Alpa: Automating Inter/Intra Operator Parallelism for Distributed DL

Lianmin Zheng, Zhuohan Li, Hao Zhang, Yonghao Zhuang, Zhifeng Chen, Yanping Huang, Yida Wang, Yuanzhong Xu, Danyang Zhuo, Eric P. Xing, Joseph E. Gonzalez, Ion Stoica

https://arxiv.org/pdf/2201.12023

Presenter: Ben Xia

Refresher: Data/Intra Op Parallelism

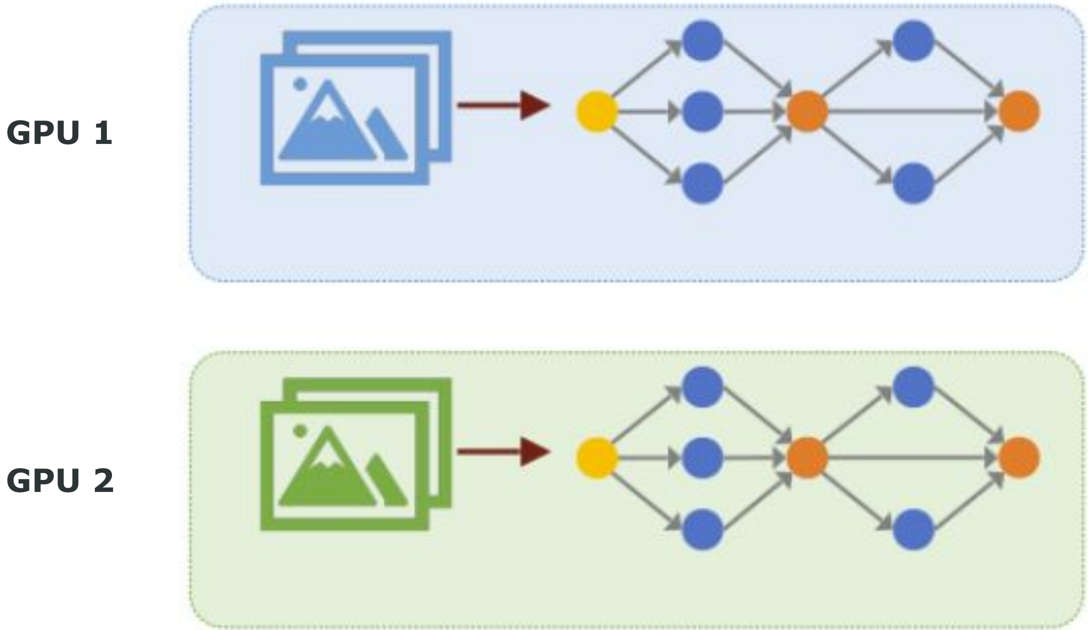


Figure from CMU 15-849 [Jia 2022]

