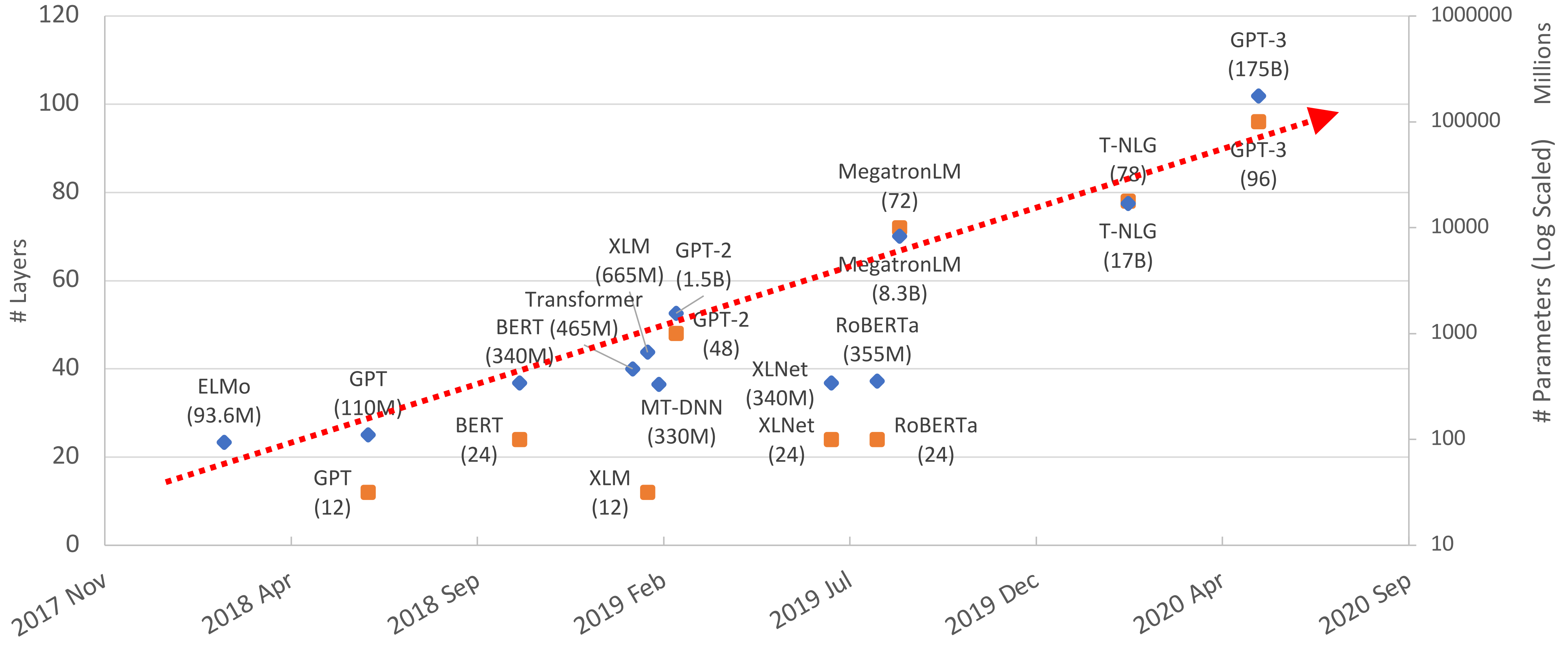
Baechi: Fast Device Placement of Machine Learning Graphs

Beomyeol Jeon[†], Linda Cai[*], Pallavi Srivastava[◊], Jintao Jiang[‡], Xiaolan Ke[†],
Yitao Meng[†], Cong Xie[†], Indranil Gupta[†]

SoCC ’20

[†]University of Illinois at Urbana-Champaign, [*]Princeton University, [◊]Microsoft, [‡]UCLA

Increasing Machine Learning (ML) Model Size

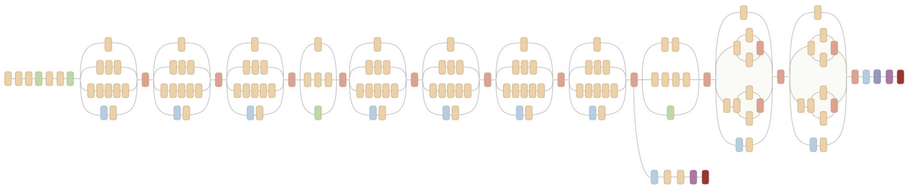


Not Enough Memory

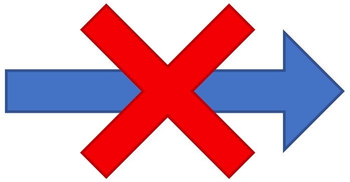
- GPUs used in AWS, Google Cloud, and Azure

GPU	P4	M60	K80	P100	T4	V100
Memory	8 GB	8 GB	12 GB	12/16 GB	16 GB	16/32 GB

- Even 32GB GPU **insufficient** for > 1.3 B parameters [1]



