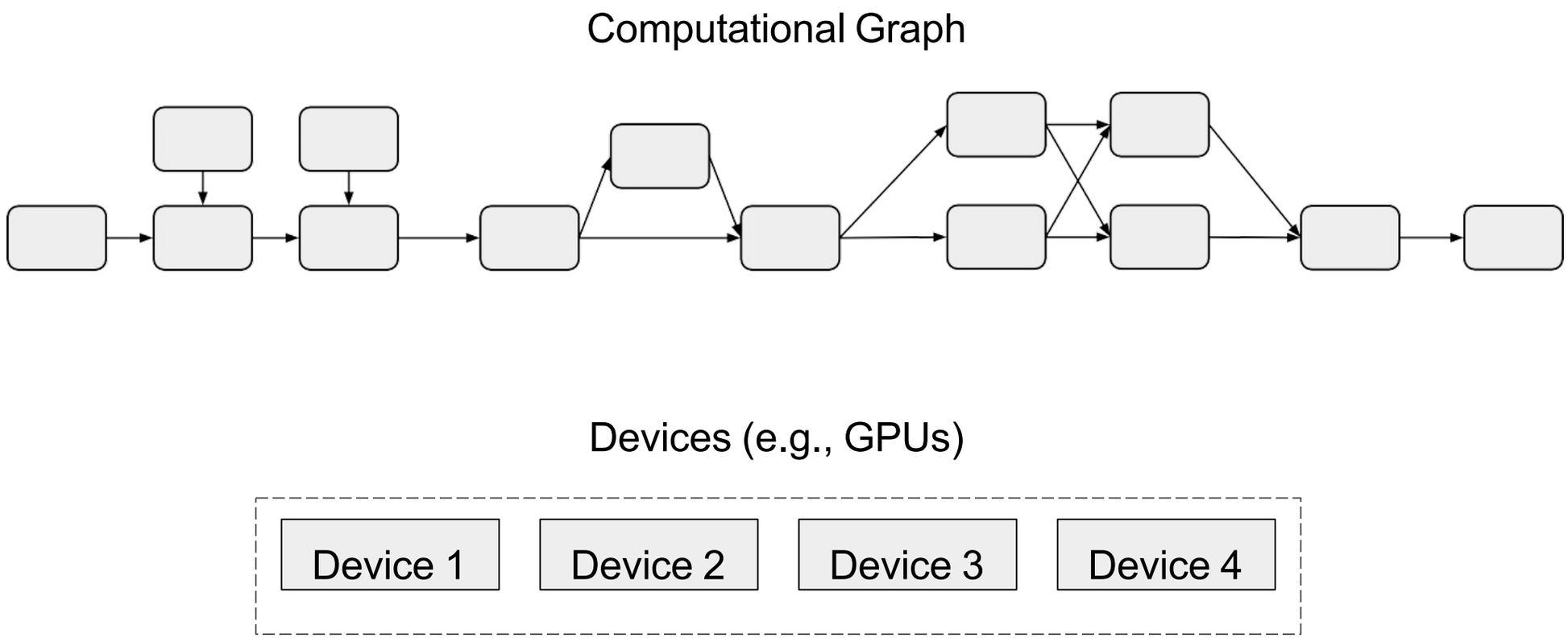
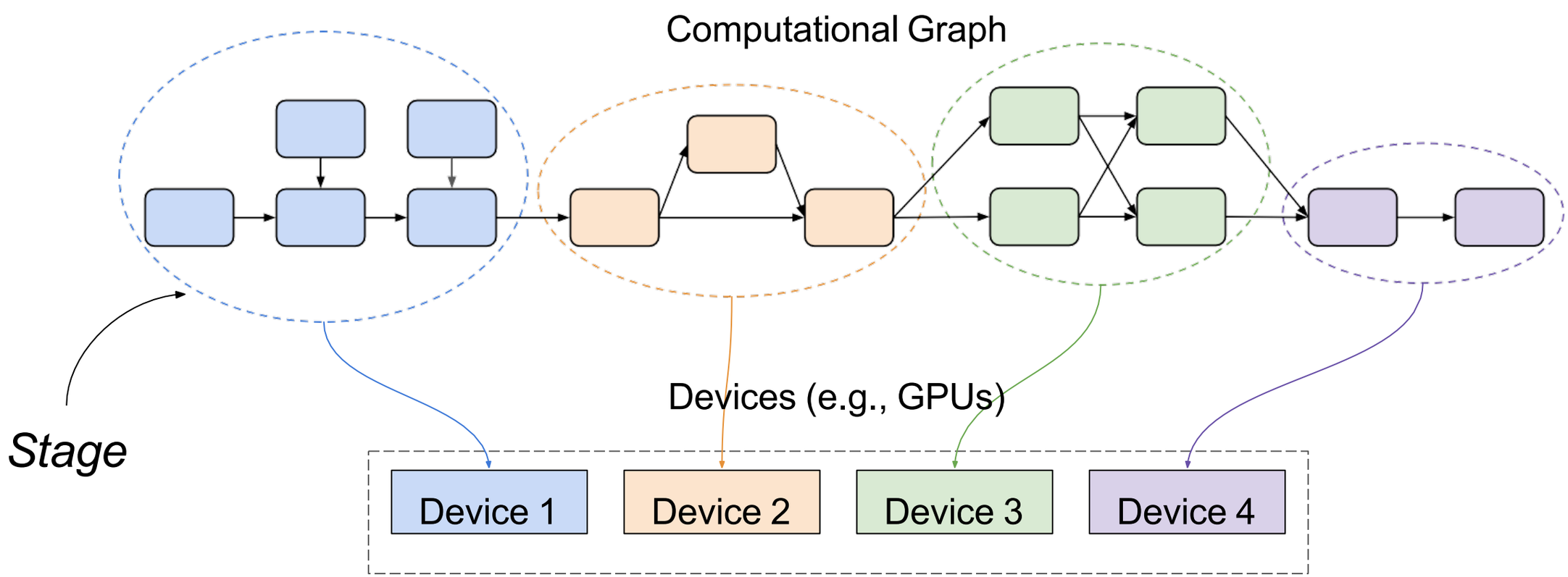
Solution 2: Model Parallelism

