ARTIFACT EVALUATED usenix ASSOCIATION AVAILABLE

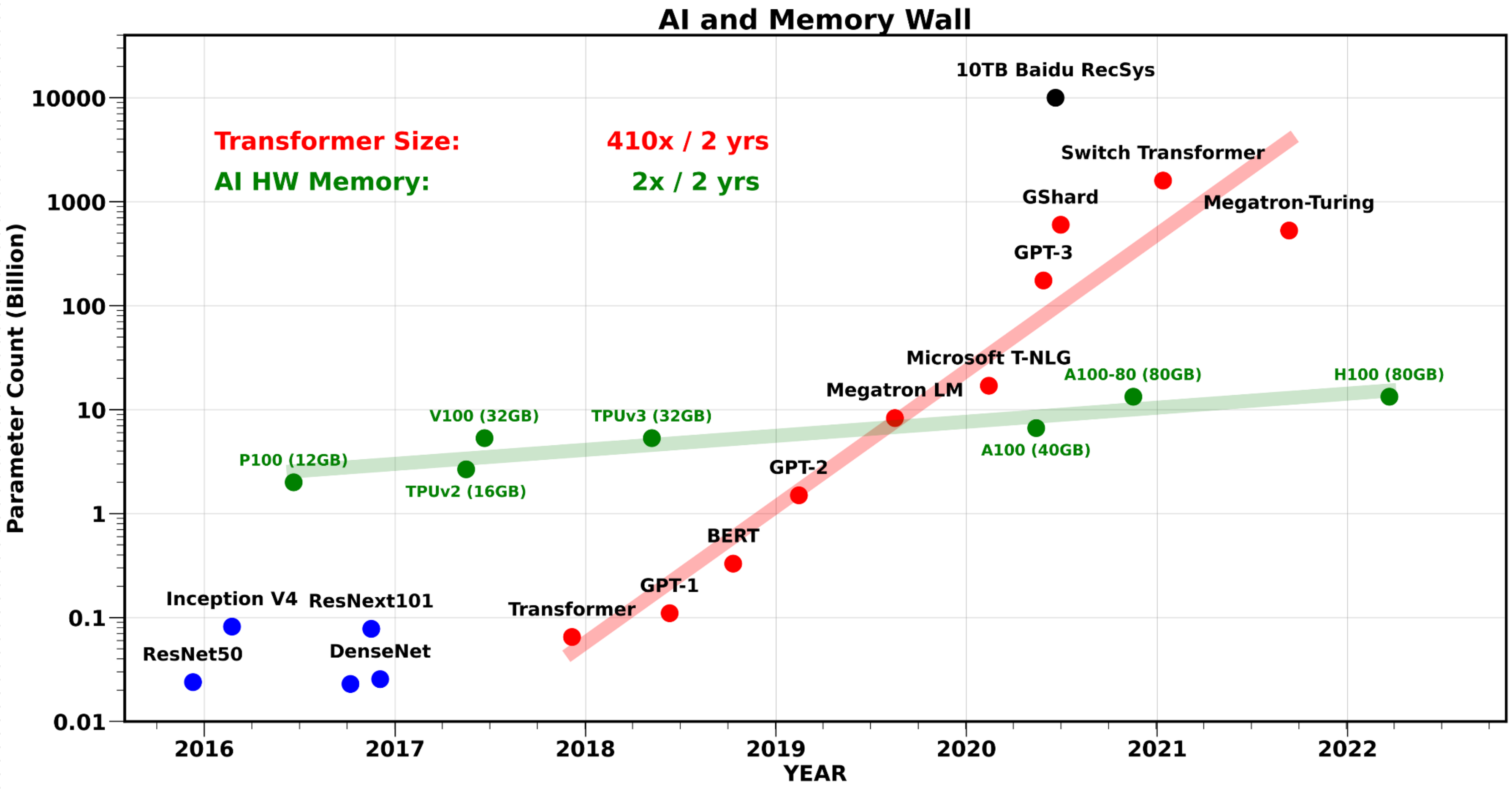
ARTIFACT EVALUATED usenix ASSOCIATION FUNCTIONAL

nnScaler: Constraint-Guided Parallelization Plan Generation for Deep Learning Training

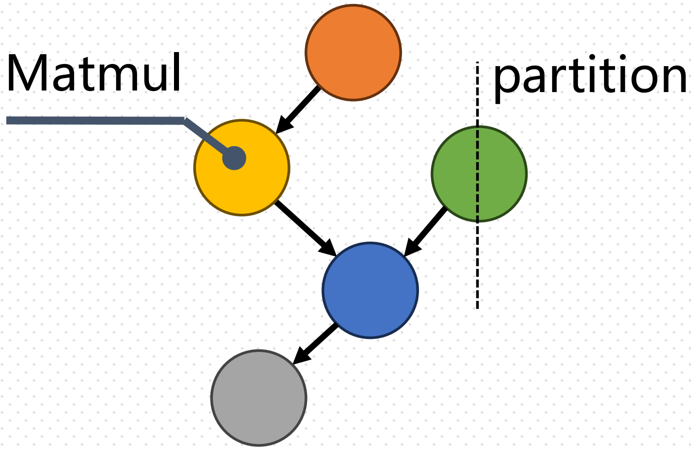
Zhiqi Lin[†], Youshan Miao[‡], Quanlu Zhang[‡], Fan Yang[‡], Yi Zhu[‡], Cheng Li[†], Saeed Maleki[◊], Xu Cao[‡], Ning Shang[‡], Yilei Yang[‡], Weijiang Xu[‡], Mao Yang[‡], Lintao Zhang[△], Lidong Zhou[‡]