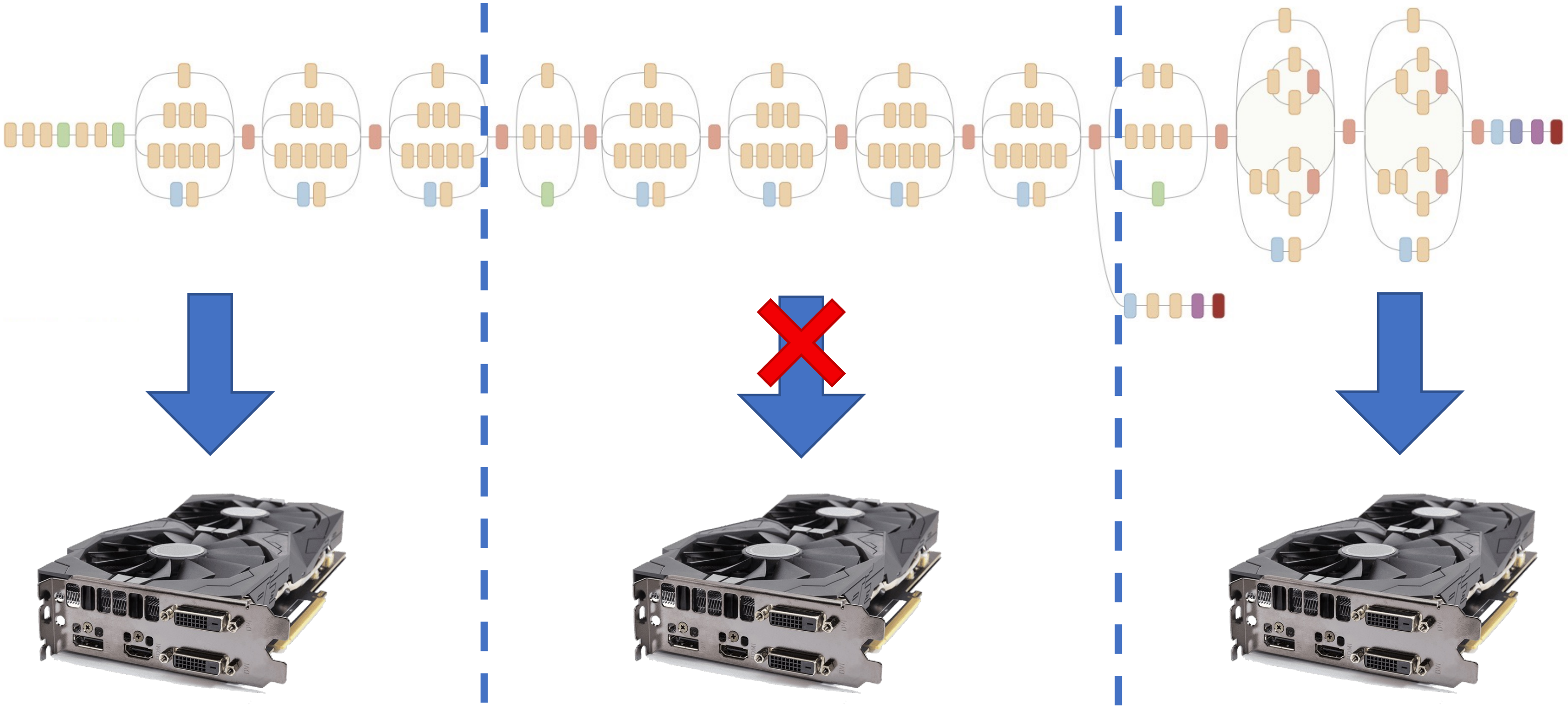
ML Model Graph



- ML training on *memory-constrained* devices
  - Smartphones, UAVs, drones, etc.

[1] https://www.microsoft.com/en-us/research/blog/turing-nlg-a-17-billion-parameter-language-model-by-microsoft/

Multi-GPU Training: Model Parallelism



How to ***place*** ML operators on devices?

Why Does Device Placement Matter?

- ML Training repeats training *steps* of updating parameters

- *Step time*: Elapsed time for a single training step of the placed ML model

- *Bad* placement $\Longrightarrow$ Step time ↑ (communication overhead ↑, no parallelism)
- *Slow* placement time $\Longrightarrow$ Entire training time ↑ (placement + training)

- **Goal:** Place a ML model ***fast*** (low placement time) and ***well*** (low step time)

Prior Work

- *Expert-designed* Approach
  - E.g. Google Neural Machine Translation (GNMT) [2]
  - Require **domain knowledge** and **significant manual efforts**

- *Learning-based* Approaches
  - Reinforcement learning (RL)
  - E.g., ColocRL [3], HierarchicalRL [4], Placeto [5]
  - Require **very long** time to place ML models (2 hours ~ 3 days)
  - Require **re-training** on different ML models and varying environment

[2] Wu et al. Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. arXiv:1609.08144.
[3] Mirhoseini et al. Device Placement Optimization with Reinforcement Learning. ICML '17.
[4] Mirhoseini et al. Hierarchical Planning for Device Placement, ICLR ’18.
[5] Addanki et al. Learning Generalizable Device Placement Algorithms for Distributed Machine Learning. NeurIPS '19.

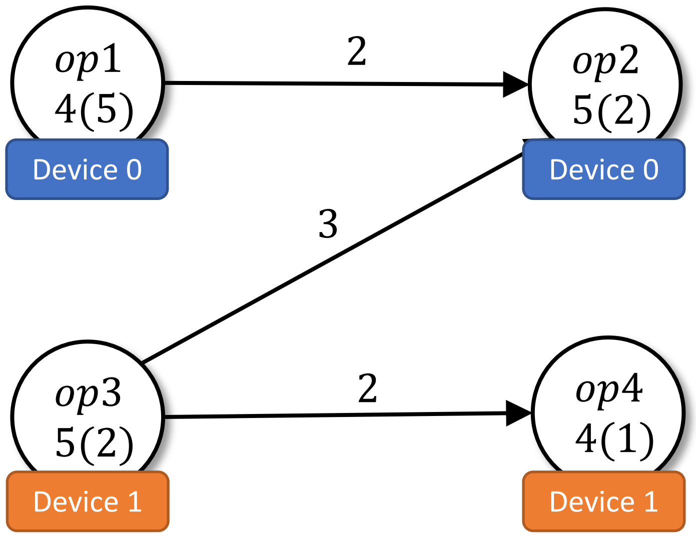
Baechi

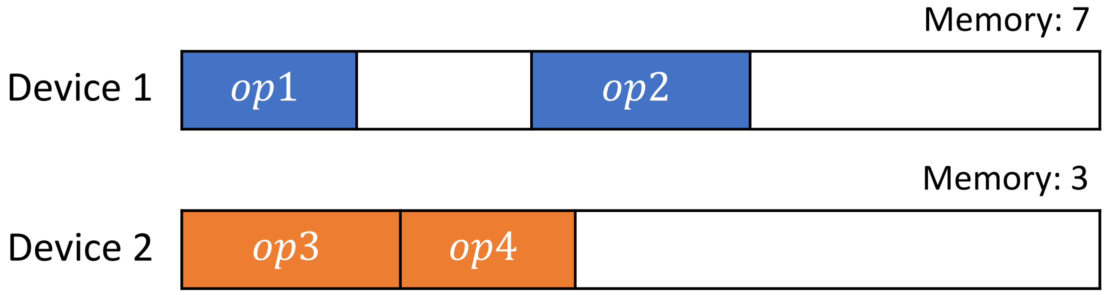
- ML placement system that incorporates ***algorithmic*** approaches into TensorFlow
- Our contributions
  - Placement algorithms for *memory-constrained* environments
    - Memory-constrained Earliest Task First (*m-ETF*)
    - Memory-constrained Small Communication Time (*m-SCT*)
      - Provably *within* a *constant factor* of the optimal execution time*
    - Memory-constrained Topological Sort (*m-TOPO*) [strawman]
  - Optimizations
    - Co-adjust Placement, Co-placement, Operator Fusion, Sequential Communication Support
- **Place quickly: 654–206K×** faster placement time than learning-based approaches
  - Place ML models on 4 GPUs within *only* **1.2** seconds
- **Place well**: *only* up to **6.2%** higher step time than expert's placements

* Conditions apply

Algorithm 1: m-ETF

- Earliest Task First (ETF) [6]
  - Schedule an operator with *earliest schedulable time* on its corresponding device *first*
  - *Infinite* memory assumed