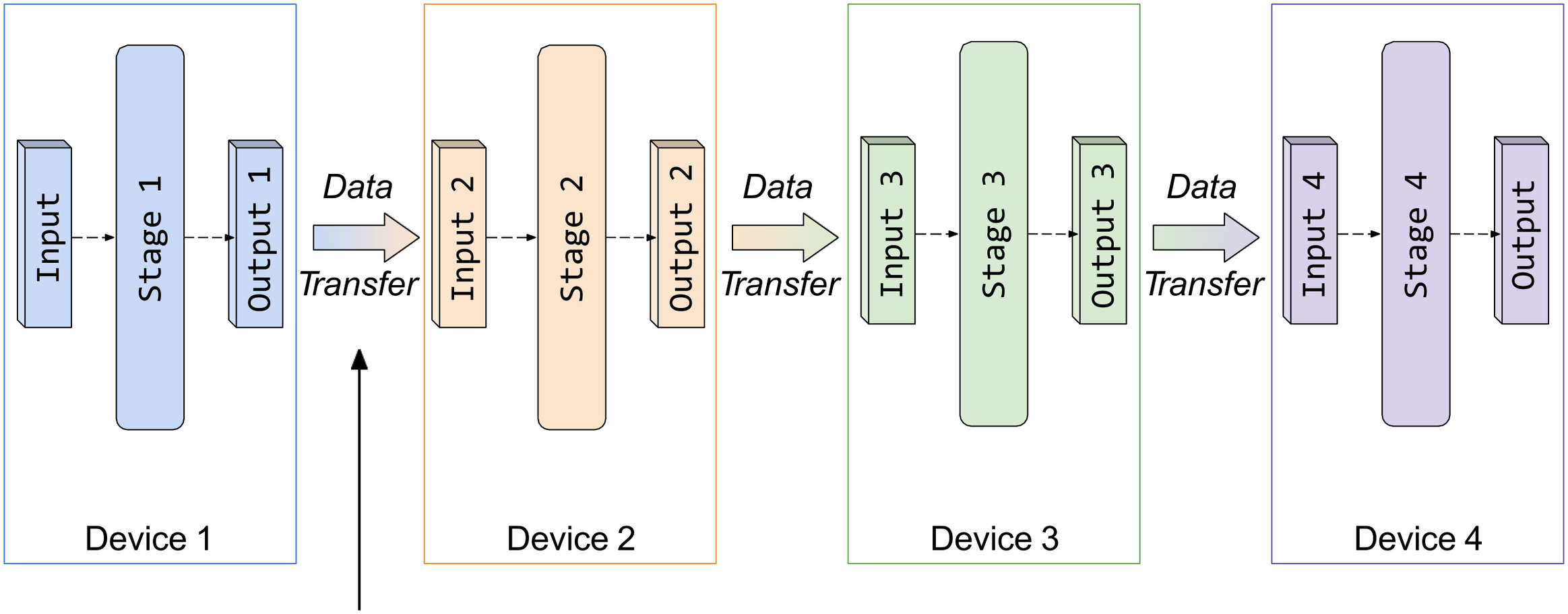
Computational Graph (Neural Networks) → Stages



