Automating Inter- and Intra-Operator Parallelism for Distributed Deep Learning

Lianmin Zheng¹,*  Zhuohan Li¹,*  Hao Zhang¹,*  Yonghao Zhuang⁴
Zhifeng Chen³  Yanping Huang³  Yida Wang²  Yuanzhong Xu³  Danyang Zhuo⁶
Eric P. Xing⁵  Joseph E. Gonzalez¹  Ion Stoica¹

¹UC Berkeley  ²Amazon Web Services  ³Google  ⁴Shanghai Jiao Tong University
⁵MBZUAI & Carnegie Mellon University  ⁶Duke University

*Equal contribution
Background
Large Models Enable Breakthroughs in Machine Learning

OpenAI debuts gigantic GPT-3 language model with 175 billion parameters

10,000× model size increase in 3 years

Training Large Models:
Not a machine learning challenge, but an OSDI challenge!

Image source: towardsdatascience.com
What are System Challenges?

1. What if the **input dataset** is very large?  
2. What if the **model** is very large?
What are System Challenges?

1. What if the input dataset is very large?  🎉 Easy.
   Use data parallelism: partition input data and replicate the model

2. What if the model is very large?  😞 Hard !!

**Challenge:** How to partition a computational graph?
Partition Computational Graphs

Strategy 1: Inter-operator Parallelism

Strategy 2: Intra-operator Parallelism

Trade-off

<table>
<thead>
<tr>
<th></th>
<th>Inter-operator Parallelism</th>
<th>Intra-operator Parallelism</th>
</tr>
</thead>
<tbody>
<tr>
<td>Communication</td>
<td>Less</td>
<td>More</td>
</tr>
<tr>
<td>Device Idle Time</td>
<td>More</td>
<td>Less</td>
</tr>
</tbody>
</table>
Partition Computational Graphs

Multiple intra-op strategies for a single node
- Row-partitioned
- Column-partitioned
- Replicated

Pipeline the execution for inter-op parallelism

Combine Intra-op and Inter-op

Training Throughput of an MoE Model
Network Topology

Challenge: How to handle heterogeneous network topology?
Prior Works

Inter-op Parallelism
(w/ pipeline)

Intra-op Parallelism
(w/ operator-level)

Megatron-LM
Mesh-Tensorflow
GShard
Megatron-LM V2
Tofu
FlexFlow
Alpa
(Ours)

GPipe
PipeDream
Dapple
ZeRO
Unity [OSDI’22]
Automatic
Alpa Compiler
A unified **compiler** that **automatically** finds and executes the best **Inter-op** and **Intra-op** parallelism for **large** deep learning models.
def train_step(model_state, batch):
    def loss_func(params):
        out = model_state.forward(params, batch["x"])
        return np.mean((out - batch["y"])**2)

    grads = grad(loss_func)(state.params)
    new_model_state = model_state.apply_gradient(grads)
    return new_model_state

# A typical JAX training loop
model_state = create_train_state()
for batch in data_loader:
    model_state = train_step(model_state, batch)
Alpa’s Main Contributions

- Two-level hierarchical space of parallelism techniques.
- Effective optimization algorithms at each level.
- Efficient compiler and runtime system implementation.
Computational Graph

Whole Search Space

Alpa Hierarchical Space

Inter-op Parallelism

Intra-op Parallelism
Alpa Compiler: Hierarchical Optimization

Computational Graph

Dynamic Programming

Inter-op Pass

Cost Estimation

Integer Linear Programming

Intra-op Pass

Runtime Orchestration

Device Cluster
Inter-op Pass

Computational Graph

x → conv → relu → conv → add → avgpool → matmul → relu → matmul → softmax
Inter-op Pass

Graph Partitioning

Stage 1

Stage 2

Stage 3

Stage 4

or

or

...
Inter-op Pass

Partitioned Computational Graph

Device Assignment
Inter-op Pass

Partitioned Computational Graph

Stage 1: $k_1$ to conv, relu, conv, add

Stage 2: $k_2$ to avgpool

Stage 3: $w_1$ to matmul, relu, matmul, softmax

Stage 4: $w_2$ to softmax

Cluster (2D Device Mesh)

Nodes

GPUs within a Node
Inter-op Pass

Stage 1:
- x
- conv
- relu
- conv
- add

Stage 2:
- avgpool

Stage 3:
- matmul
- relu
- matmul

Stage 4:
- softmax

Submesh Choice 1

Submesh Choice 2

or

or

or...

k1

k2

w1

w2
Inter-op Pass

Solved together by Dynamic Programming

More details on the DP algorithm can be found in the paper.
Intra-op Pass

Stage

Solved by
Integer Linear Programming

Submesh

Stage with intra-operator parallelization
Minimize Computation cost + Communication cost

Decision vector
Parallel strategies of each operator

Minimize Computation cost + Communication cost

More details on the ILP algorithm can be found in the paper.
Compilation Time Optimization

- Communication-aware operator clustering in ILP & DP
- Early stopping in DP
- Distributed Compilation

**Alpa Compilation Time:** < 40 min for the largest experiment.

- Can be further reduced by at least 50% with search space pruning.
Runtime Orchestration

Intra-op Parallelism

Compilation

Parallelized Stage 1

Parallelized Stage 2

Parallelized Stage n

Static Mesh Executable 1

Static Mesh Executable 2

Static Mesh Executable n

Submesh 1

Submesh 2

Submesh n

Cross-mesh Communication

Inter-op Parallelism
Evaluation
Evaluation: Comparing with Previous Works

Weak scaling results where the model size grow with #GPUs.
Evaluated on 8 AWS EC2 p3.16xlarge nodes with 8 16GB V100s each (64 GPUs in total).

GPT (up to 39B)

- Match specialized manual systems.

GShard MoE (up to 70B)

- Outperform the manual baseline by up to 8x.

Wide-ResNet (up to 13B)

- Generalize to models without manual plans.
Evaluation: Ablation Study with Inter-op and Intra-op Only

Combining inter- and intra-operator parallelism scales to more devices.
Case Study: Wide-ResNet Partition on 16 GPUs.
Hierarchical view: inter-op and intra-op
Match or outperform specialized systems
Generalizes to new models

@alpa.parallelize: automatic model-parallel training

Alpa  alpa.ai

🌟 Star 390