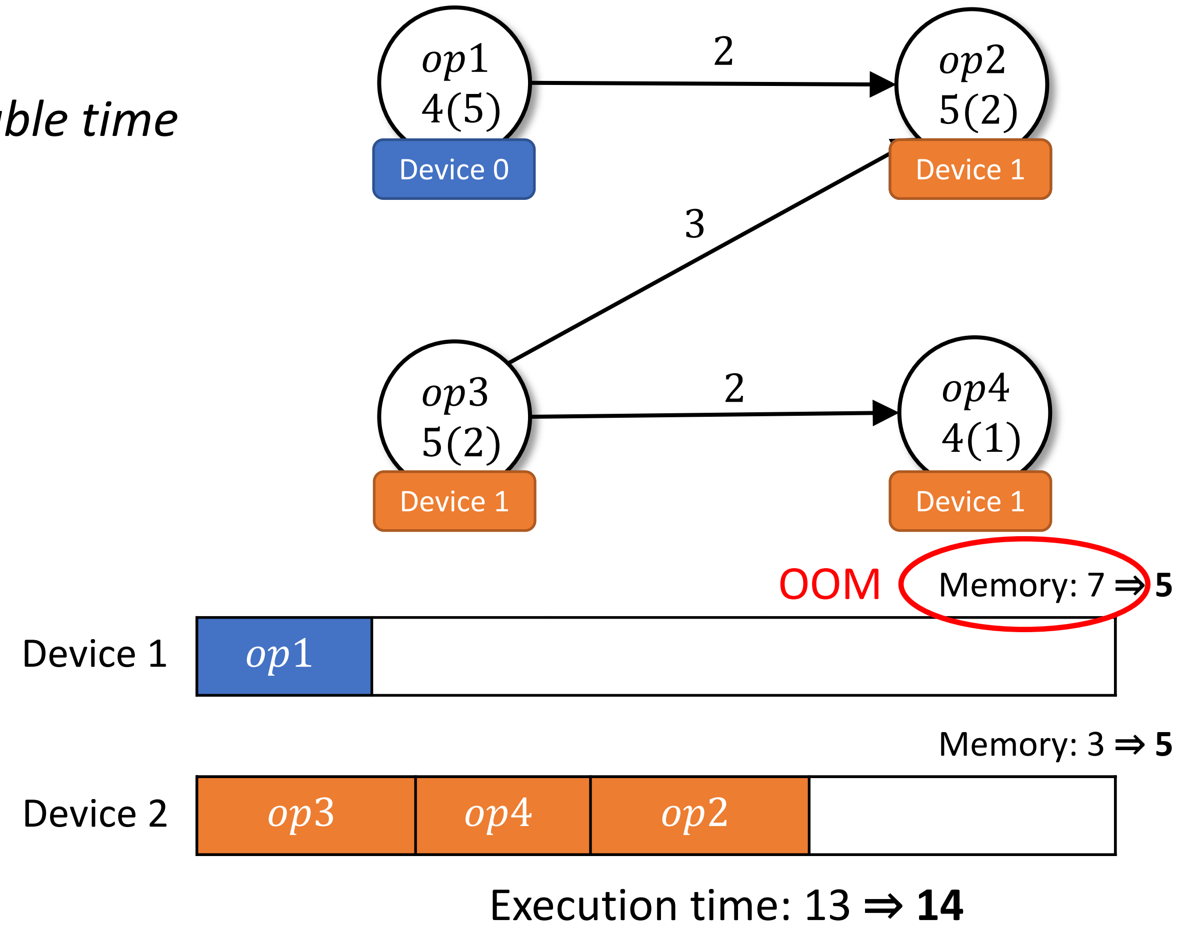


Execution time: 13

[6] Hwang et al. Scheduling Precedence Graphs in Systems with Interprocessor Communication Times. SIAM Journal on Computing, 18(2)

Algorithm 1: m-ETF

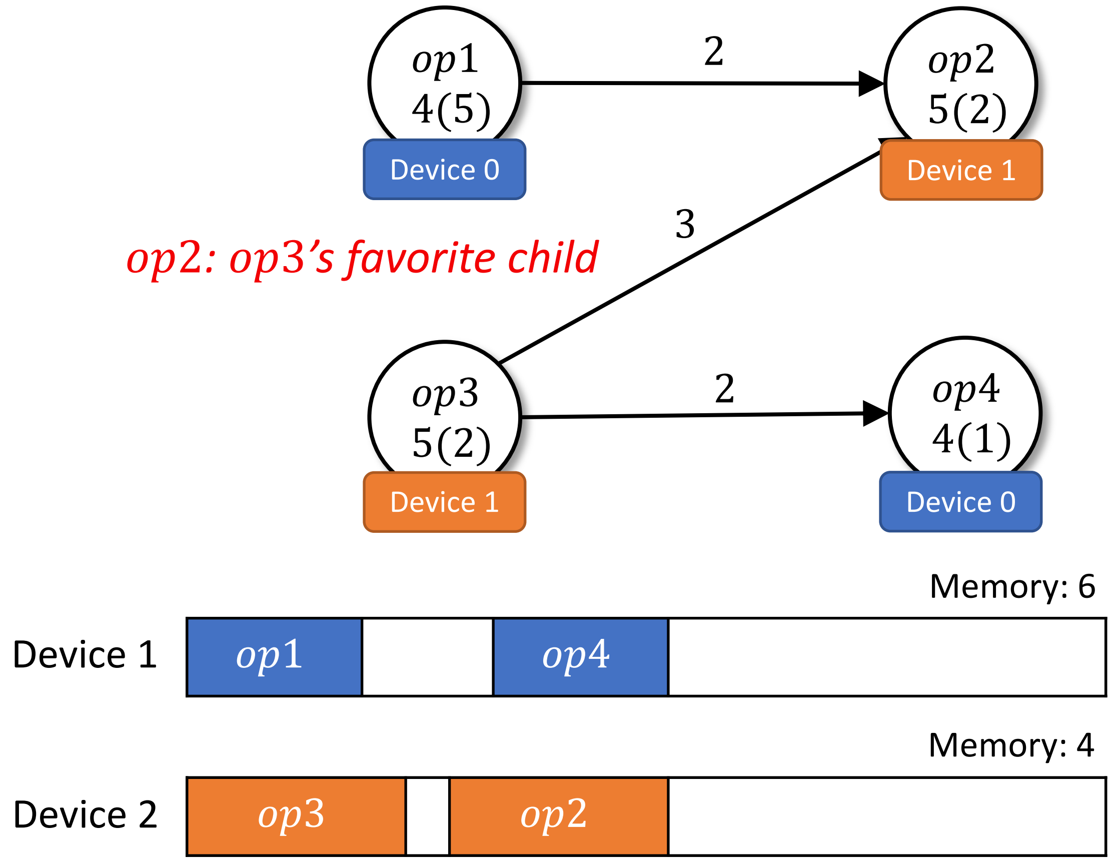
- Earliest Task First (ETF) [6]
  - Schedule an operator with *earliest schedulable time* on its corresponding device *first*
  - *Infinite* memory assumed
- **Our modified version:** ***m-ETF***
  - What if device memory limit is **5**?
  - *Exclude* devices with *insufficient* memory from placement



[6] Hwang et al. Scheduling Precedence Graphs in Systems with Interprocessor Communication Times. SIAM Journal on Computing, 18(2)

Algorithm 2: m-SCT

- Small Communication Time (SCT) [7]
  - Find operator's *favorite child* that is scheduled on the *same* device via ILP