Computational Graph (Neural Networks) → Stages

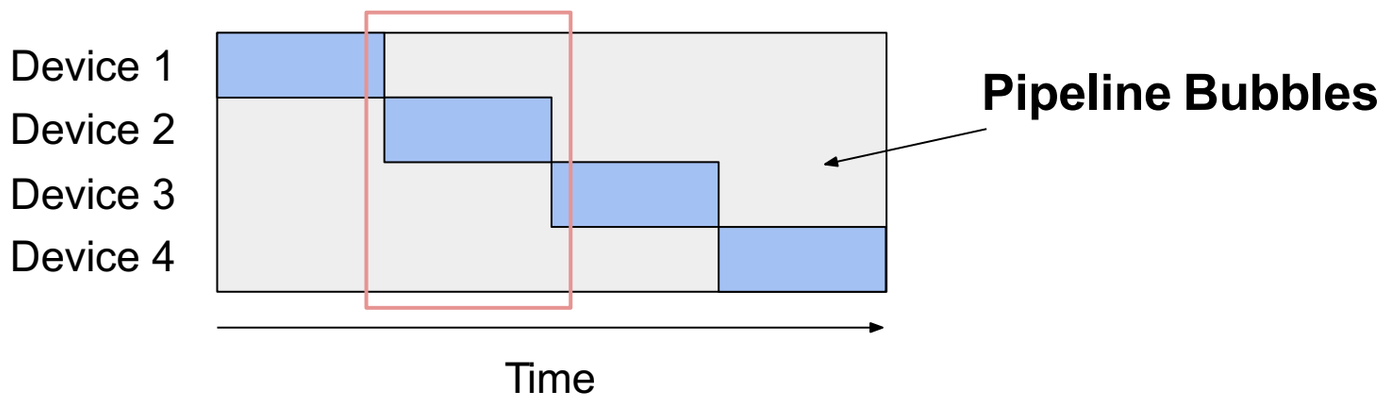


Execution & Data Movement



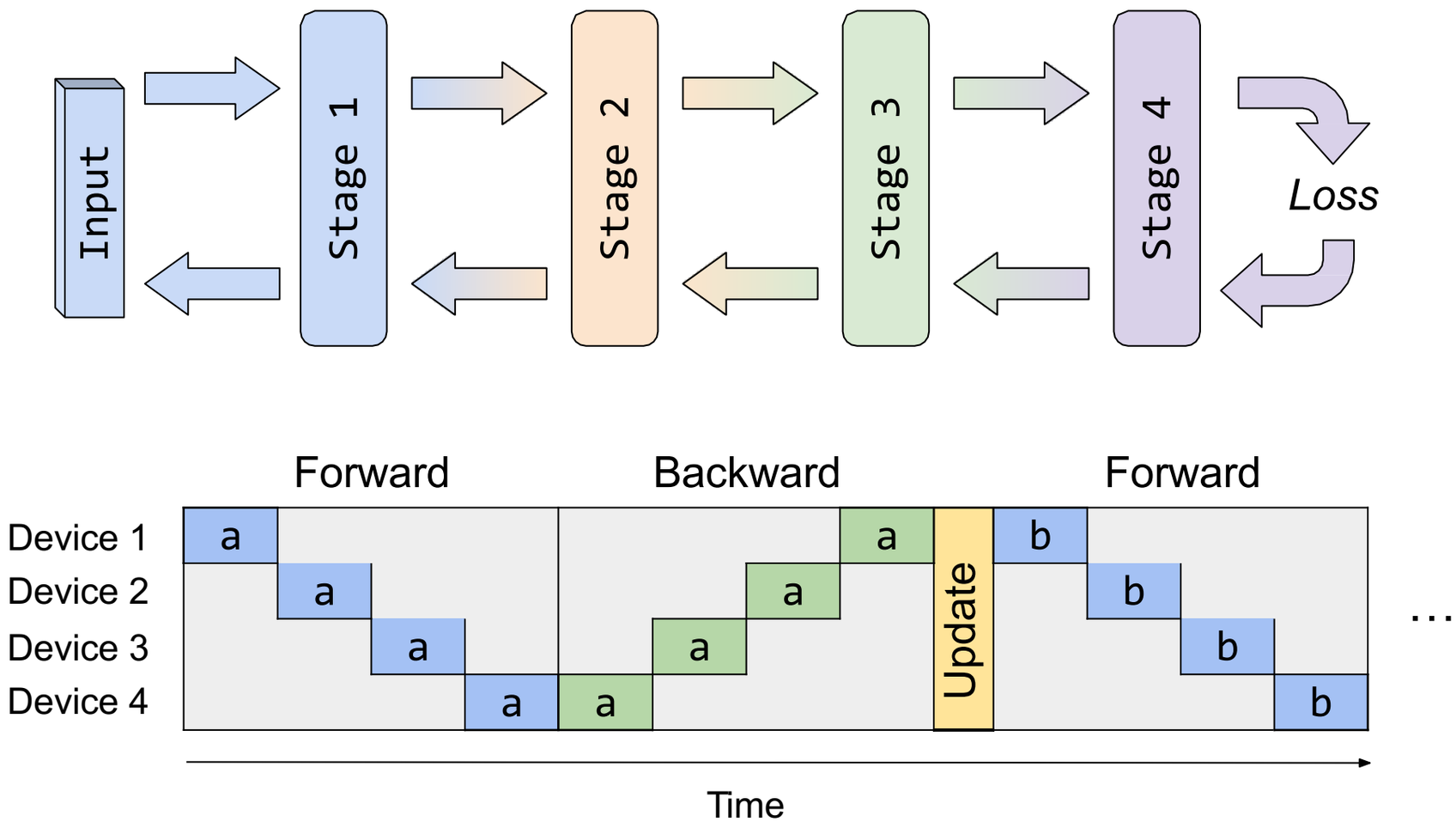
Note: The time spent on data transfer is typically **small,** since we only communicates stage outputs at stage boundaries between two stages.

Timeline: Visualization of Inter-Operator Parallelism



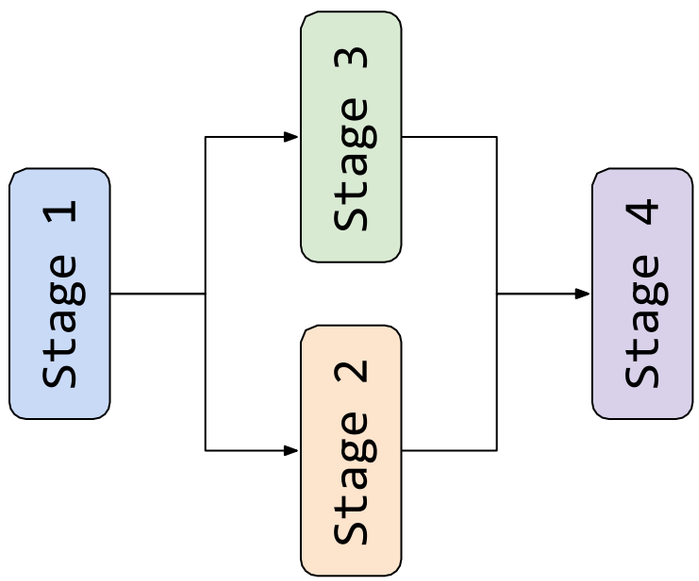
- Gray area ( ) indicates devices being idle (a.k.a. Pipeline bubbles).
- Only 1 device activated at a time.
- **Pipeline bubble percentage** `= bubble_area / total_area`
  `= (D - 1) / D`, assuming `D` devices.

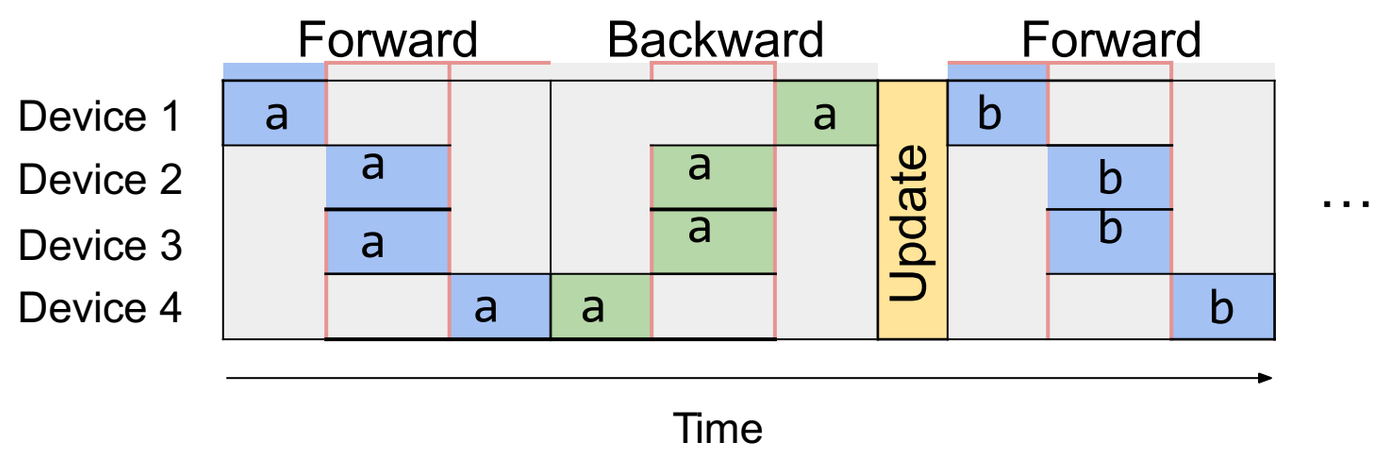
Training: Forward & Backward Dependency



Device Placement

Idea: Slice the branches of a neural network into multiple stages so they can be calculated concurrently.