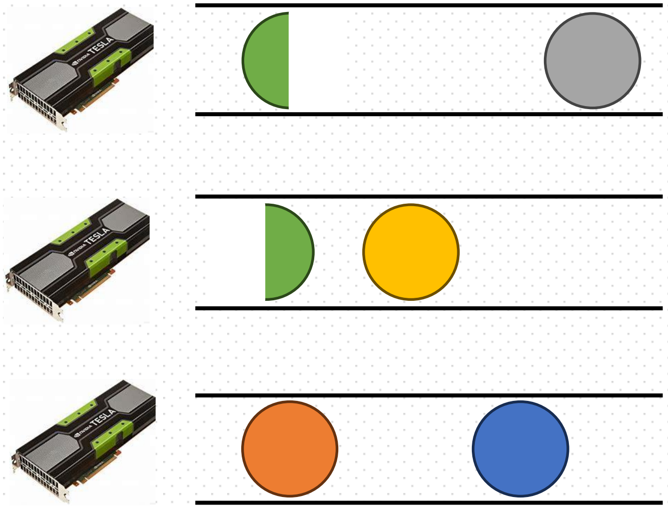
[†]University of Science and Technology of China,
[‡]Microsoft Research, [◊]xAI, [△]BaseBit Technologies

Large Models Need Parallelization



Parallelization Plans Matter





① **Graph partition**

- Nodes: operators
- Edges: tensors

② **Spatial-temporal schedule**

- Operator placement
- Determine execution order

Parallelization plans affect training performance greatly

Hard to Find Efficient Plans

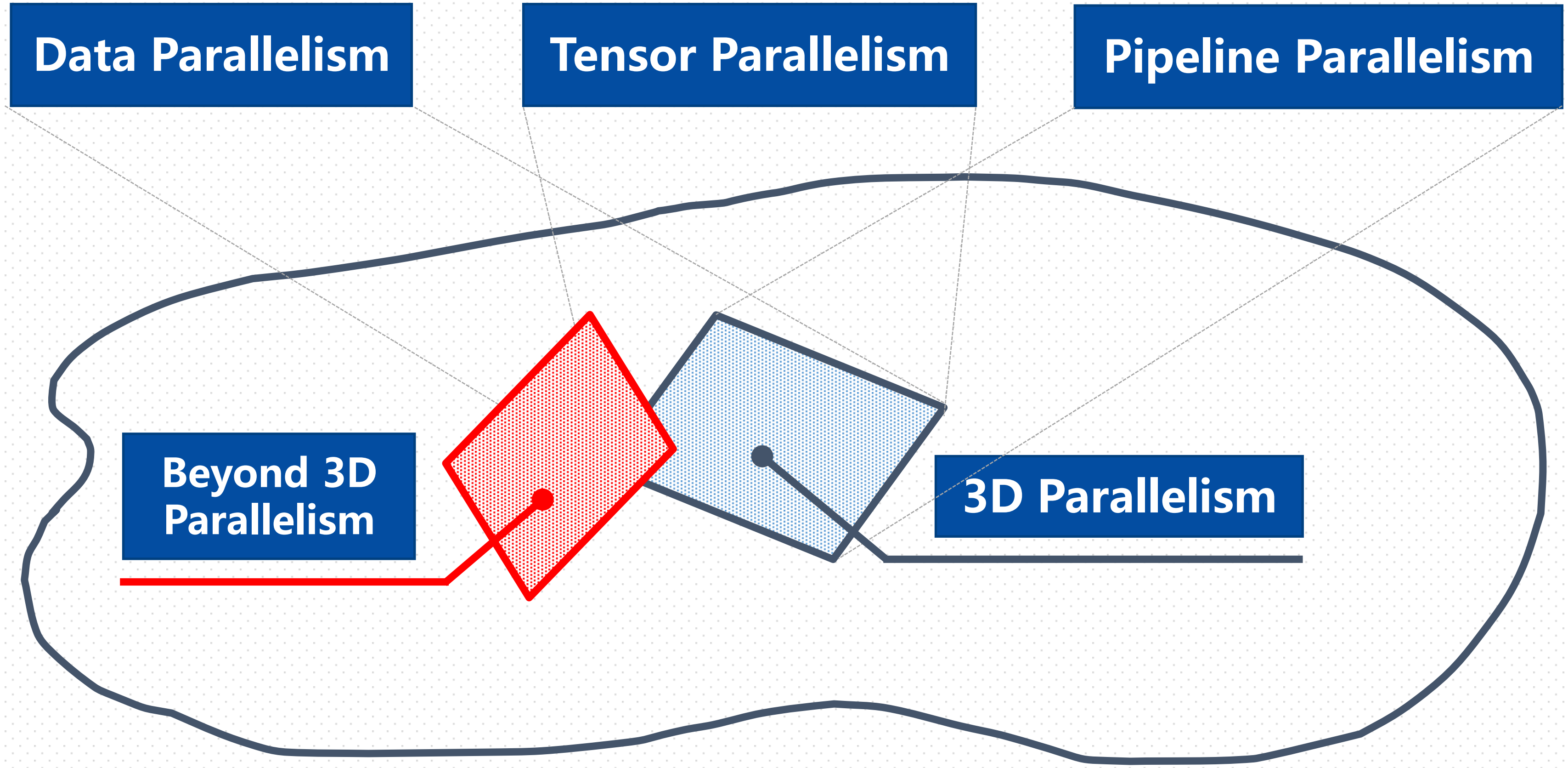
Given a model and GPUs

- ❑ Many operators
- ❑ Many partition choices
- ❑ Many placement choices
- ❑ Different execution orders for independent operators in one GPU

A combinatorial search space