Execution time: 11

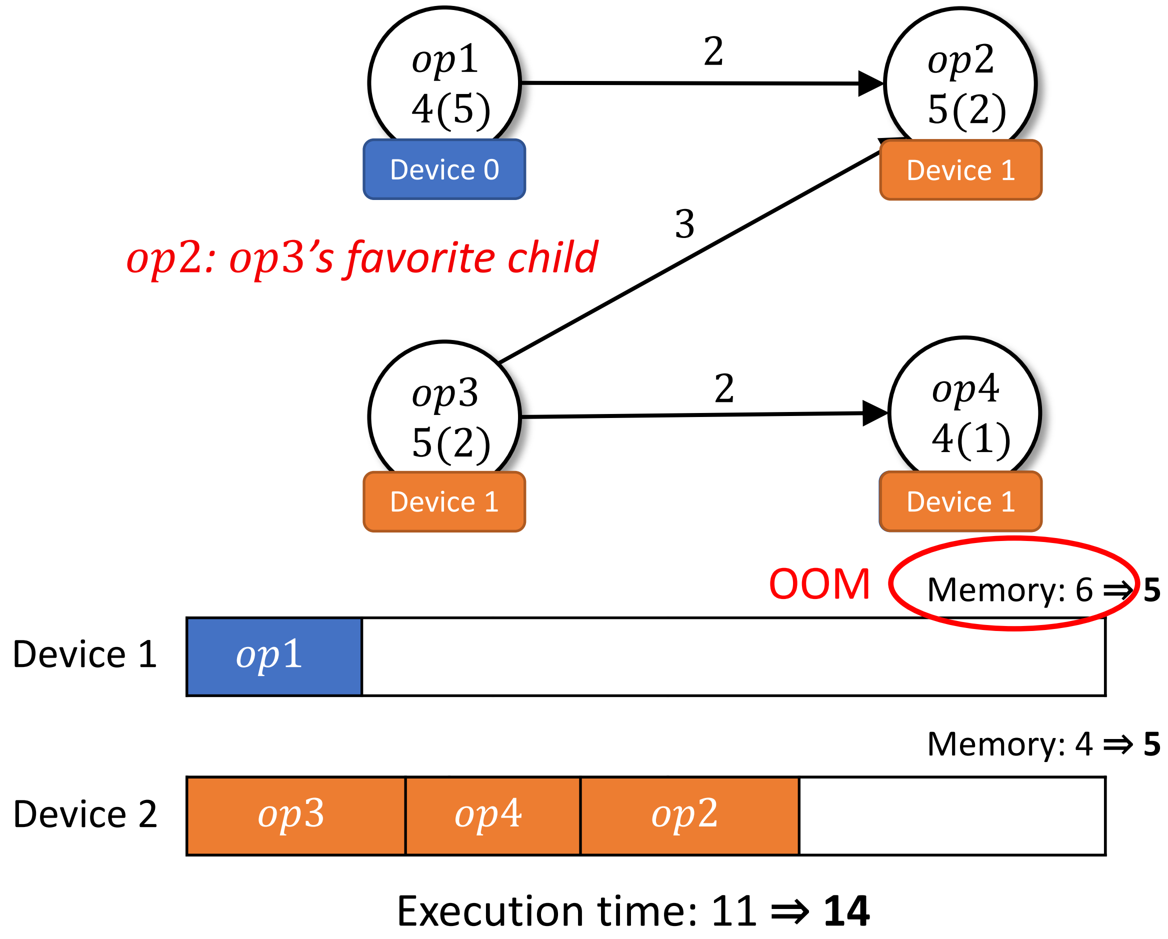
[7] Hanen and Munier. An Approximation Algorithm for Scheduling Dependent Tasks on m Processors with Small Communication Delays. ETFA '95

* Conditions apply

Algorithm 2: m-SCT

- Small Communication Time (SCT) [7]
  - Find operator's *favorite child* that is scheduled on the *same* device via ILP
- **Our modified version:** *m-SCT*
  - Determine favorite child via *relaxed ILP*
  - Each device memory limit is **5**

Theorem 1. *m-SCT's execution time has a* ***constant*** *approximation ratio with respect to the optimal execution time*.*



[7] Hanen and Munier. An Approximation Algorithm for Scheduling Dependent Tasks on m Processors with Small Communication Delays. ETFA '95

* Conditions apply

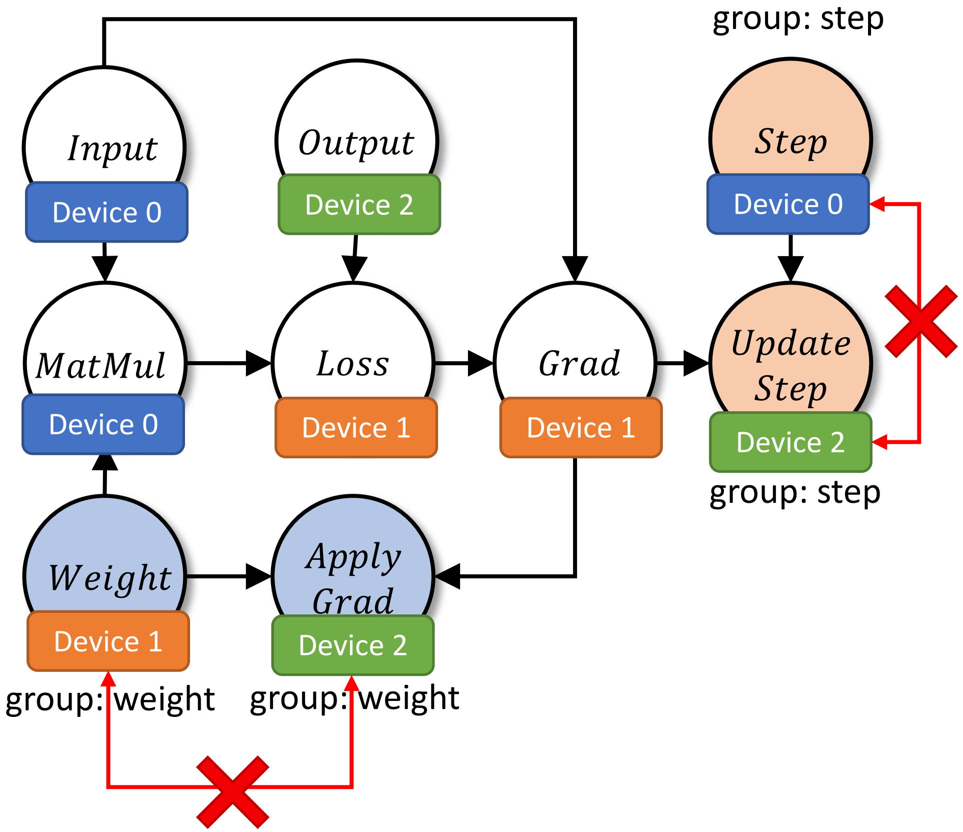
Do the Algorithms Work for TensorFlow?

- Generated placement results were **infeasible**
- Performance was **awful**

- Challenges
  1) TensorFlow colocation constraints
  2) Excessive communication overheads
  3) Massive number of operators
  4) Different network architectures: parallel vs. sequential