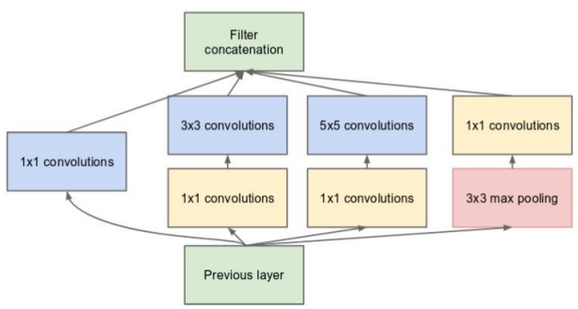




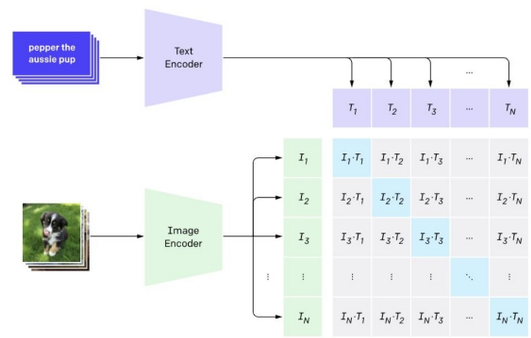
Mirhoseini, Azalia, et al. "Device placement optimization with reinforcement learning." *ICML 2017.*

Device Placement: Limitations

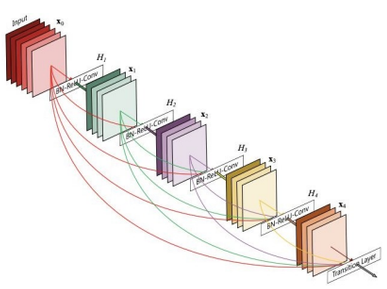
Only works for specific NNs with branches:



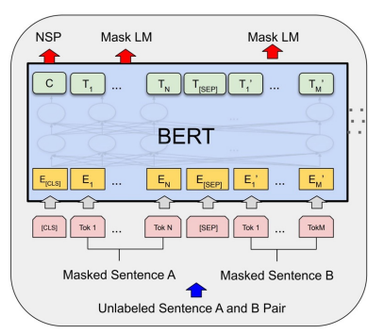
✅ Inception Module



✅ Contrastive Model

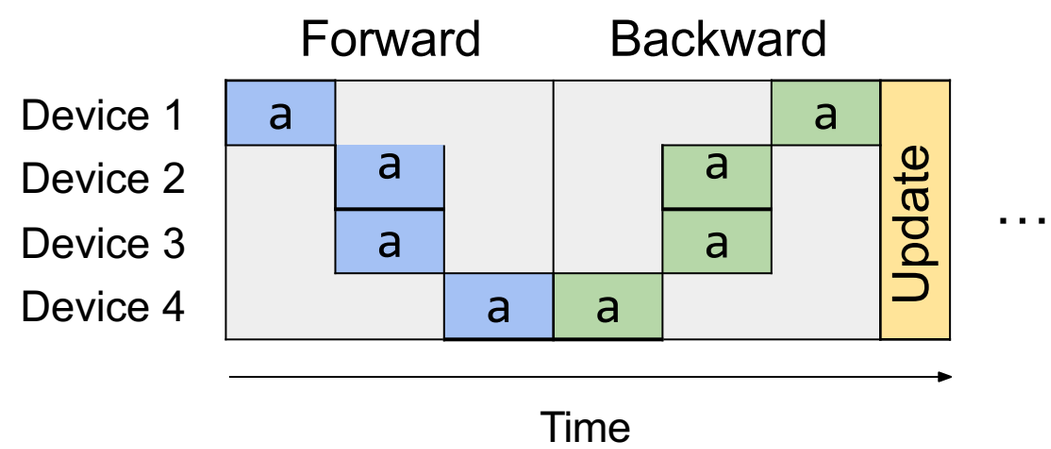


❌Other ConvNets



❌Transformers

Device Utilization is still low:



...

Note: device placement needs to be combined with the other pipeline schedules discussed later to further improve device utilization.

Synchronous Pipeline Parallel Schedule

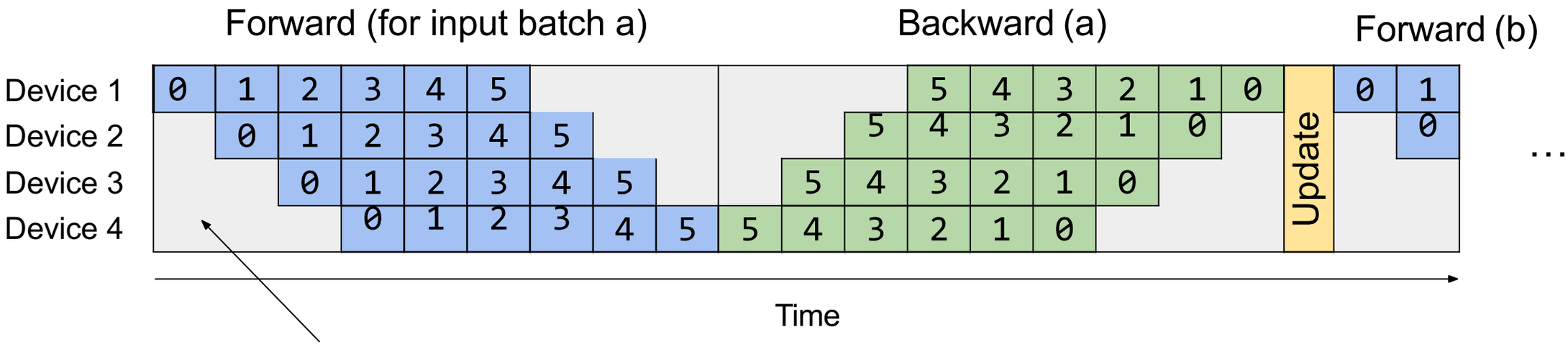
Idea: Modify pipeline schedule to improve efficiency, but keep the computation and convergence semantics exactly the same as if training with a single device.