Worse when model and the cluster become larger

Large Space vs. Efficient Plan?



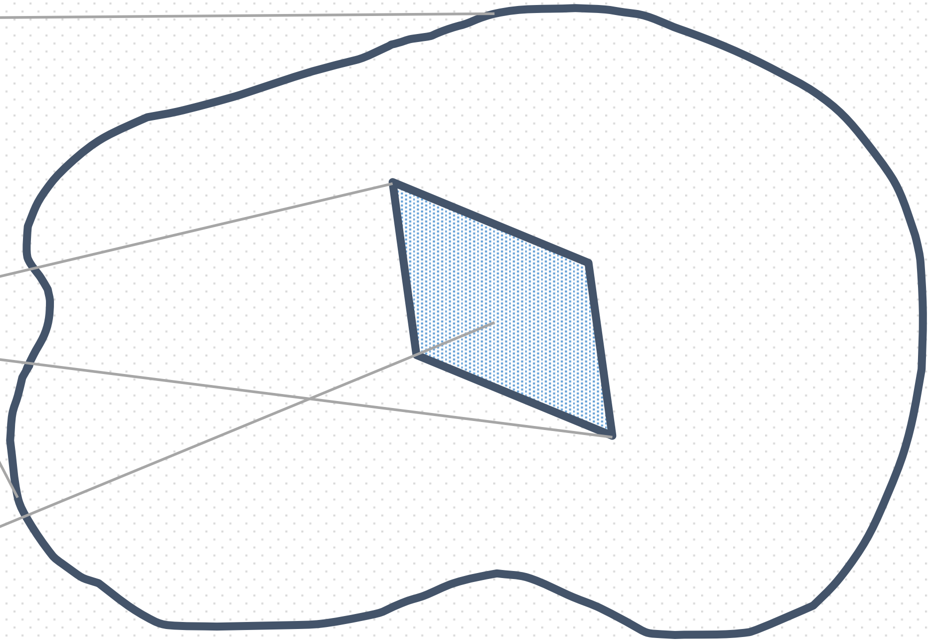
Current practice: limiting plan search within a well-studied search space

Solution Outline of nnScaler

Parallelization Space Primitives to Compose Generic Search Spaces

Space Construction Constraints to Reduce Search Space

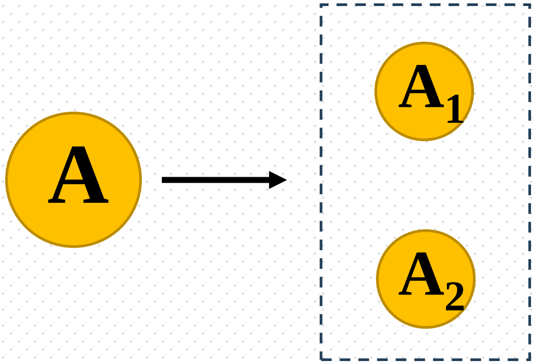
Conventional Search Algorithms Applicable to the Constrained Space

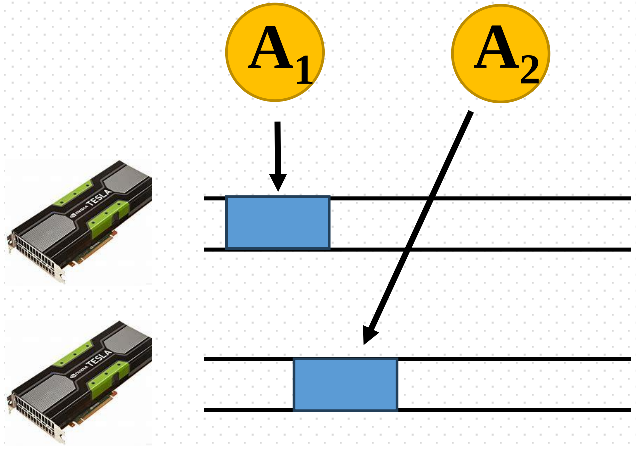


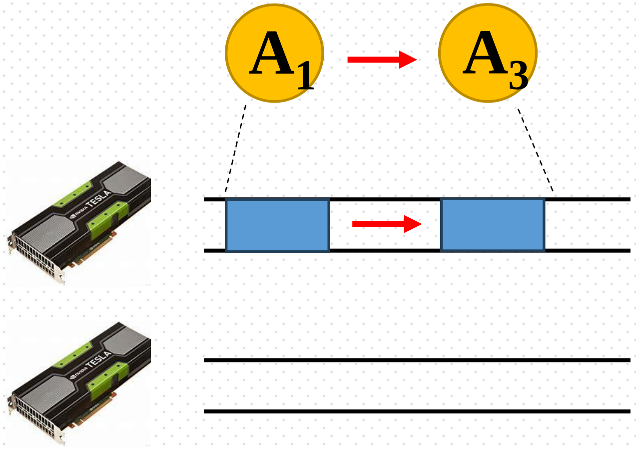
Parallelization Space

Empowering developers to find their own plans

Space Construction Primitives







Transformation

op-trans(op, algo, num)