Refresher: Model Parallelism

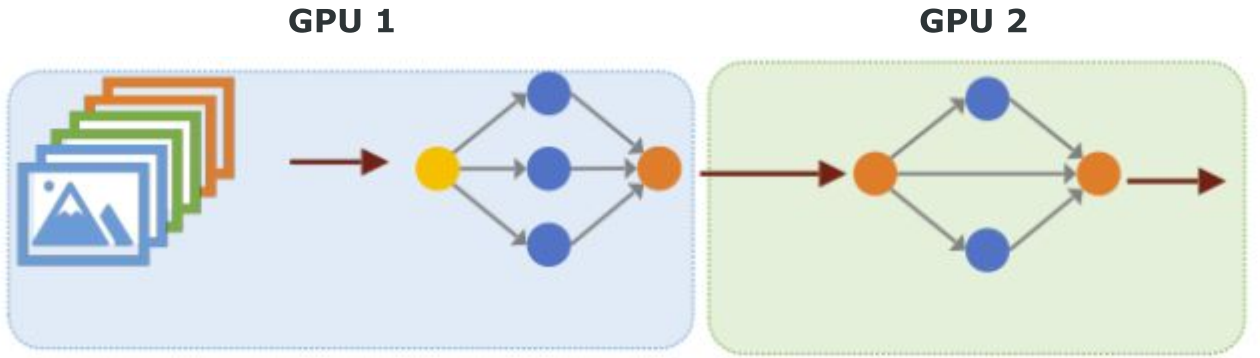
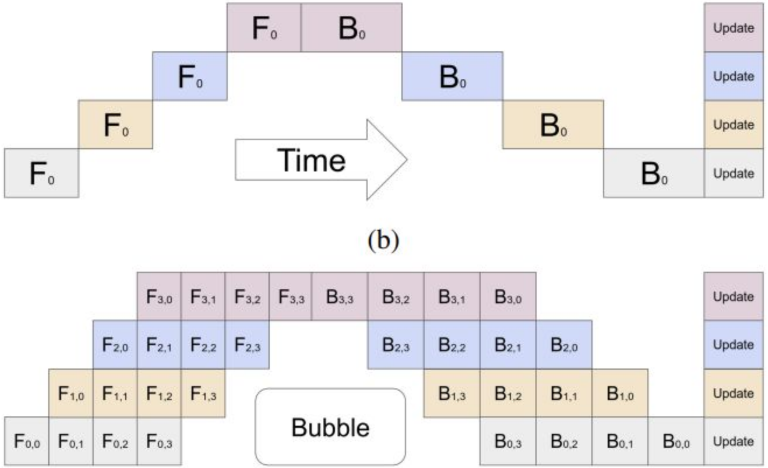
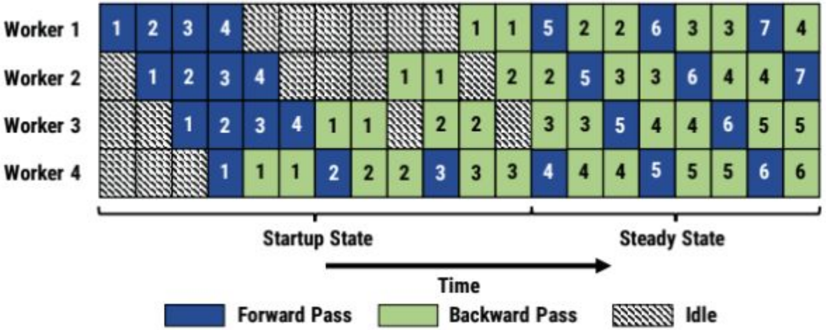


Figure from CMU 15-849 [Jia 2022]

Refresher: Pipeline Parallelism

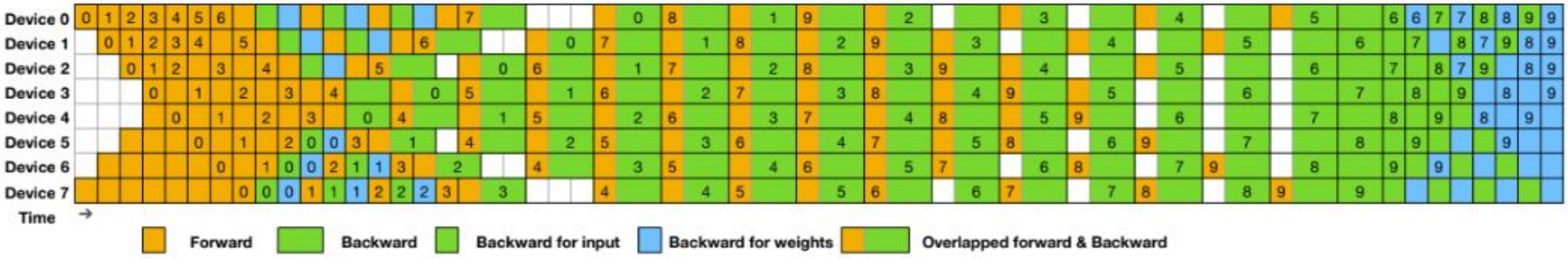


Source: GPipe: Easy Scaling with Micro-Batch Pipeline Parallelism https://arxiv.org/pdf/1811.06965