GPipe

Idea: Partition the input batch into multiple "*micro-batches*". Pipeline the micro-batches. Accumulate the gradients of the micro-batches:

$$\nabla L_\theta(x) = \frac{1}{N} \sum_{i=1}^{N} \nabla L_\theta(x_i)$$

Example: Slice each input batch into 6 micro-batches:



Pipeline bubbles percentage = `(D - 1) / (D - 1 + N)`
with `D` devices and `N` micro-batches.

Huang, Yanping, et al. "Gpipe: Efficient training of giant neural networks using pipeline parallelism." *NeurIPS 2019.*

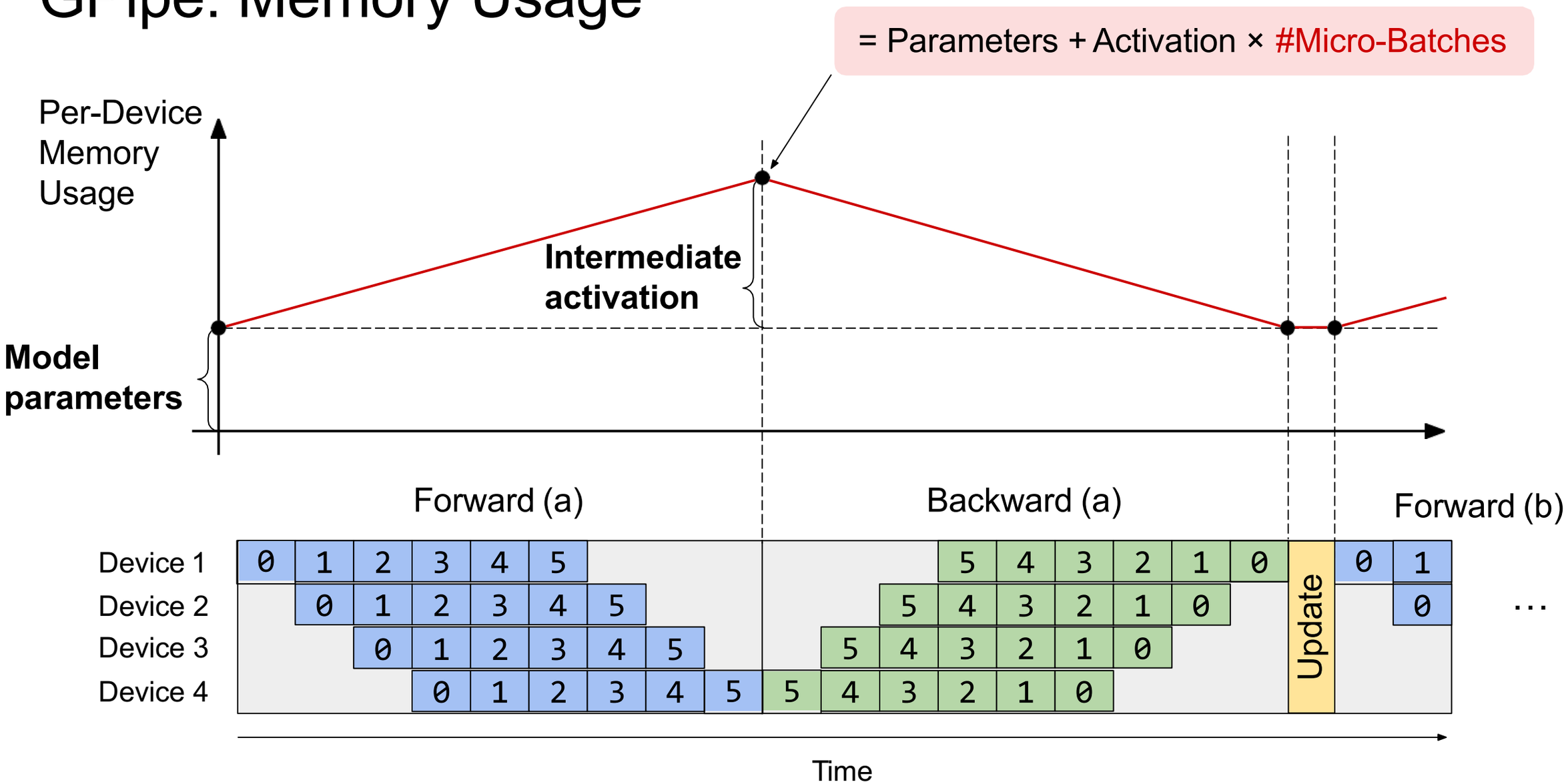
GPipe: Experimental Results

Table: Normalized training throughput using GPipe with different number of devices (stages) and different number of micro-batches M on TPUs.