Choices of operator partitioning schemes

Placement

op-assign(op, device)

Assign an operator to a device

Ordering

op-order(op1, op2)

Order independent operators in a device

Constraints in the Primitives

Transformation	Placement	Ordering
op-trans(op, $\boxed{\text{algo,}}$ $\boxed{\text{num}}$)	op-assign(op, $\boxed{\text{device}}$)	op-order($\boxed{\text{op1, op2}}$)

Constrain available partition algorithms

Constrain the number of sub-operators

Constrain the scope an operator can place

Constrain micro-batch ids between operators

Constraints => reduce the search space

Constraint as a Powerful Abstraction

Constraints: the insights of domain experts

Primitives	Constraints
(1) sub-ops = op-trans(op,algo,n)	$n = \|\mathbf{D}\|$
(2) op-assign(sub-op$_i$,d$_i$) (3) op-assign(sub-op$_j$,d$_j$)	$d_i, d_j \in \mathbf{D}$, $d_i \neq d_j$

Constraints of data/tensor parallelism

Primitives	Constraints
(1) op-assign(G$_i$, d$_i$) (2) op-assign(G$_j$, d$_j$)	$d_i, d_j \in \mathbf{D}$, $d_i \neq d_j$

Primitives	Constraints
(1) op-order((fG$_i$,m),(fG$_i$,n)) (2) op-order((bG$_i$,m),(bG$_i$,n))	$m < n$
(3) op-order((fG$_i$,m+$ofst$),(bG$_i$,m)) (4) op-order((bG$_i$,m),(fG$_i$,m+$ofst$+1))	$ofst = \|\mathbf{D}\| - i$, $m \geq 0$

Constraints of pipeline parallelism (1F1B)

Existing Parallelism as Constraints

Operators	Primitives	Constraints
op $\in$ {Attn $\cup$ FF}	sub_ops = op-trans(op,algo,n)	$n = C \cdot \|\mathbf{D}_i\|$
	op-assign(sub_op$_i^j$,d$_i$)	$0 \leq j < \|C\|$ $d_i \in \mathbf{D}_i$

CoShard

Primitives	Constraints
(1) op-order((f$_1$G$_i$,m+2),(f$_2$G$_i$,m+1))	$m \geq 0$
(2) op-order((f$_2$G$_i$,m+1),(f$_3$G$_i$,m))	$m > 0$
(3) op-order((f$_3$G$_i$,m),(bG$_i$,m-ofst))	$m > ofst$

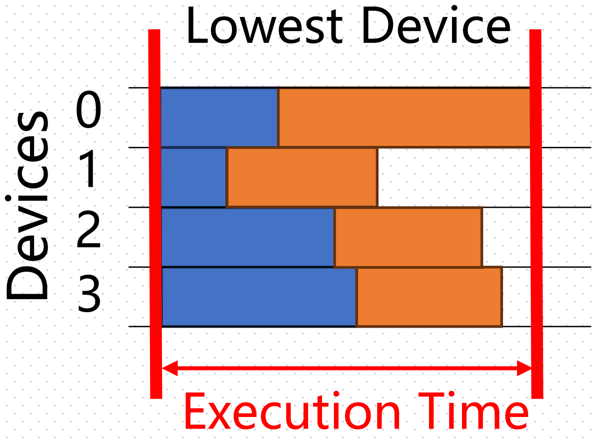
3F1B

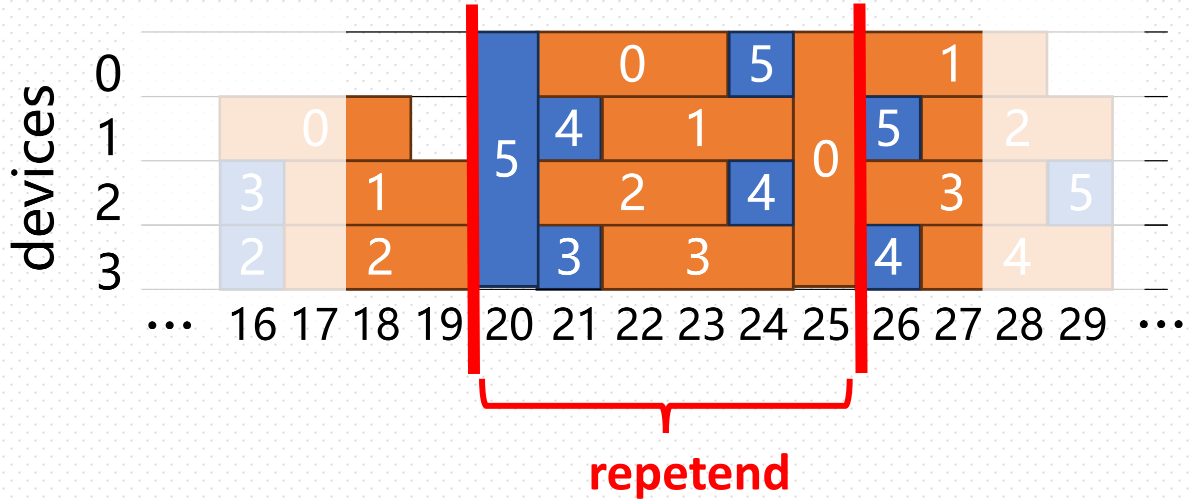
Operators	Primitives	Constraints
op $\in \mathbf{E}$	sub_ops = op-trans(op,algo,n) op-assign(sub_op$_i$,d$_i$)	$n = \|\mathbf{D}\|$ $d_i \in \mathbf{D}$ $0 \leq i < \|D\|$
ops $\notin \mathbf{E}$	staged_spmd(ops, $\mathbf{D}$)	

