UNIVERSITY OF MICHIGAN

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

University of Michigan
EECS 583 Group 4

Presenter: Hithesh P. Reddy, Lai Wang, Noah Kaplan, Parin Senta

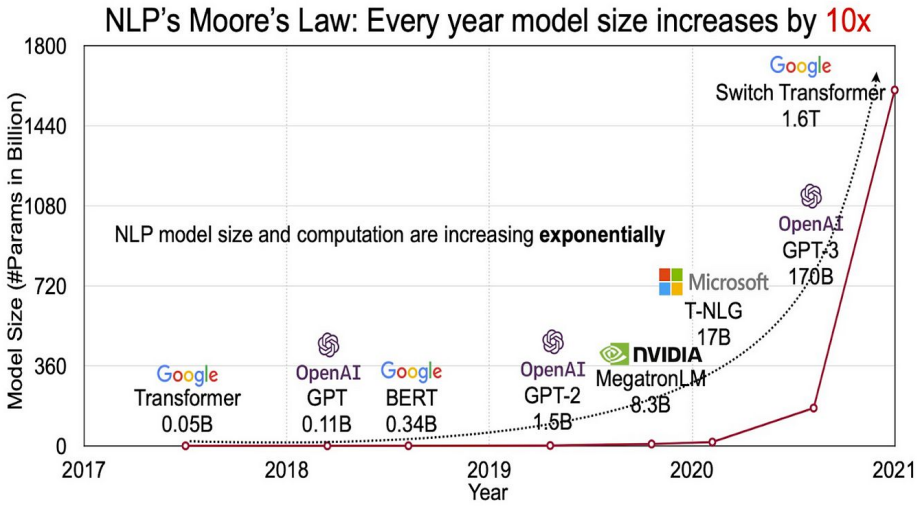
Alpa: Automating Inter- and Intra-Operator Parallelism for Distributed Deep Learning

Lianmin Zheng[1,*] Zhuohan Li[1,*] Hao Zhang[1,*] Yonghao Zhuang[4]
Zhifeng Chen[3] Yanping Huang[3] Yida Wang[2] Yuanzhong Xu[3] Danyang Zhuo[6]
Eric P. Xing[5] Joseph E. Gonzalez[1] Ion Stoica[1]

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

Motivation

- Recent advances in deep learning result in increasing model size
  - Transformer-based LLM
  - Hundreds of billions of parameters
- Large model cannot fit in single device/machine
- Distributed Deep Learning is a trend
- Parallelization strategies greatly affect training speed
  - Model & Data specific
  - Hand-craft (manual) approach needs time & expertise
- Automating the parallelization!

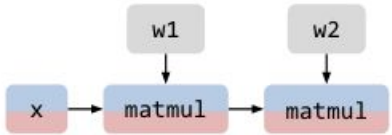


What is Alpa?

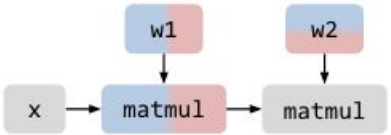
- A **compiler** generates model-parallel execution plan
  - 3 passes at IR level (Jax IR + cluster configuration)
- Optimize and utilize intra-operator parallelism and inter-operator parallelism
  - View as ILP + DP problem
  - Prune + cost model used to prune and reduce the search space
- Scales well with the number of GPUs
- Reasonable compilation time
  - < 1 hr for GPT3-36B w/ 64 GPUs

ML Parallelism

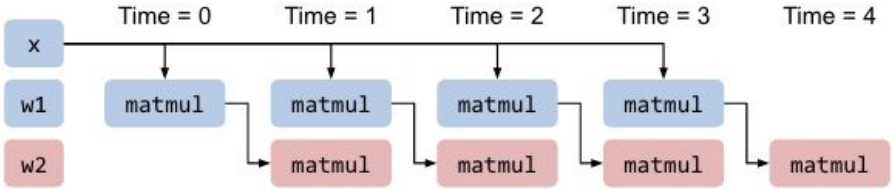
- Data Parallel
  - Partitioned data across batch, replicated model
- Operator Parallel
  - Partitioned operator across non-batch axis
  - “SPMD”, extensive communication between devices (all-reduce, gather)
- Pipeline Parallel
  - Partitioned the model into stages (“group of ops)
  - Transfer only intermediate activates between stages (point-to-point)