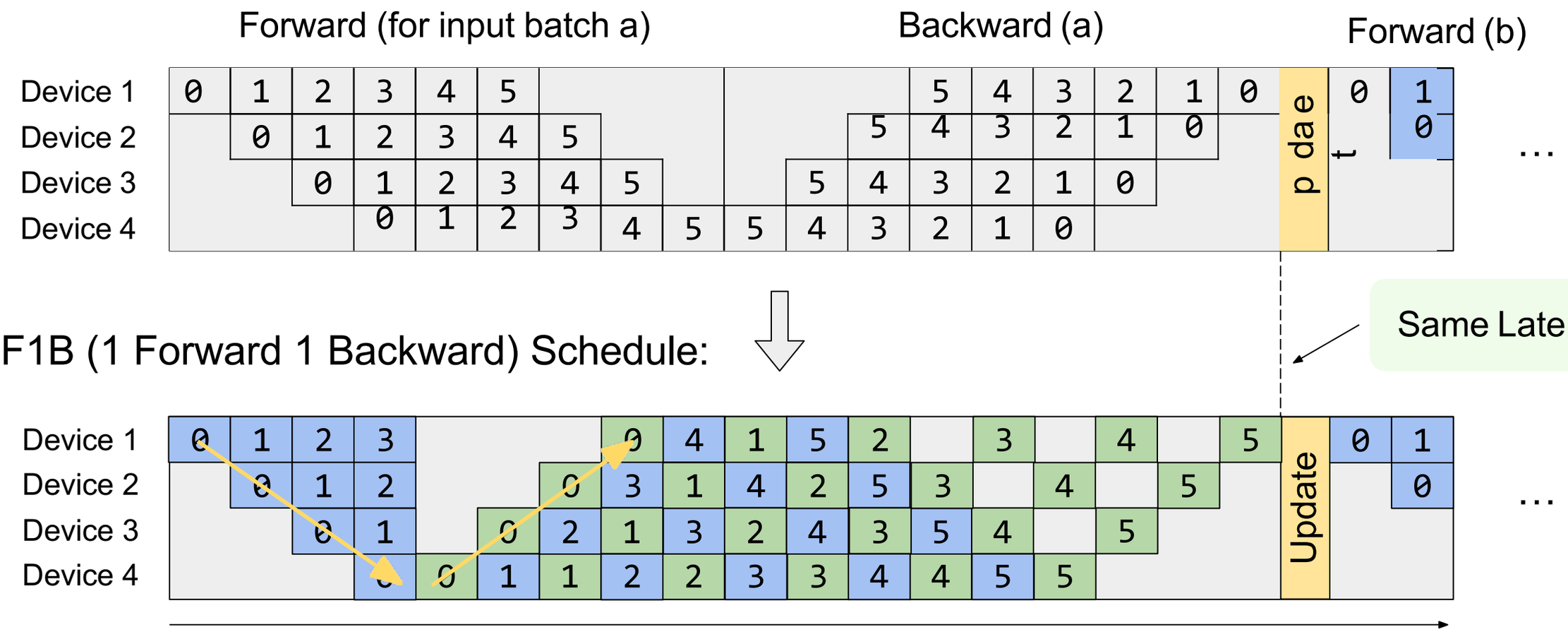
	#TPUs = 2	**#TPUs = 4**	**#TPUs = 8**
#Micro-batches = 1	1	1.07	1.3
#Micro-batches = 4	1.7	3.2	4.8
#Micro-batches = 32	1.8	3.4	6.3

Huang, Yanping, et al. "Gpipe: Efficient training of giant neural networks using pipeline parallelism." *NeurIPS 2019.*

GPipe: Memory Usage

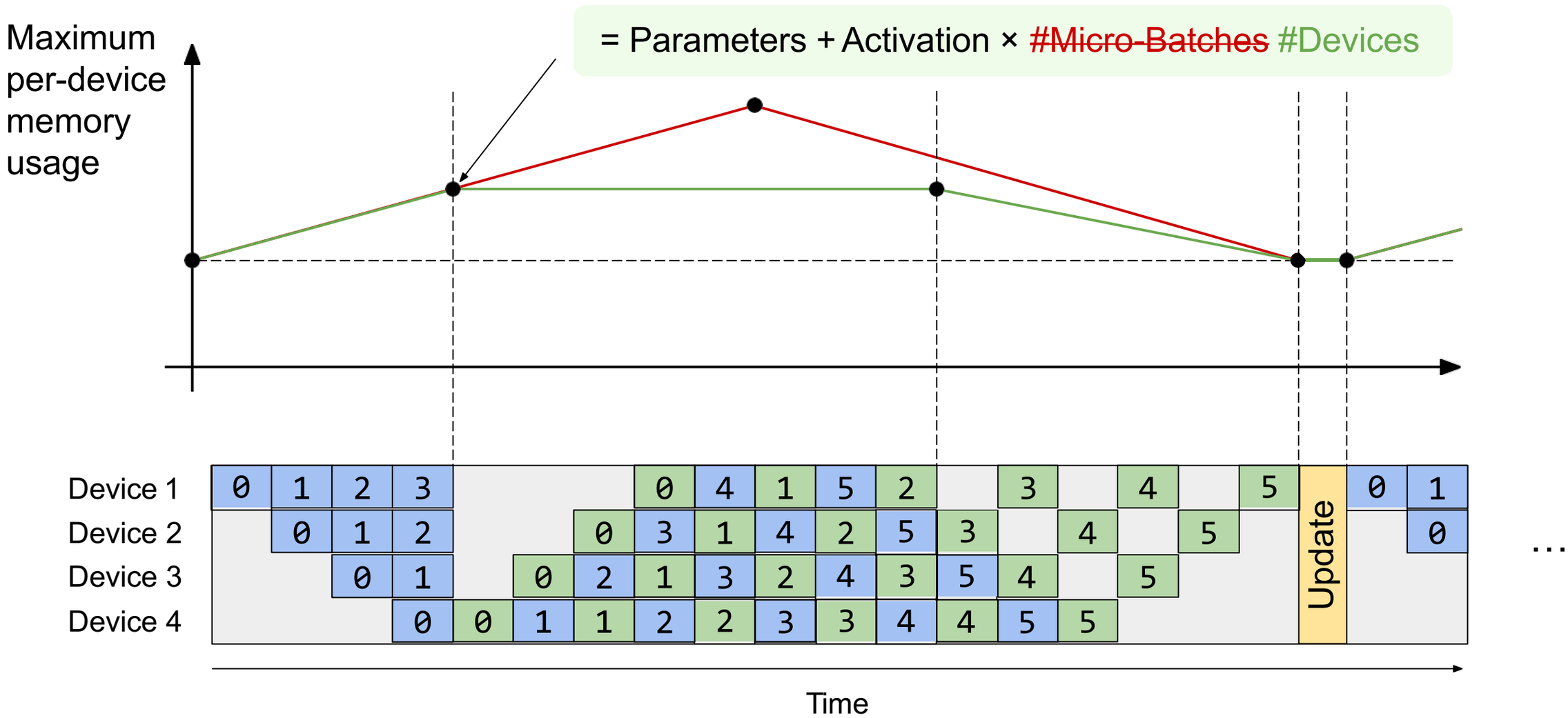


GPipe Schedule:



Fan, Shiqing, et al. "DAPPLE: A pipelined data parallel approach for training large models." *PPoPP 2021.*

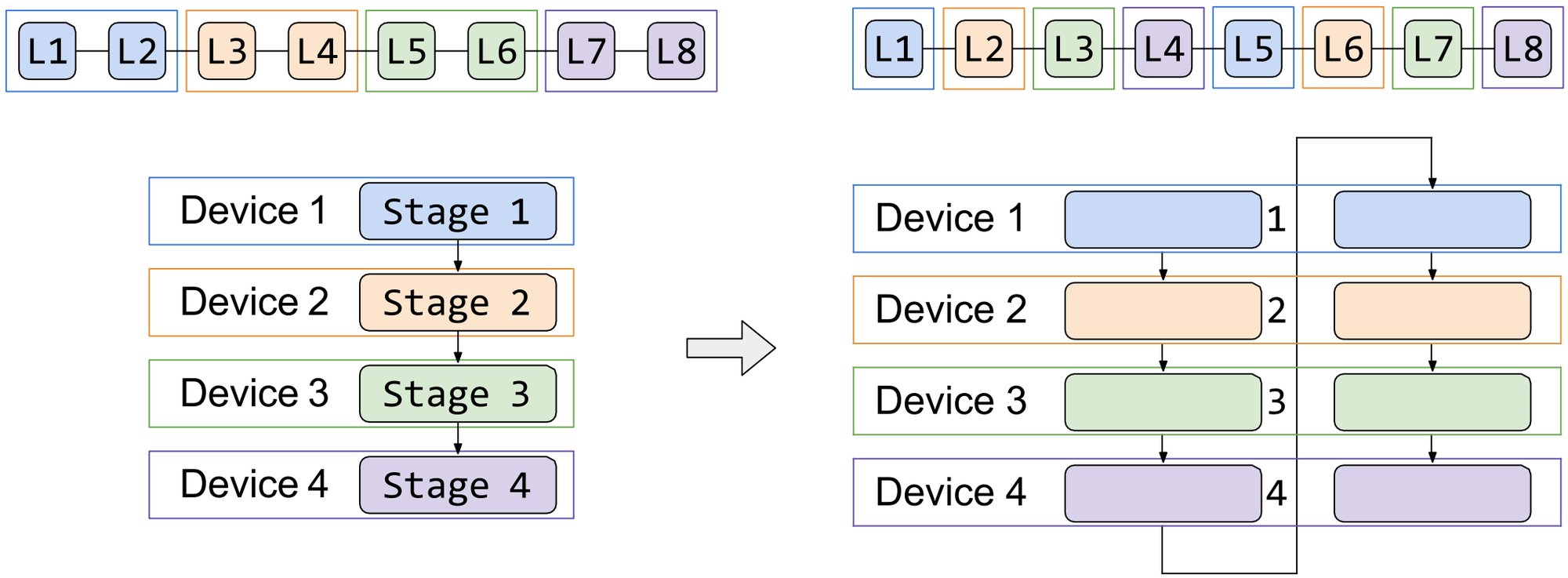
1F1B Memory Usage



Fan, Shiqing, et al. "DAPPLE: A pipelined data parallel approach for training large models." *PPoPP 2021.*

Interleaved 1F1B

Idea: Slice the neural network into more fine-grained stages and assign multiple stages to reduce pipeline bubble.