Interlaced Pipeline

New Constraints for Emerging Models

Plan Search Policy





$$\boldsymbol{minimize} \quad \max_{d \in D} \sum_{\mathrm{op} \in d_{op}} Comp_{op} + Comm_{op}$$

Transformation & Placement

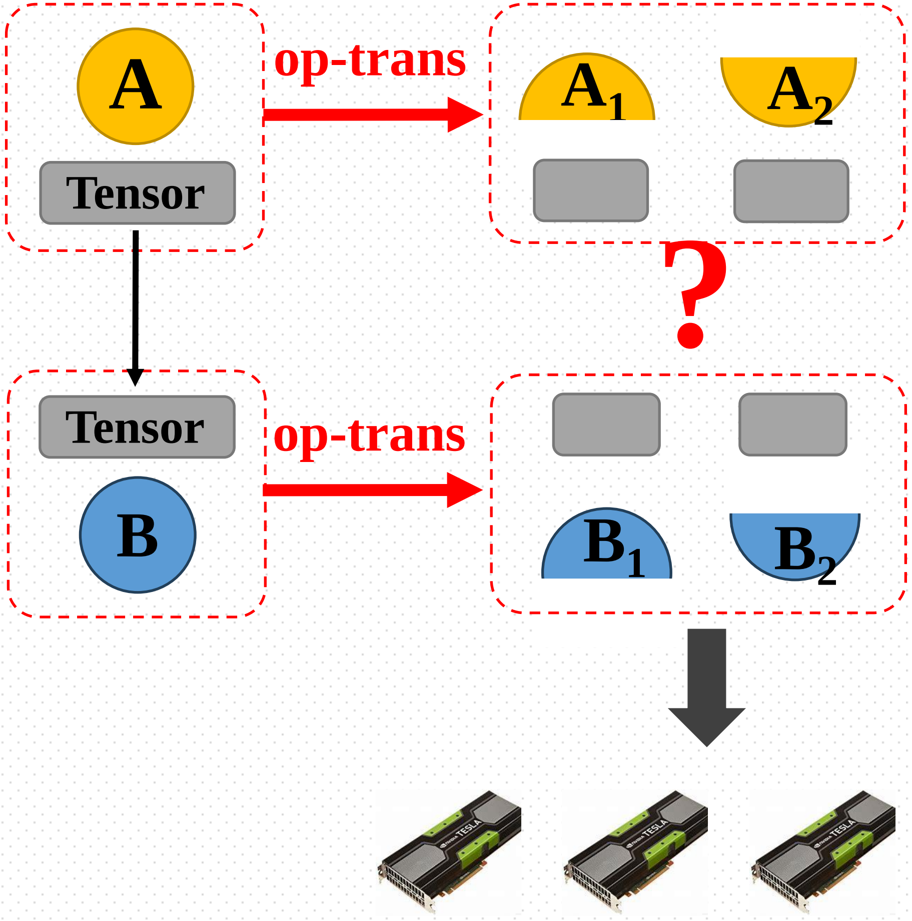
single micro-batch
minimize device execution time
<DP, ILP> search