Challenges #1: TensorFlow Colocation Constraints

- TensorFlow *requires* some operators to be *colocated*

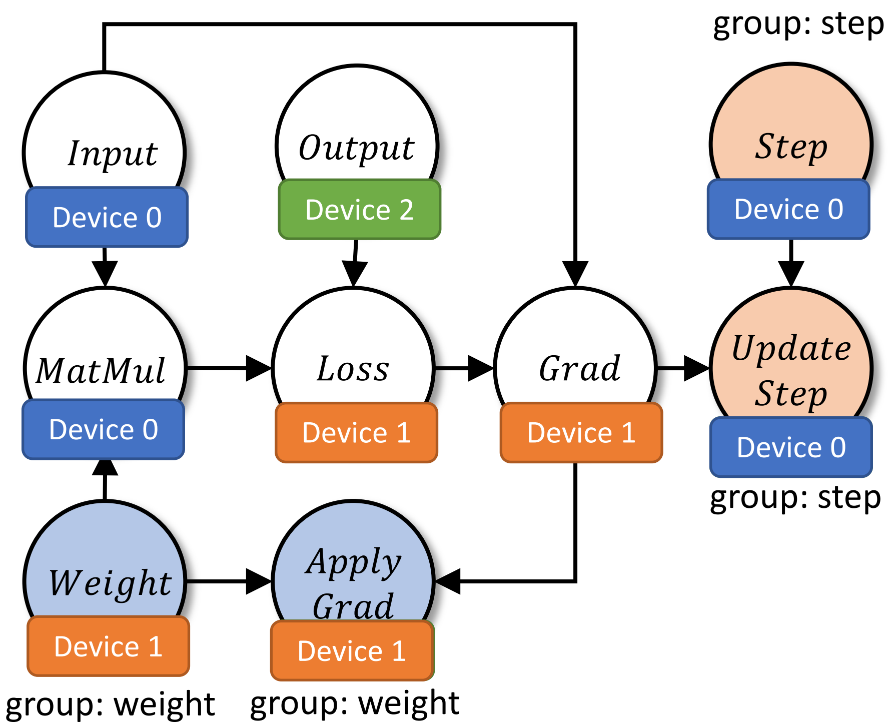


Challenges #1: TensorFlow Colocation Constraints

- TensorFlow *requires* some operators to be *colocated*

⇨ Tried *post-adjust placement*

- Fix *colocation-unaware* placement to satisfy the colocation constraints
  - Compute-dominant, memory-dominant, majority
- **Inconsistent** performance gain



Challenges #1: TensorFlow Colocation Constraints

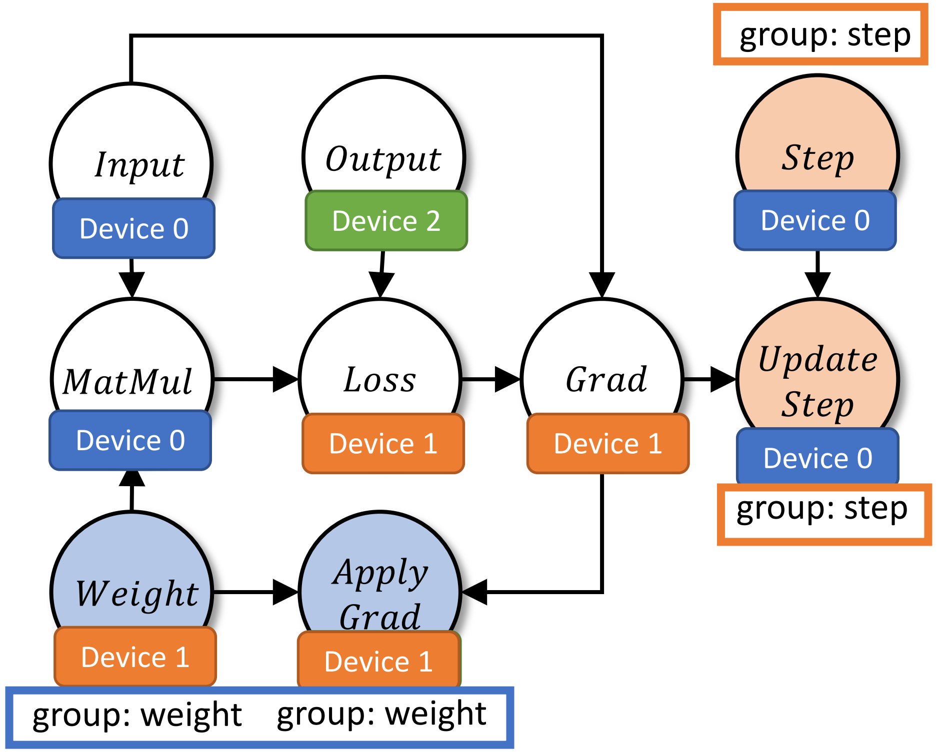
- TensorFlow *requires* some operators to be *colocated*

⇨ Tried *post-adjust placement*

- Fix *colocation-unaware* placement to satisfy the colocation constraints
  - Compute-dominant, memory-dominant, majority
- **Inconsistent** performance gain

⇨ *Co-adjust placement*

- Consider colocations *while* creating schedule
- 1st operator in a group placed ⇒ other ops in the group placed on the same device