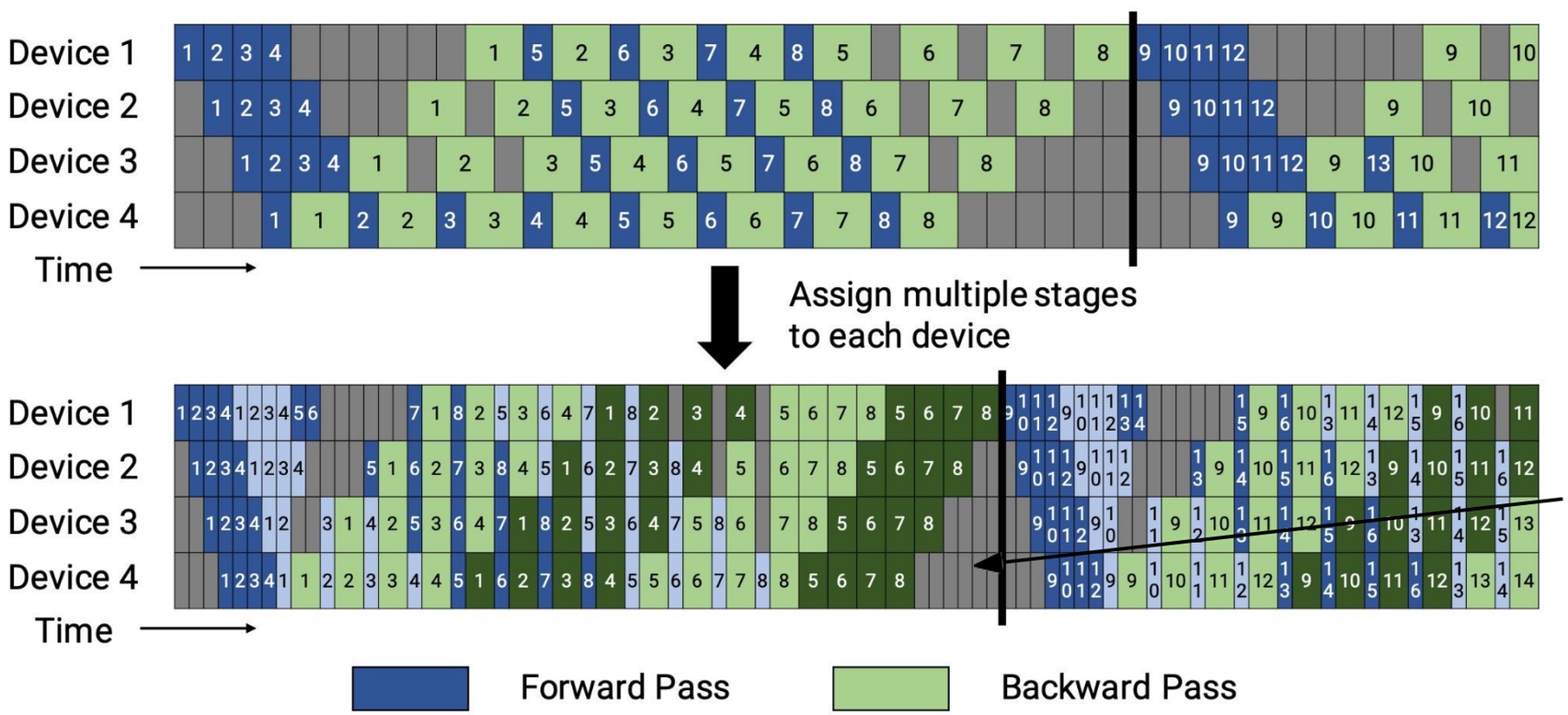


Narayanan, Deepak, et al. "Efficient large-scale language model training on gpu clusters using megatron-lm." *SC 2021.*

Interleaved 1F1B

Pro:
Higher pipeline efficiency with fewer pipeline bubbles.

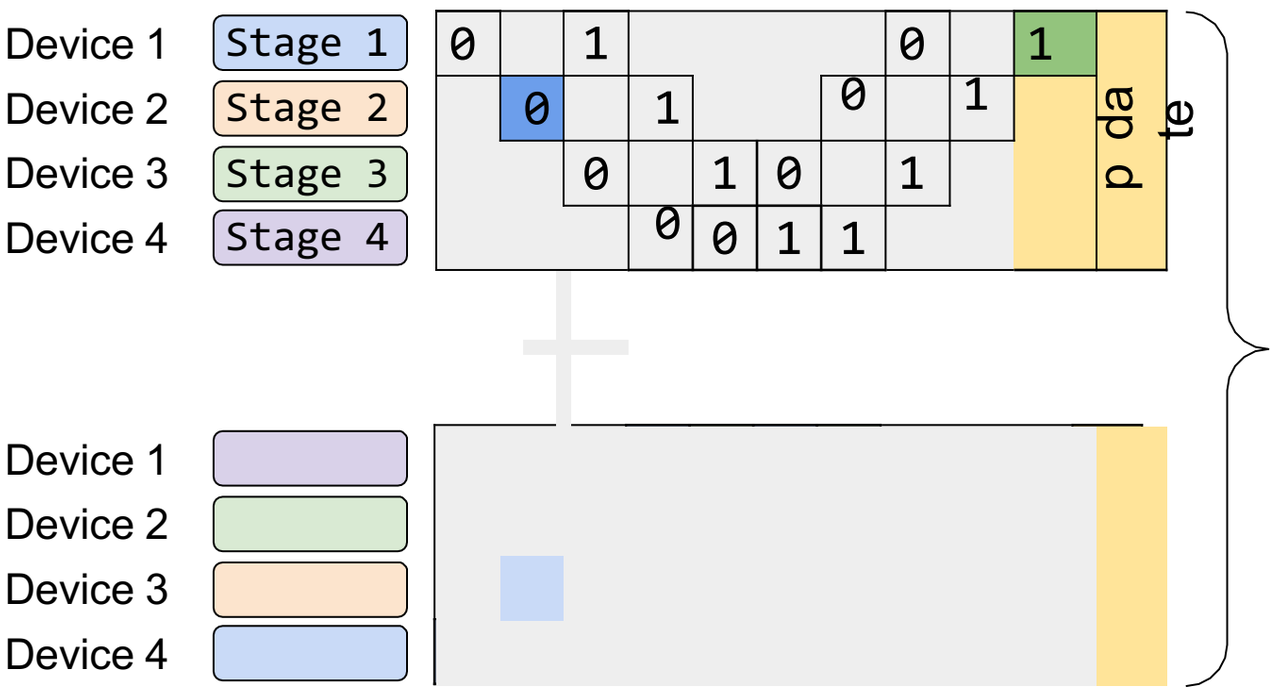
Con:
More communication overhead between stages.



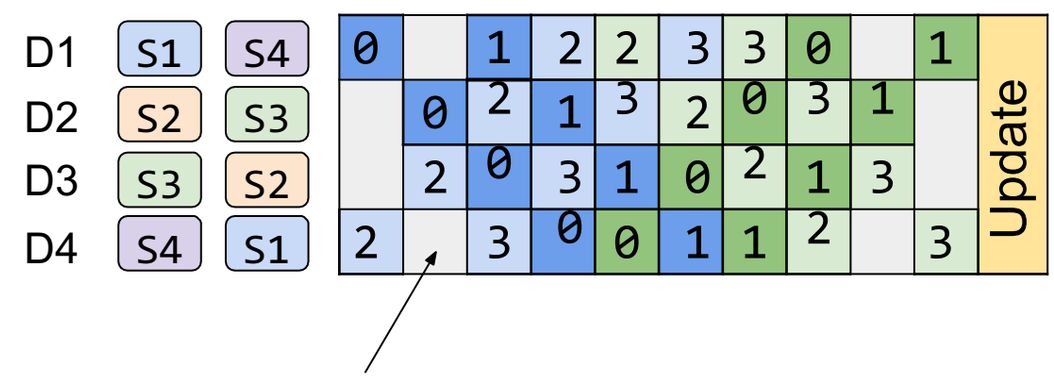
Pipeline bubbles percentage = `(D - 1) / (D - 1 + KN)` with `D` devices, `K` stages on each device, and `N` micro-batches.

Chimera

Idea: Store bi-directional stages and combine bidirectional pipeline to further reduce pipeline bubbles.



Extra copy of parameters & extra synchronization.



Pipeline bubbles percentage
= (D - 2) / (D - 2 + 2N)
with D devices and N micro-batches.

Li, Shigang, and Torsten Hoefler. "Chimera: efficiently training large-scale neural networks with bidirectional pipelines." *SC 21.*