→

Temporal Ordering

multiple micro-batches
maximize device utilization
<Tessel> search

Plan Materialization for Execution

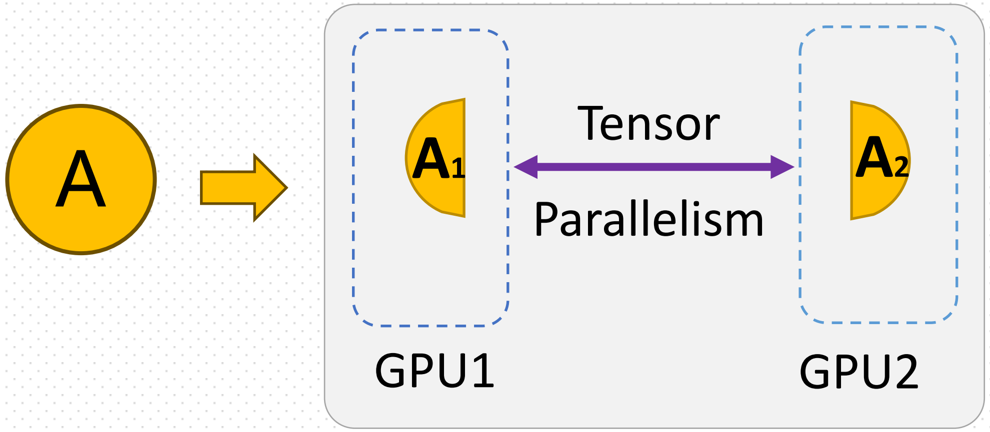


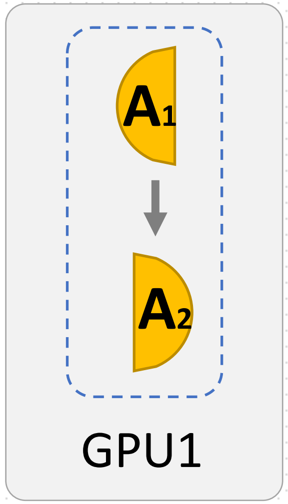
Additional Considerations

- **Tensor lineage** during transformation
- Efficient **communications** for equivalency
- Overall plan **correctness**
- Executable: PyTorch code

vTensor-pTensor abstraction (Section 6)

New Plan: CoShard





- **Reduced** peak memory usage
- **Lower** communication cost
- Beyond tensor parallelism

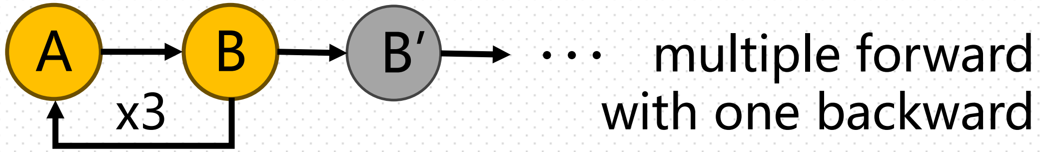
Tensor Parallelism
partitioned operators must be placed on **different** devices

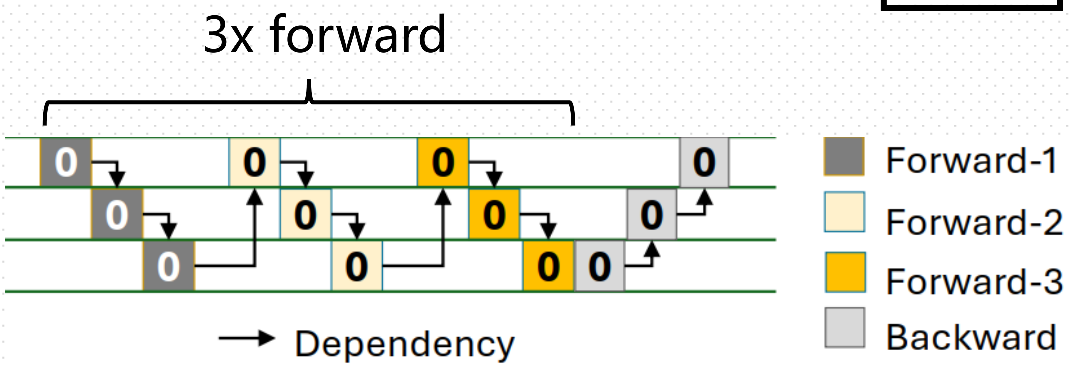
Coshard
partitioned operators can be **co-located** on a same device

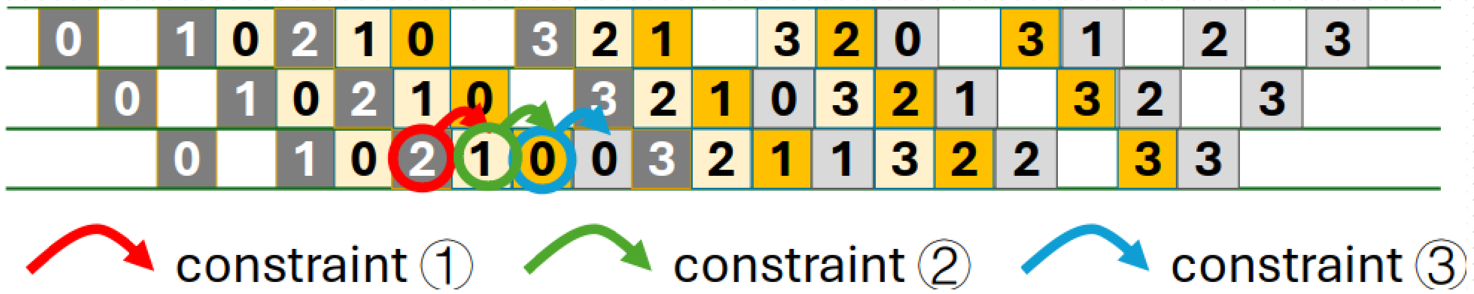
Operators	**Primitives**	**Constraints**
op $\in \{\text{Attn} \cup \text{FF}\}$	`sub_ops = op-trans(op,algo,n)`	$n = C \cdot \mid \mathbf{D}_i \mid$
	`op-assign(sub_op`$_{i}^{j}$`,d`$_i$`)`	$0 \leq j < \lvert C \rvert$ $d_i \in \mathbf{D}_i$