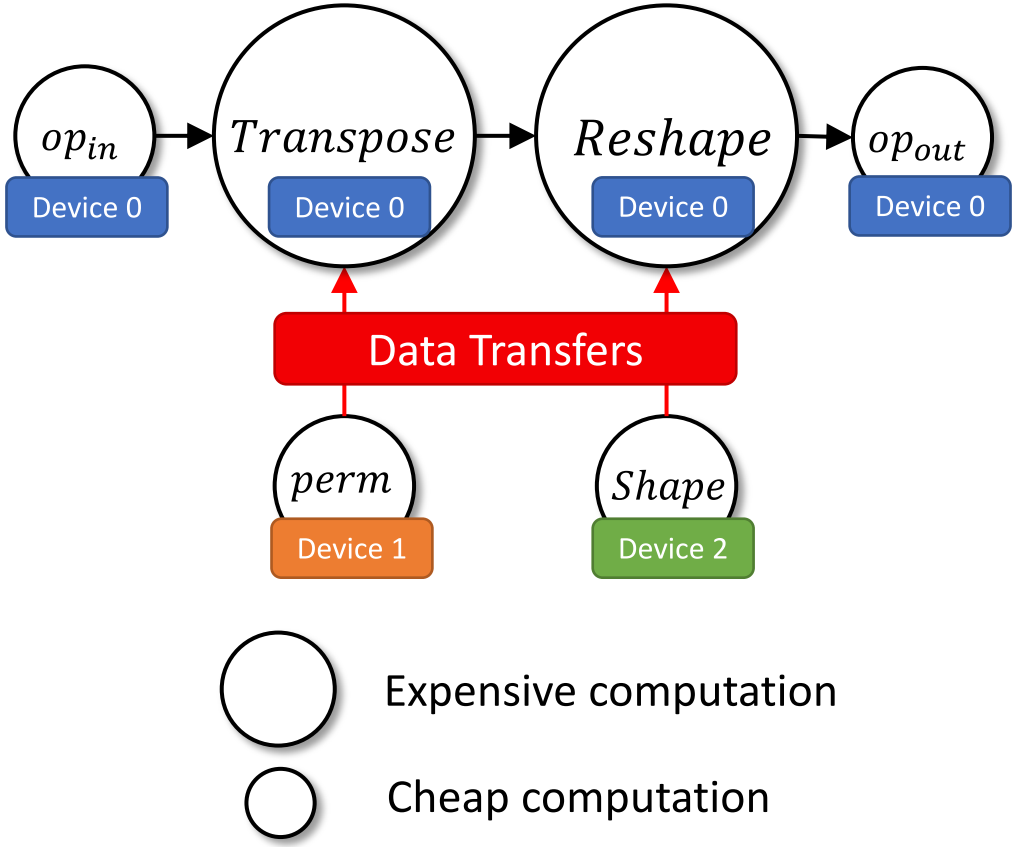


Challenge #2: Communication Blowup

- Splitting an ML model graph

  ⇒ Communication ↑

  ⇒ Step time ↑



Challenge #2: Communication Blowup

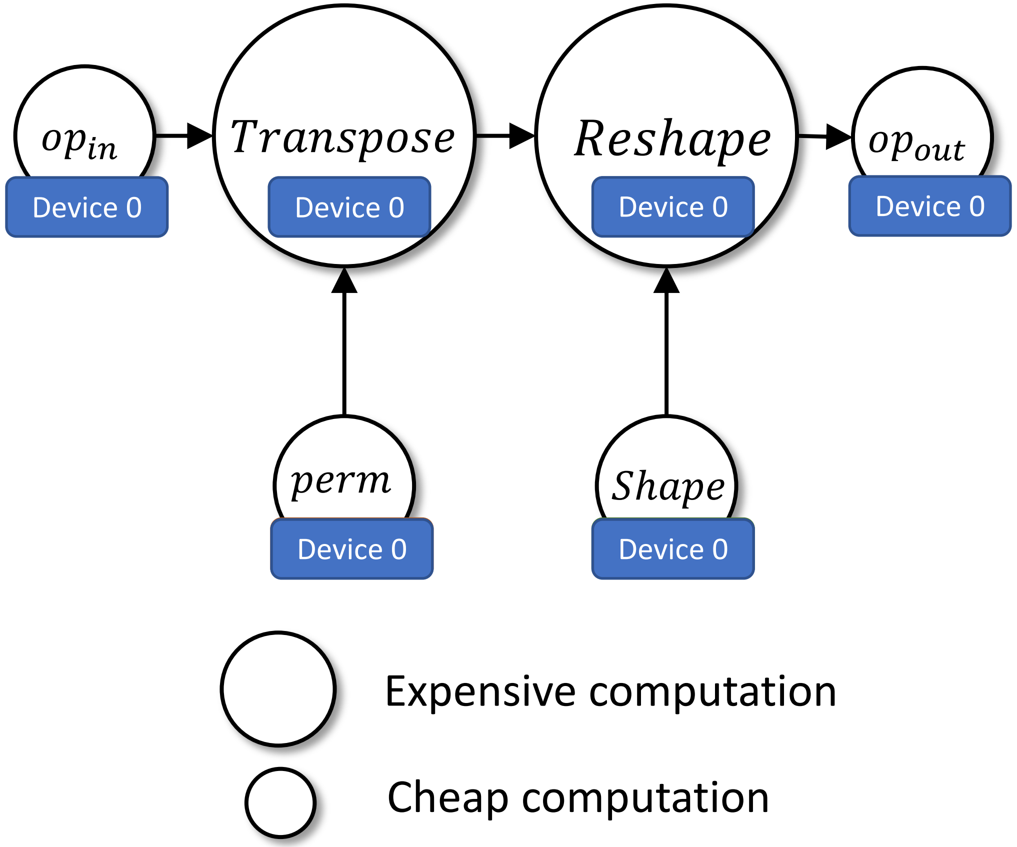
- Splitting an ML model graph

  ⇒ Communication ↑

  ⇒ Step time ↑

⇨ Operator *Co-placement*

- Operator's output is *only* used by its successor
  ⇒ Place them *together*
- Place respectively-matched forward and backward operators *together*

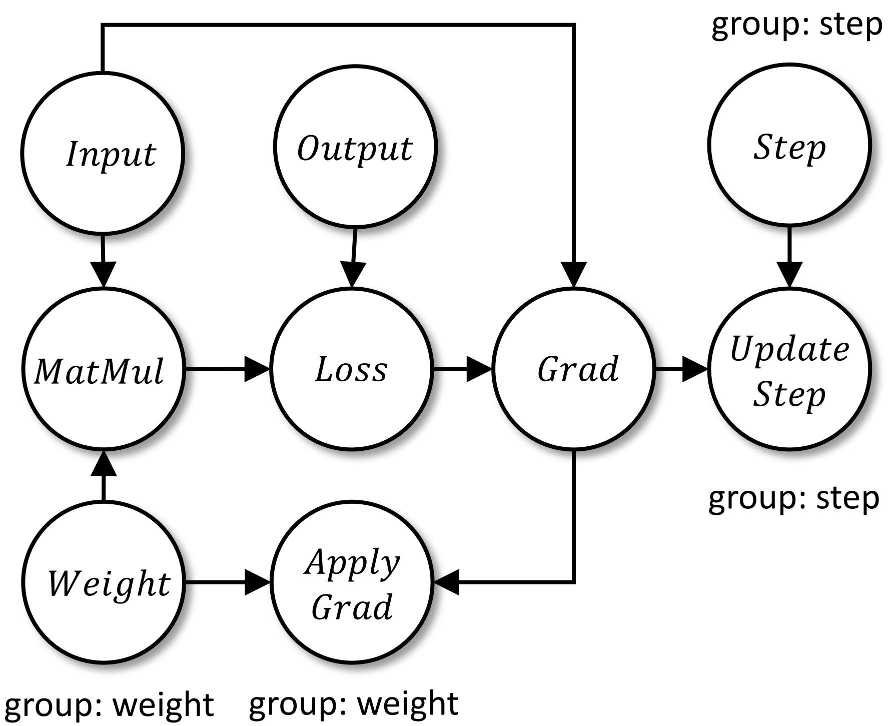


Challenge #3: Massive Number of Operators

- Number of operators $\uparrow \Longrightarrow$ Placement time $\uparrow$
- E.g., 4-layer GNMT
  - 22,340 operators $\Longrightarrow$ 7-minute placement time

⇨ *Operator Fusion*

- **Fuse** operators that are *directly connected* and *in the same co-placement group*