(a) Data Parallelism



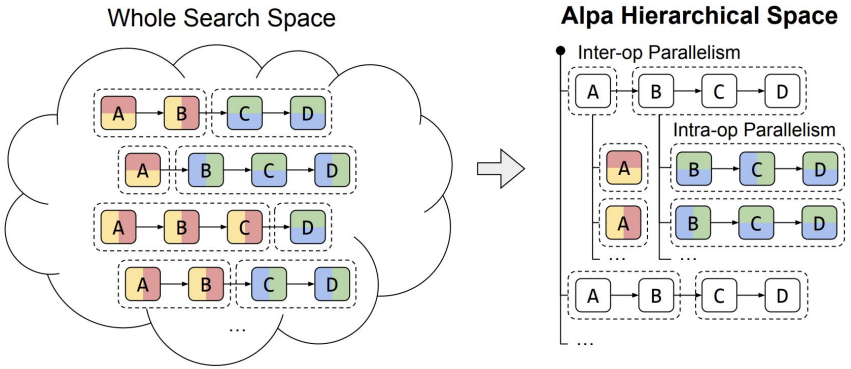
(b) Operator Parallelism



(d) Pipeline Parallelism

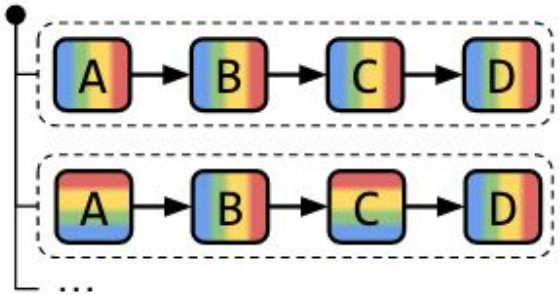
Hard Problem

- The configurations of each individual parallelism increase the search space
- Huge Combinatorial space of #devices, #parallels, #layers, #ops
  - Prior auto-parallelization are limited:
    - Combine data parallel with only one other approach to reduce the space
    - But misses performance opportunities
- Alpa: Only 2 views of parallelism, intra & inter operator
  - Distinguished by whether involve partitioning operators along any tensor axis



Alpa’s View: Intra-Operator Parallelism

- Partition of tensor along some axis so that devices execute different portions of the operator at the same time.
- Includes Data parallel & Operator parallel
- Substantial communication among devices needed.

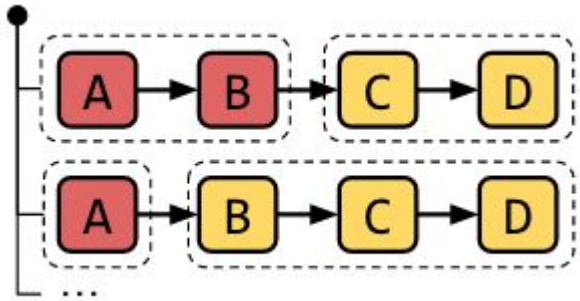


Device Idle Time - Less

Communication - More

Alpa’s View: Inter-Operator Parallelism

- Assign groups of operators to execute on distributed devices
- Divide the model into stages inside which are multiple layers
- Devices communicate only between stages.
- Some devices may idle due to dependencies

