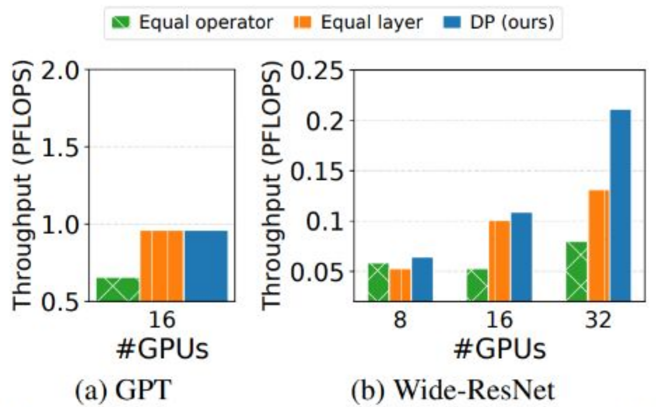


Figure 9: Inter-operator parallelism ablation study.

Contributions

- Construct 2-level Hierarchical parallel execution space combining inter and intra - operator parallelism
- Design algorithms to to get near-optimal execution plans for each level
- Implement Alpa system for the distributed GPU cluster
  - Multiple compilation passes for generating execution plans using hierarchical optimization algorithms
  - A runtime architecture that orchestrates the inter-op parallelism between device meshes
  - Several optimizations for compilation and cross-mesh communication
- Evaluate Alpa on training large models with billions of parameters

Limitations

- Does not model the communication cost between different stages
  - Communication cost is small
  - Modeling this will require exponentially more intra-op passes and DP states
- Number of microbatches B for inter-operator parallelism is not optimized
- Inter-op pass models pipeline parallelism with a static linear schedule
  - Eg. does not models scheduling different branches of a graph on different meshes
- Does not handles graphs with overlapping computation and communication
  - Tensor shapes should be known at compilation time

Potential Improvements

- Pruning the search space for faster search on ILP and DP
  - Currently the algorithms consider all possible combinations for optimization
  - It may be possible to make it faster by removing some low performing combinations from the search space based on heuristics

- Introducing more levels in the search hierarchy
  - Alpa introduces hierarchical model parallelism with 2 levels (intra and inter operator)
  - It may be possible to introduce more levels into this search space to make the optimization more efficient

Thank You