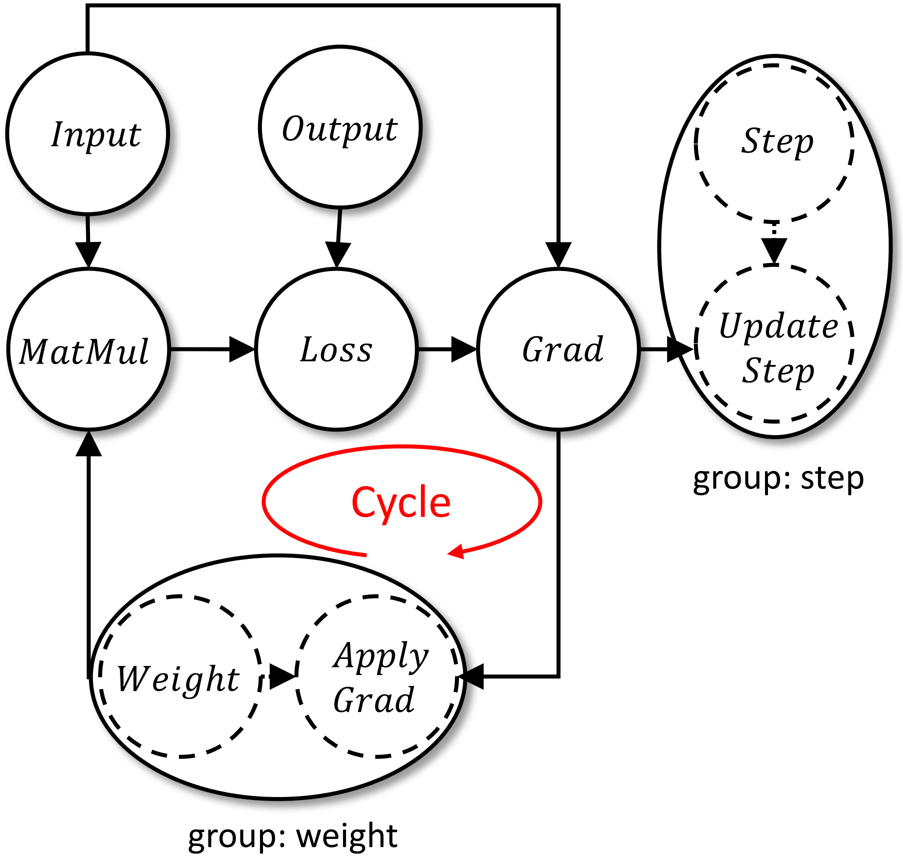


Challenge #3: Massive Number of Operators

- Number of operators ↑ ⟹ Placement time ↑

- E.g., 4-layer GNMT
  - 22,340 operators ⟹ 7-minute placement time

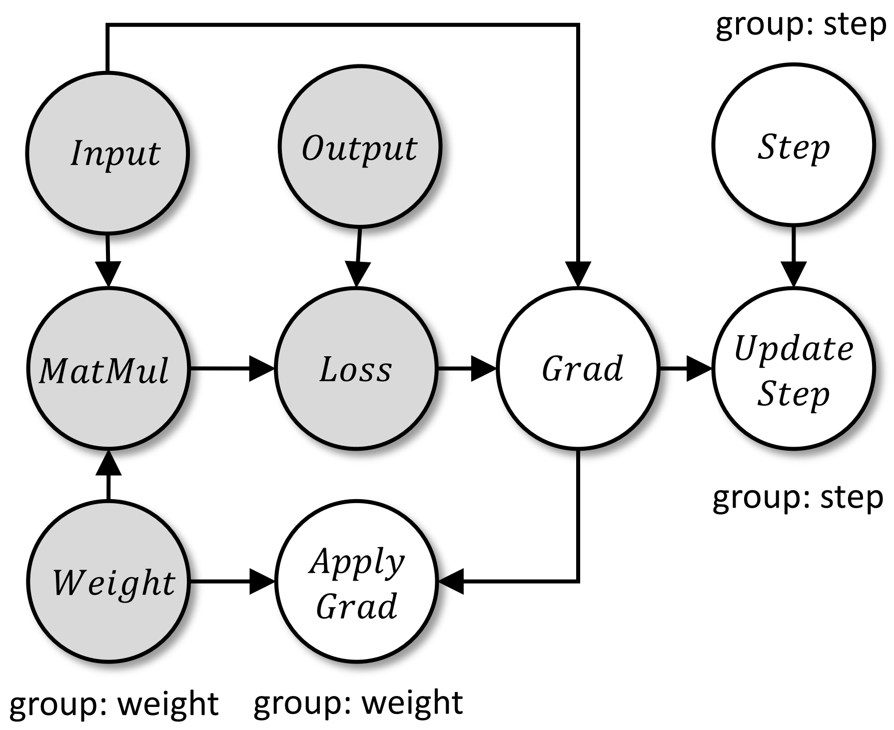
⇨ *Operator Fusion*
  - **Fuse** operators that are *directly connected* and *in the same co-placement group*
  - May introduce cycles
    - Checking **all** cycles – Expensive, Not scalable
    - ***Conservative*** but ***scalable*** heuristic
  - Minimize step time



Challenge #3: Massive Number of Operators

⇨ *Forward-Operator-based Placement*

- Place ops by *only* considering forward ops
  - Place backward ops as their corresponding forward ops on the same device
- With sufficient memory*



- 4-layer GNMT
  - # operators: 22,340 ⟹ **706**
  - Placement time: 7 minutes ⟹ **1.2 seconds**

* each GPU memory ≥ model graph memory requirement

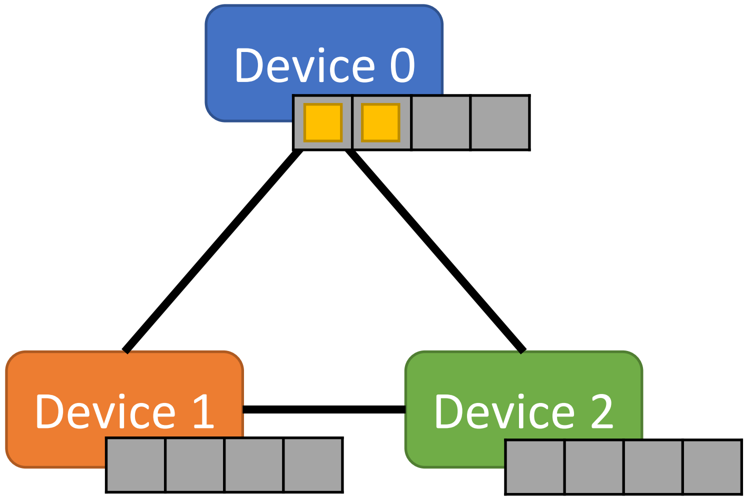
Challenge #4: Different Network Architecture

- m-SCT and m-ETF assume *parallel communication*

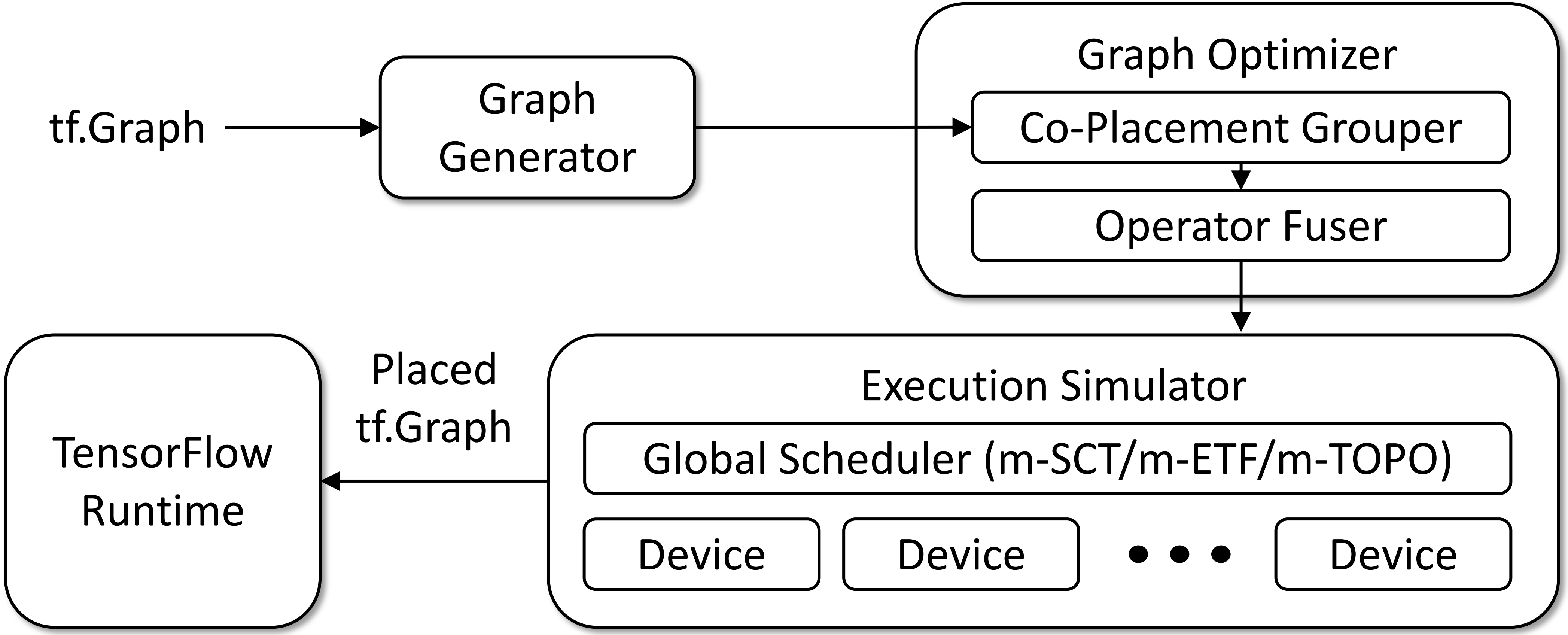
- Environment with a constrained network
  - Only *sequential communication* is supported
  - E.g., Indirect GPU-to-GPU communication