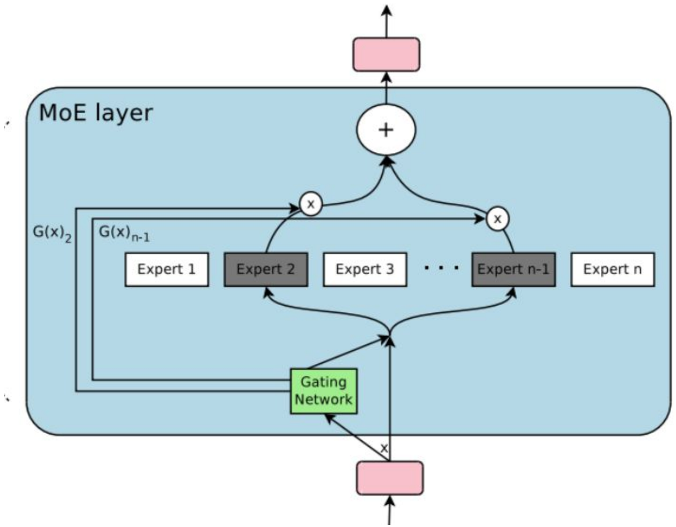
Source: PipeDream: Generalized Pipeline Parallelism for DNN Training https://arxiv.org/pdf/1806.03377

Refresher: Pipeline Parallelism

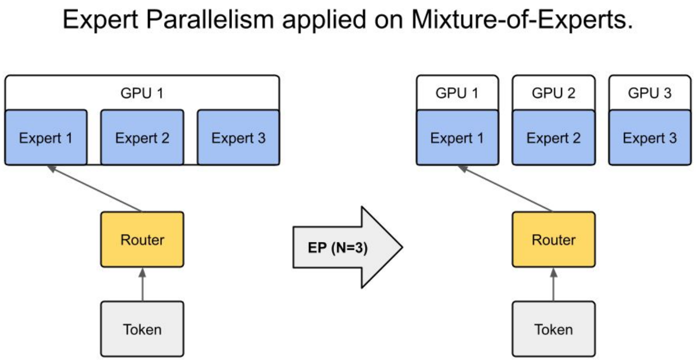


Source: DeepSeek-V3 Technical Report https://arxiv.org/pdf/2412.19437

Refresher: Expert Parallelism

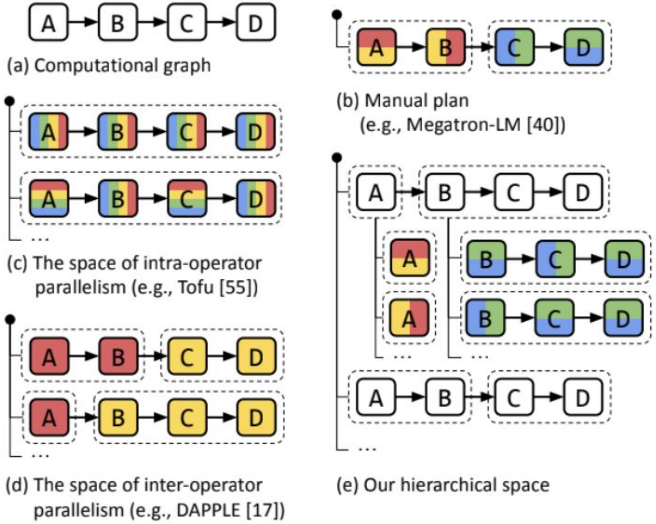


Source: Outrageously Large Neural Networks: The Sparsely Gated Mixture of Experts Layer
https://arxiv.org/pdf/1701.06538



Source: https://docs.nvidia.com/nemo-framework/user-guide/latest/nemotoolkit/features/parallelisms.html