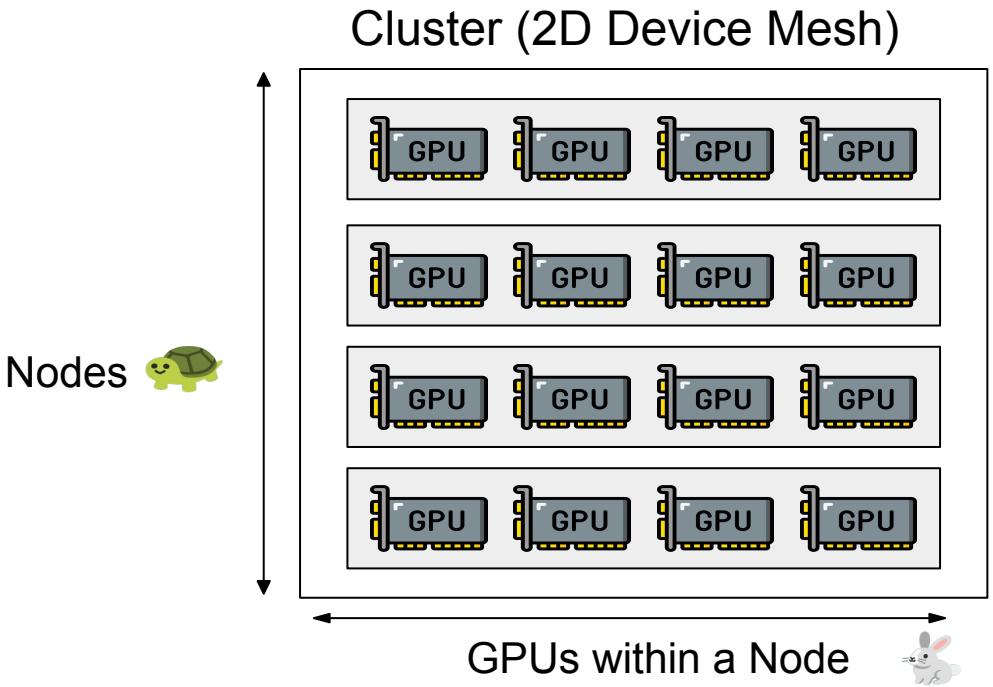


Device Idle Time - More

Communication - Less

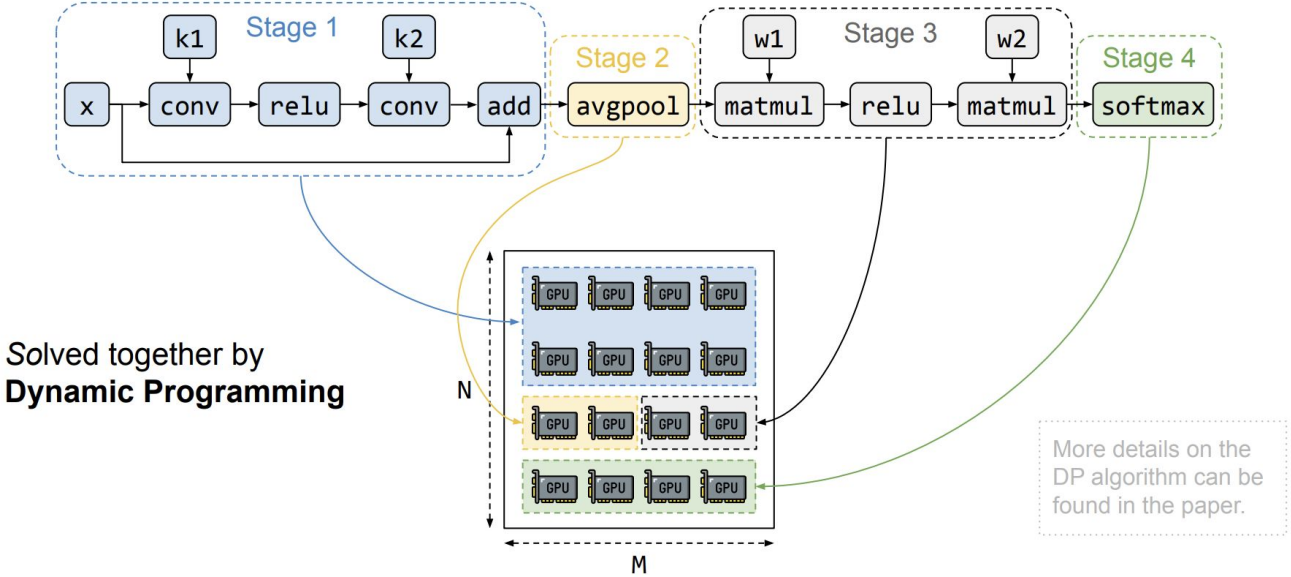
Alpa's View: Device Mesh

- Devices within the mesh have equivalent compute capability

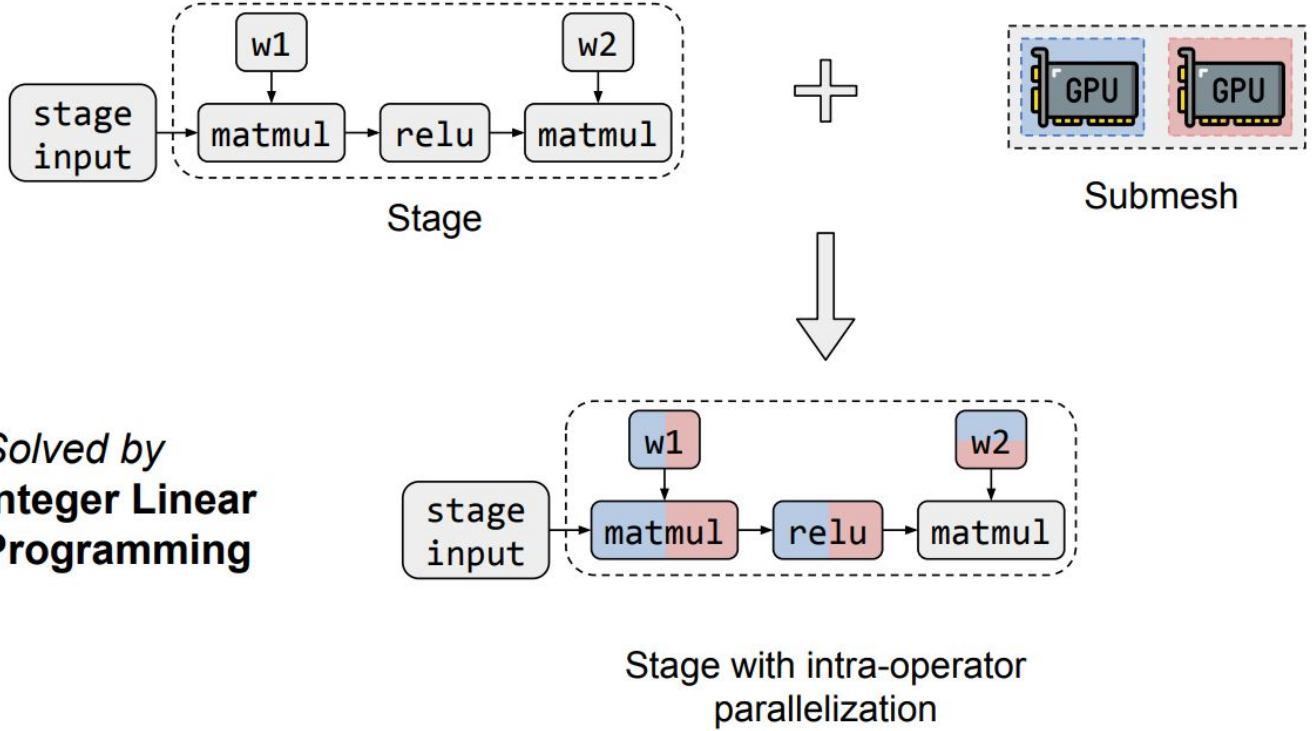


Inter-Operator Search Space

- **Partition Constraints**
  - 1D sub-mesh of shape (1, 2^m)
  - 2D sub-mesh of shape (n, M)



Intra-Operator Search Space



Optimization & Prune

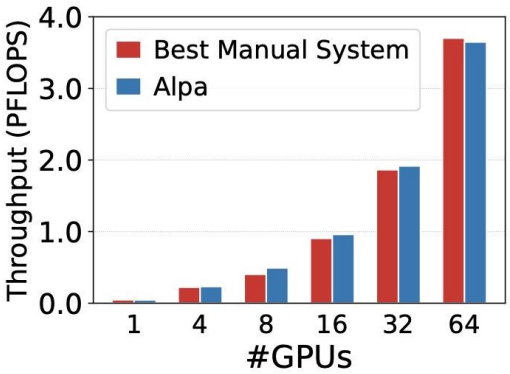
- Early stop in DP
- Compile different pipeline stage in parallel
- Operator fusion for “light operators” (ReLU, element-wise ops)
- Cost model at XLA instruction level for matmul and communication primitives.

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

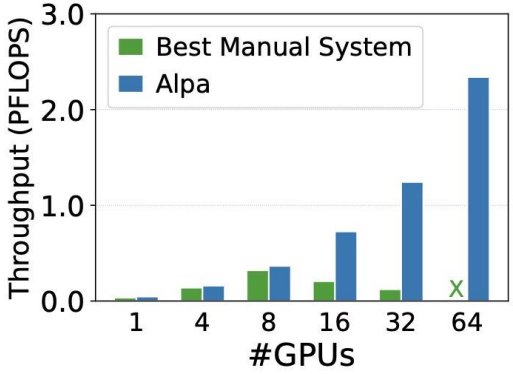
Evaluations: GPT-3, GShard MoE & Wide-ResNet

GPT (up to 39B)



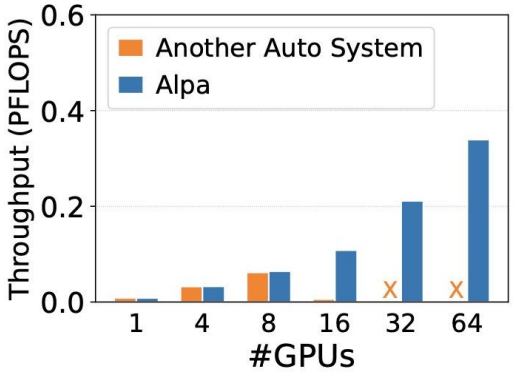
Match specialized manual systems.

GShard MoE (up to 70B)



Outperform the manual baseline by up to 8x.

Wide-ResNet (up to 13B)



Generalize to models without manual plans.

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).

Limitations

- Cannot handle heterogeneous device mappings
- Doesn’t model communication cost between pipeline stages
- Alpa lets you divide a batch into sub-batches. Must specify this as hyperparameter
- Every pipeline stage is executed serially (assumes false dependencies)
- Alpa compiles to a particular tensor shape.

Conclusion

- Alpa optimizes performance of Neural Networks across multiple nodes by exploiting parallelism inherent to DNN models
- It’s fast! (reasonably)
  - But there are opportunities to improve the compilation time
- It outperforms hand tuned partitioning!
  - But it only works for homogeneous nodes

References

Zheng, Lianmin, et al. "Alpa: Automating inter-and {Intra-Operator} parallelism for distributed deep learning." *16th USENIX Symposium on Operating Systems Design and Implementation (OSDI 22)*. 2022.

https://www.usenix.org/conference/osdi22/presentation/zheng-lianmin

Q&A