⇨ *Sequential Communication Support*

- Introduce device communication queues
- Support computation-communication overlap
- *Cache* received data to avoid duplicate transfers



Baechi WorkFlow



How Long Does It Take to Generate Placement?

- 4 NVIDIA RTX 2080 GPUs (8GB) with shared communication
  - No NVLink (Direct GPU-to-GPU communication)

Model	HierarchicalRL [34]	Placeto [2]	Baechi (m-SCT)
Inception-V3	11 hrs 50 mins	1 hr 49 mins	1-10 seconds
NMT (GNMT)	1 day 21 hrs 14 mins	2 days 20 hrs 40 mins	1.2-48 seconds

Inception-V3:
654×–42.6K× Speedup

GNMT:
3392×–206K× Speedup

How Fast Are Placed Models (Step Times)?

- Expert-designed placement
  - Inception V3 [4], GNMT [2]

Model	Batch Size	Single GPU	Expert	m-TOPO	m-ETF	m-SCT	Speedup over Single GPU m-ETF	Speedup over Single GPU m-SCT	Speedup over Expert (4 GPUs) m-ETF	Speedup over Expert (4 GPUs) m-SCT
Inception-V3	32	0.269	0.269	0.286	0.269	0.269	0.00% (1 GPU Expert)			
	64	0.491	0.491	0.521	0.491	0.491	0.00% (1 GPU Expert)			
GNMT (length: 40)	128	0.251	0.214	0.265	0.224	0.212	12.1%	18.4%	-4.5%	0.9%
	256	0.474	0.376	0.481	0.354	0.369	33.9%	28.5%	6.2%	1.9%
GNMT (length: 50)	128	0.319	0.259	0.348	0.264	0.267	20.9%	19.5%	-1.9%	-3.0%
	256	0.618	0.484	0.609	0.502	0.516	23.1%	19.8%	-3.6%	-6.2%

m-TOPO:
up to 34% higher than expert

m-ETF
-4.5% to **6.2%** speedup

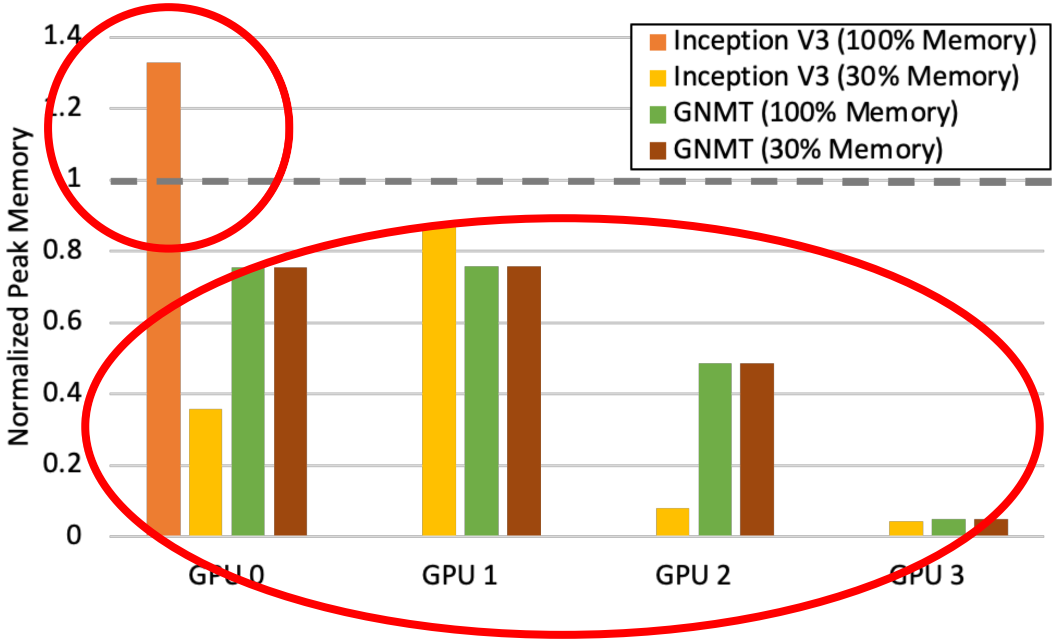
m-SCT
-6.2% to **1.9%** speedup

[2] Wu et al., Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. arXiv:1609.08144
[4] Mirhoseini et al., Hierarchical Planning for Device Placement. ICLR '18

What If Memory Is Constrained?

- 30% per GPU memory (2.4 GB)

Model	Single GPU	Expert	m-TOPO	m-ETF	m-SCT
Inception-V3	OOM	OOM	0.690 (58.6%)	0.312 (13.8%)	0.292 (7.9%)
GNMT	OOM	0.221 (3.2%)	0.272 (2.6%)	0.230 (2.6%)	0.212 (0.0%)



m-SCT: only up to 13.8% slower than sufficient memory

How Much Are Optimization Benefits?

- All optimizations applied (m-SCT)

Model	Un-Optimized			Optimized		
	Num. Ops	Placement (seconds)	Step (seconds)	Num. Ops	Placement (seconds)	Step (seconds)
Inception-V3	6884	68.0	0.302	17	0.9	0.269
GNMT (length: 40)	18050	275.1	0.580	542	1.2	0.212
GNMT (length: 50)	22340	406.1	0.793	706	2.4	0.267

Number of Operators:
96.8%–99.8% Reduction

Placement times:
75.6×–229.3× Speedup

Step times:
1.1×–3.0× Speedup

Takeaways

- Current state-of-the-art learning-based ML placement algorithms
  - Require **very long** placement time (2 hours ~ 3 days)
  - Require **re-training** the placement model on ML model and environment changes
- **Baechi** is a ***fast*** placement system by using ***algorithmic*** approaches
  - Placement algorithms for memory-constrained environments
    - *m-TOPO* (Topological Sort)*, m-ETF* (Earlier Task First), *m-SCT* (Small Communication Time)
  - Optimizations
    - Co-adjust Placement, Co-placement, Operator Fusion, Sequential Communication Support
  - **Place fast: 654–206K×** faster placement time than learning-based approaches
    - Place ML models on 4 GPUs within *only* **1.2** seconds
  - **Place well**: *only* up to **6.2%** higher step time than expert's placements