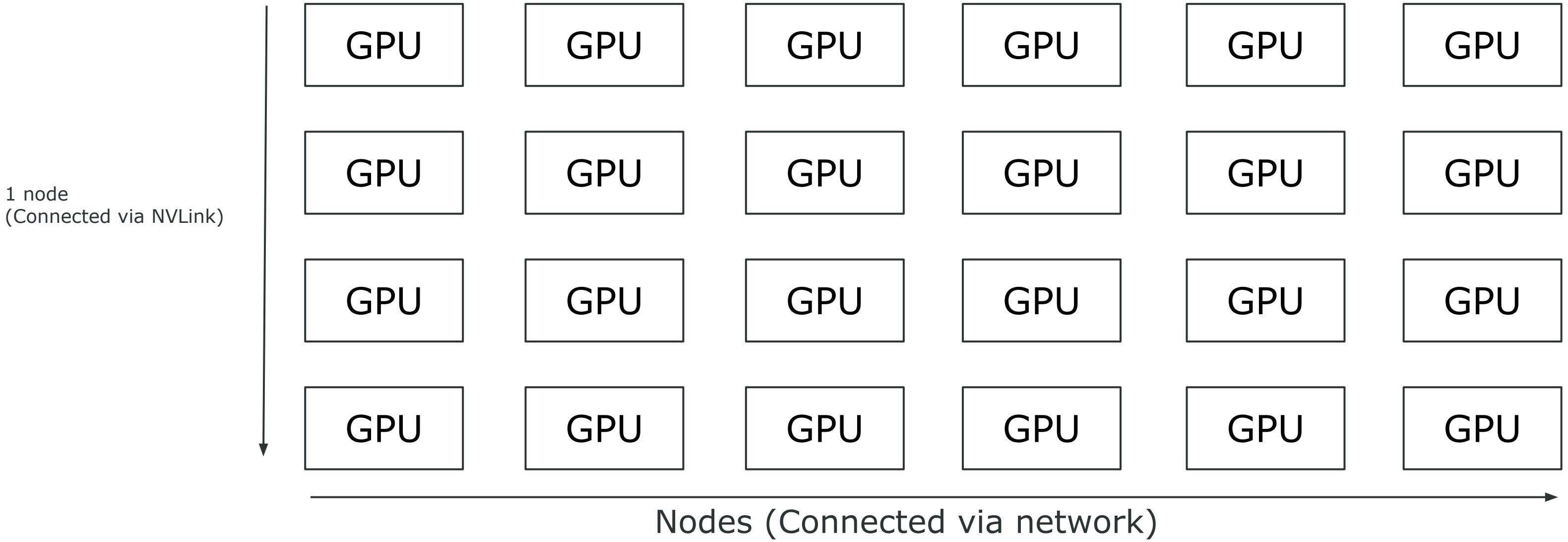
Alpa: Compiler for Parallel Execution Plans



Source: Alpa: Automating Inter- and Intra-Operator Parallelism for Distributed Deep Learning https://arxiv.org/pdf/2201.12023

Alpa: Device Mesh Formulation

- Compute clusters are viewed as a simplified 2D grid.
- Communication along each dimension of the grid w/ different bandwidths and speed.