New Plan: 3F1B

- **AlphaFold: 3F1B Parallelization Plan**



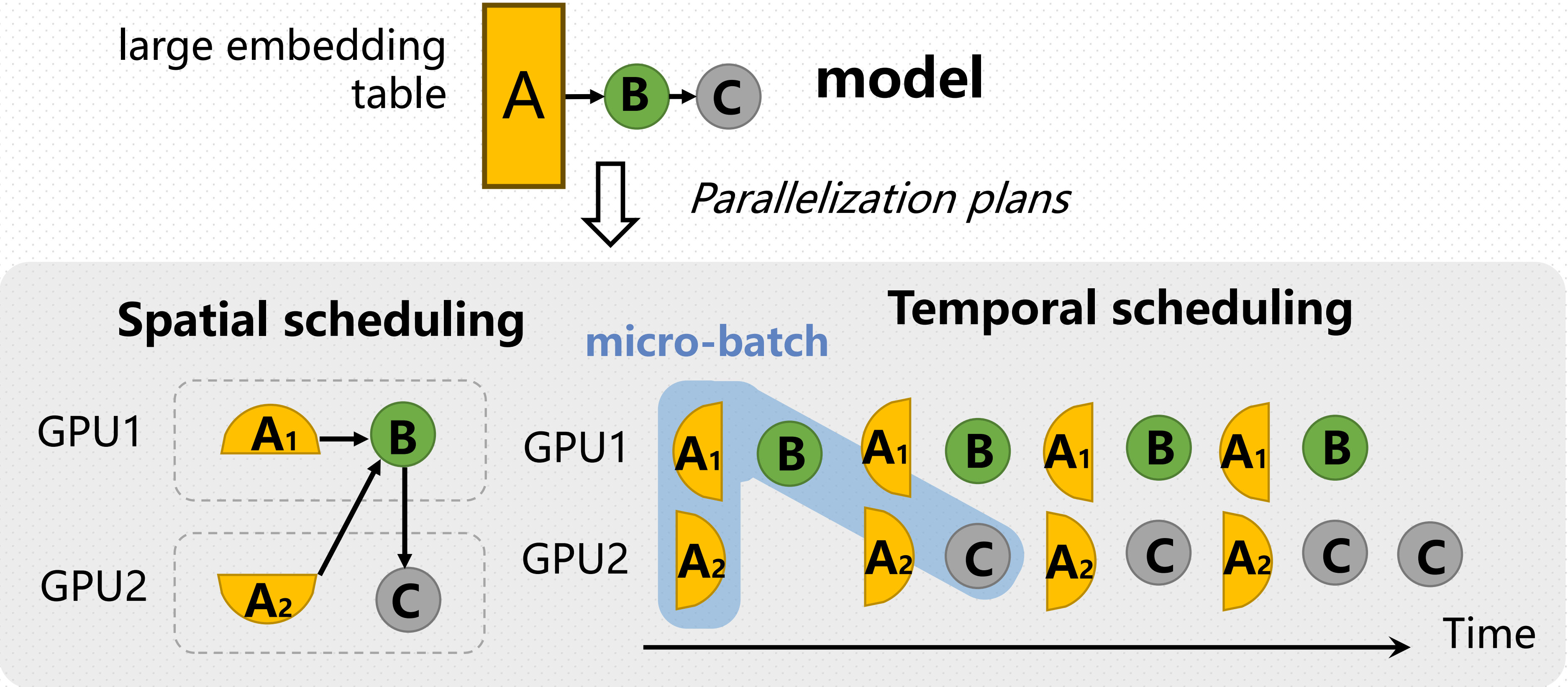




Primitives	Constraints
① op-order((f$_1$G$_i$,m+2),(f$_2$G$_i$,m+1))	$m \geq 0$
② op-order((f$_2$G$_i$,m+1),(f$_3$G$_i$,m))	$m > 0$
③ op-order((f$_3$G$_i$,m),(bG$_i$,m-ofst))	$m > ofst$