Alpa: Hierarchical Search

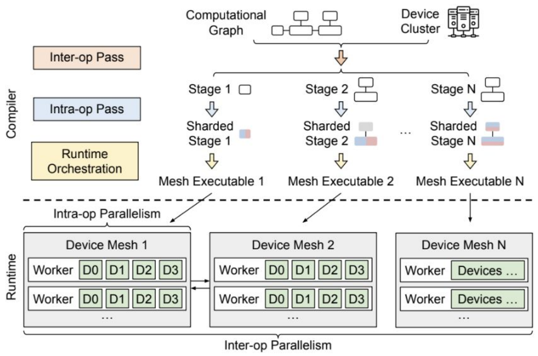
Inter-Op Pass

Dynamic Programming

Intra-Op Pass

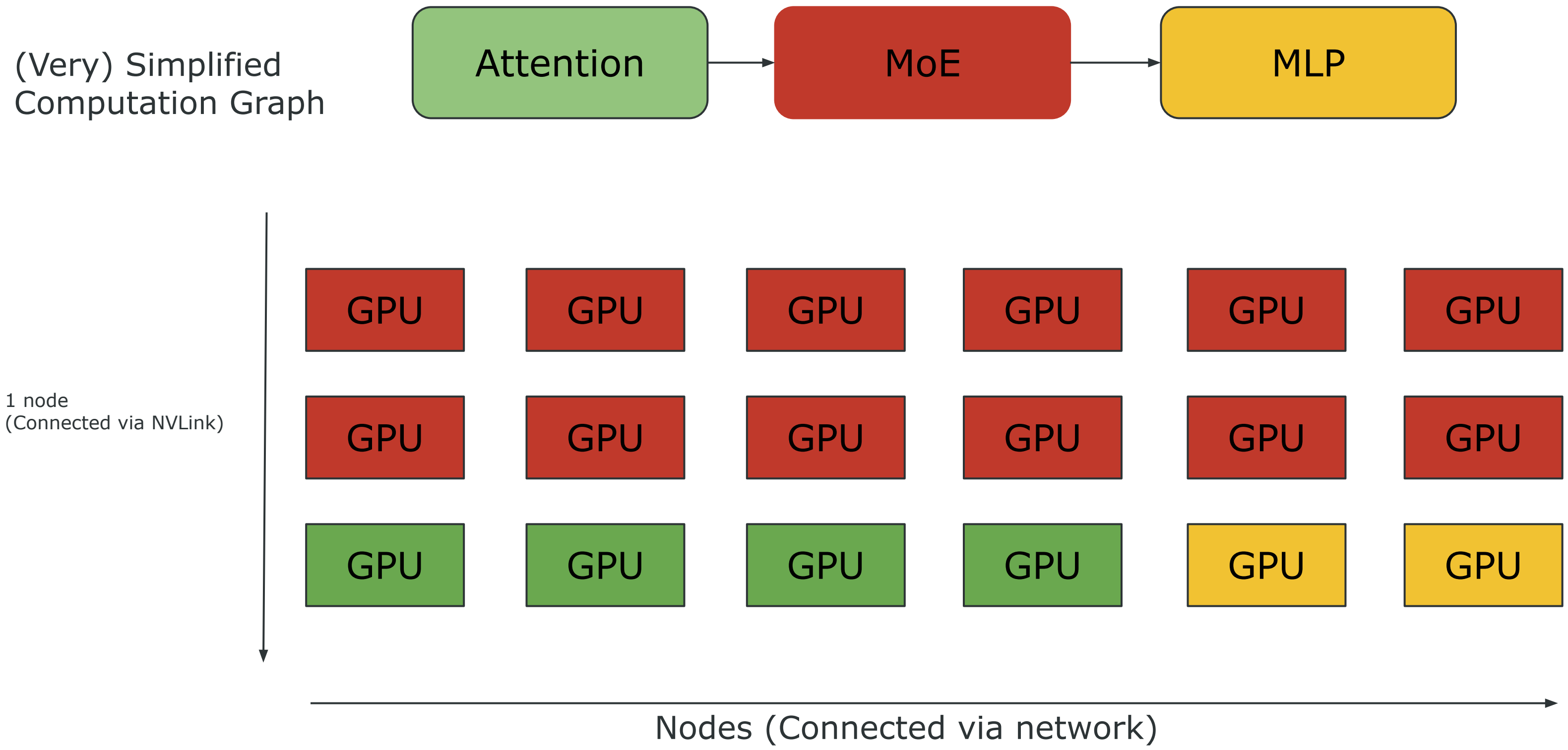
Integer Linear Programming

Runtime Orchestration



Source: Alpa: Automating Inter- and Intra-Operator Parallelism for Distributed Deep Learning https://arxiv.org/pdf/2201.12023

Alpa: Inter Op Pass



Alpa: Inter Op Pass

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

$$F(s,k,d;t_{max}) = \min_{\substack{k \le i \le K\\ n_s \cdot m_s \le d}} \left\{ \begin{array}{l} t_{intra}((o_k,\ldots,o_i), Mesh(n_s,m_s), s) \\ + F(s-1, i+1, d - n_s \cdot m_s; t_{max}) \\ \mid t_{intra}((o_k,\ldots,o_i), Mesh(n_s,m_s), s) \le t_{max} \end{array} \right\}$$

$$T^*(t_{max}) = \min_s \{F(s,0,N \cdot M; t_{max})\} + (B-1) \cdot t_{max}.$$

$$G(k,r) = \min_{1 \le i \le k} \left\{ \begin{array}{l} \max\{G(i-1,r-1), C(i,k)\} \\ \mid FLOP(o_i,\ldots,o_k) \le \dfrac{(1+\delta) FLOP_{total}}{L} \end{array} \right\}$$

Find the optimal submesh assignments via dynamic programming to minimize total runtime.

Ok this one is for operator clustering but that's still inter-op

Intra Op Pass

- Once we have a submesh assignment for a group of operators, we need to determine how to optimize its runtime.
  - Different subsets of a layer can either be sharded or replicated across multiple workers.

Spec	Device 0	Device 1	Device 2	Device 3
RR	$A[0:N, 0:M]$	$A[0:N, 0:M]$	$A[0:N, 0:M]$	$A[0:N, 0:M]$
S^0S^1	$A[0:\frac{N}{2}, 0:\frac{M}{2}]$	$A[0:\frac{N}{2}, \frac{M}{2}:M]$	$A[\frac{N}{2}:N, 0:\frac{M}{2}]$	$A[\frac{N}{2}:N, \frac{M}{2}:M]$
S^1S^0	$A[0:\frac{N}{2}, 0:\frac{M}{2}]$	$A[\frac{N}{2}:N, 0:\frac{M}{2}]$	$A[0:\frac{N}{2}, \frac{M}{2}:M]$	$A[\frac{N}{2}:N, \frac{M}{2}:M]$
S^0R	$A[0:\frac{N}{2}, 0:M]$	$A[0:\frac{N}{2}, 0:M]$	$A[\frac{N}{2}:N, 0:M]$	$A[\frac{N}{2}:N, 0:M]$
S^1R	$A[0:\frac{N}{2}, 0:M]$	$A[\frac{N}{2}:N, 0:M]$	$A[0:\frac{N}{2}, 0:M]$	$A[\frac{N}{2}:N, 0:M]$
RS^0	$A[0:N, 0:\frac{M}{2}]$	$A[0:N, 0:\frac{M}{2}]$	$A[0:N, \frac{M}{2}:M]$	$A[0:N, \frac{M}{2}:M]$
RS^1	$A[0:N, 0:\frac{M}{2}]$	$A[0:N, \frac{M}{2}:M]$	$A[0:N, 0:\frac{M}{2}]$	$A[0:N, \frac{M}{2}:M]$
$S^{01}R$	$A[0:\frac{N}{4}, 0:M]$	$A[\frac{N}{4}:\frac{N}{2}, 0:M]$	$A[\frac{N}{2}:\frac{3N}{4}, 0:M]$	$A[\frac{3N}{4}:N, 0:M]$
RS^{01}	$A[0:N, 0:\frac{M}{4}]$	$A[0:N, \frac{M}{4}:\frac{M}{2}]$	$A[0:N, \frac{M}{2}:\frac{3M}{4}]$	$A[0:N, \frac{3M}{4}:M]$

Intra Op Pass

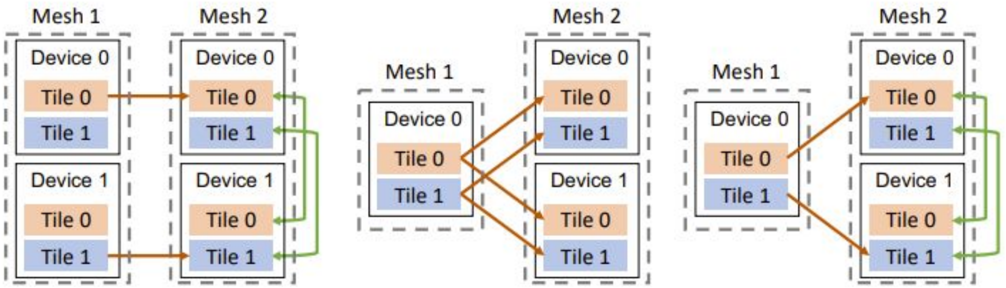
- Once we have a submesh assignment for a group of operators, we need to determine how to optimize its runtime.
    - Intra-Op Parallelism/Data Parallelism
    - Model Parallelism (Placing different operators on different devices)
    - etc.