New Plan: Interlaced Pipeline

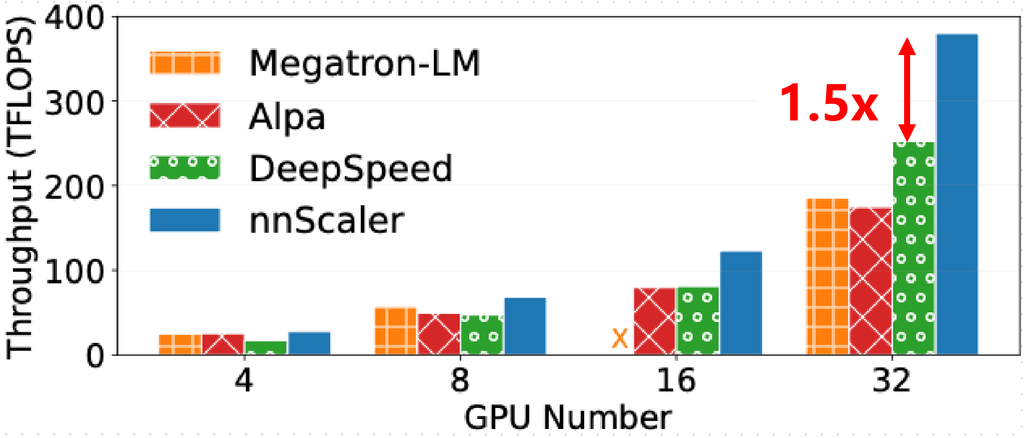
- **T5：Interlaced pipeline**



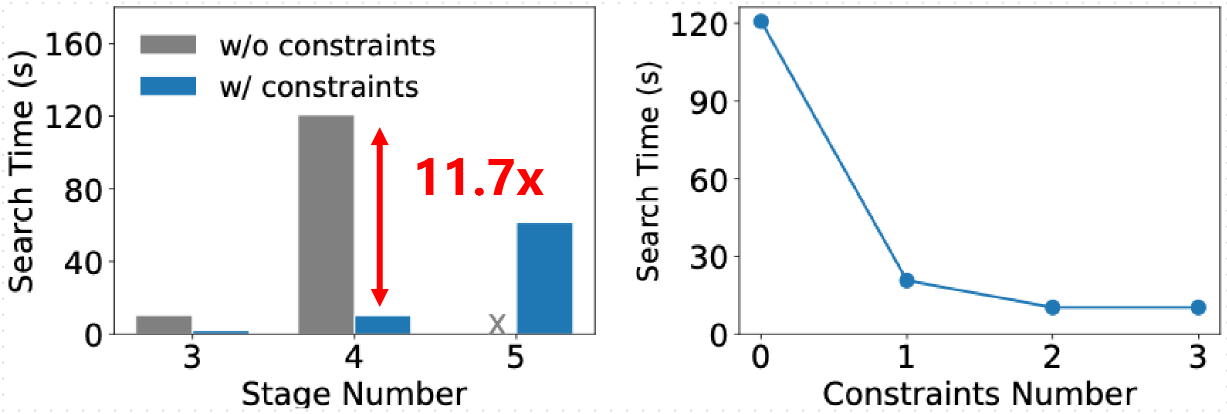
full-device tensor parallelism for embedding

Operators	Primitives	Constraints
op $\in \mathbf{E}$	sub_ops = op-trans(op,algo,n) op-assign(sub_op$_i$,d$_i$)	$n = \lvert\mathbf{D}\rvert$ $d_i \in \mathbf{D}$ $0 \le i < \lvert D\rvert$
ops $\notin \mathbf{E}$	staged_spmd(ops, $\mathbf{D}$)	

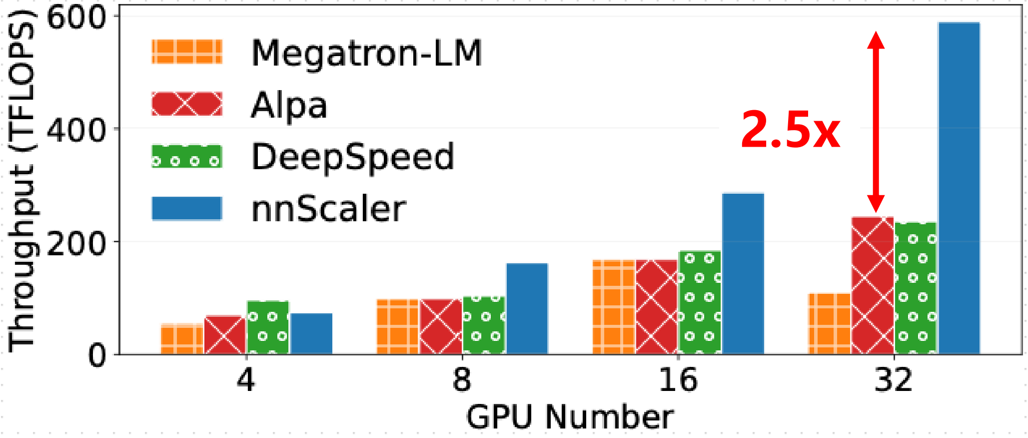
Evaluation: Performance



Swin-Transformer Model



AlphaFold 2 Search Time

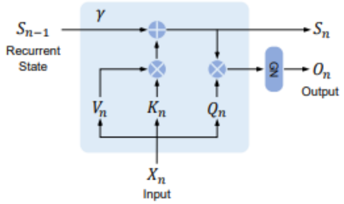


T5 Model

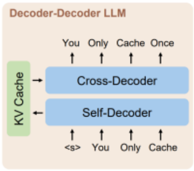
- DGX-2 32x V100-32GB
- Training Throughput Improvement：
  - **1.5-2.5x**
- Search Speed Improvement with Constraints：
  - **11.7x**

nnScaler in Practice

PreTrain and PostTrain (Finetune)



RetNet



YOCO

Phi-3

Graphormer

Llama2

...

NVIDIA

DGX-2, A100, H100

AMD

MI-200

Deployment

512x GPUs（NV, AMD）
92B large model

Summary

The primitives and constraints:

abstractions for training flexibility and efficiency

nnScaler:

a powerful tool to facilitate model training

ARTIFACT EVALUATED usenix ASSOCIATION AVAILABLE

ARTIFACT EVALUATED usenix ASSOCIATION FUNCTIONAL

nnScaler: Constraint-Guided Parallelization Plan Generation for Deep Learning Training

Thank you for listening
Q & A