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

Computation/Communication Cost   Resharding Cost

Solvable via Integer Linear Programming

Parallelism Orchestration

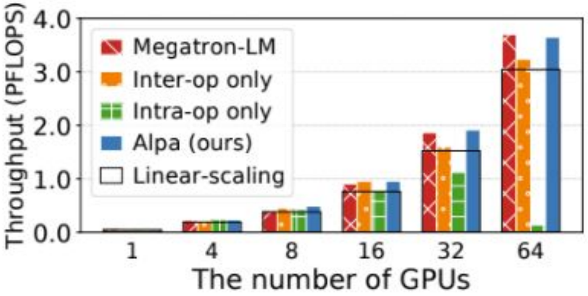


(a) Megatron-LM (b) Naïve send/recv (c) Local all-gather

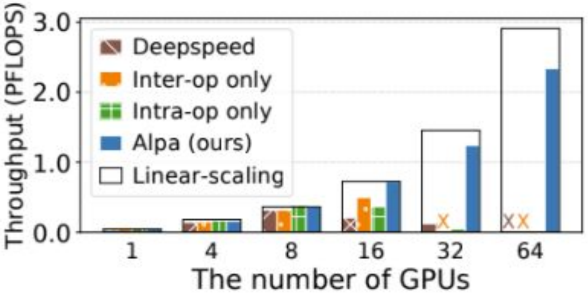
1. Find correspondences between source and destination mesh
   a. Add point-to-point send/recv's
2. Remove redundant send's (based on destination mesh sharding configuration)
   a. Allow destination to all-gather -> Faster communication than point-to-point send's

Performance

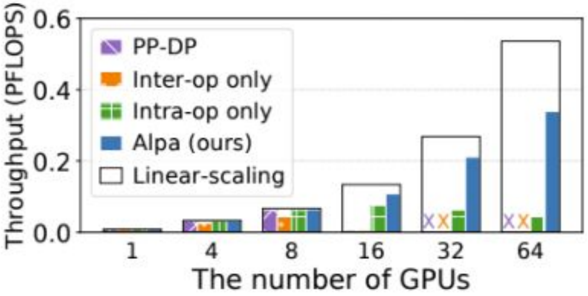
AWS p3.16xLarge



(a) GPT



(b) MoE



(c) Wide-ResNet

Roughly equal performance to Megatron, but generalizes to non-GPT architectures

Pure inter/intra op parallelism fails to scale for larger models
3.5x faster than DeepSpeed on 2 nodes, 9.7x faster on 4 nodes

Compilation Speed

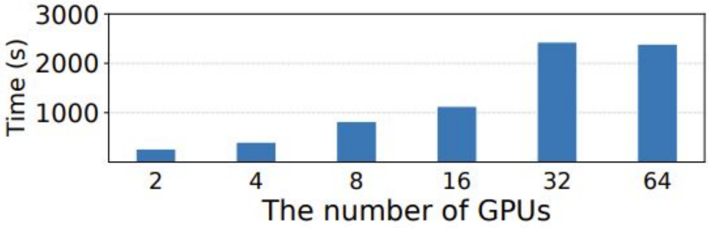


Table 5: Compilation time breakdown of GPT-39B.

Steps	Ours	w/o optimization
Compilation	1582.66 s	> 16hr
Profiling	804.48 s	> 24hr
Stage Construction DP	1.65 s	N/A
Other	4.47 s	N/A
Total	2393.26 s	> 40hr

$$O(K^5 NM(N + \log(M))^2)$$

Dynamic program asymptotic runtime