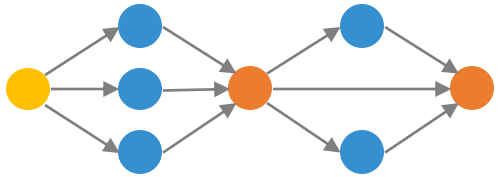
System for Artificial Intelligence

Parallelization and Training II

Siyuan Feng
Shanghai Innovation Institute

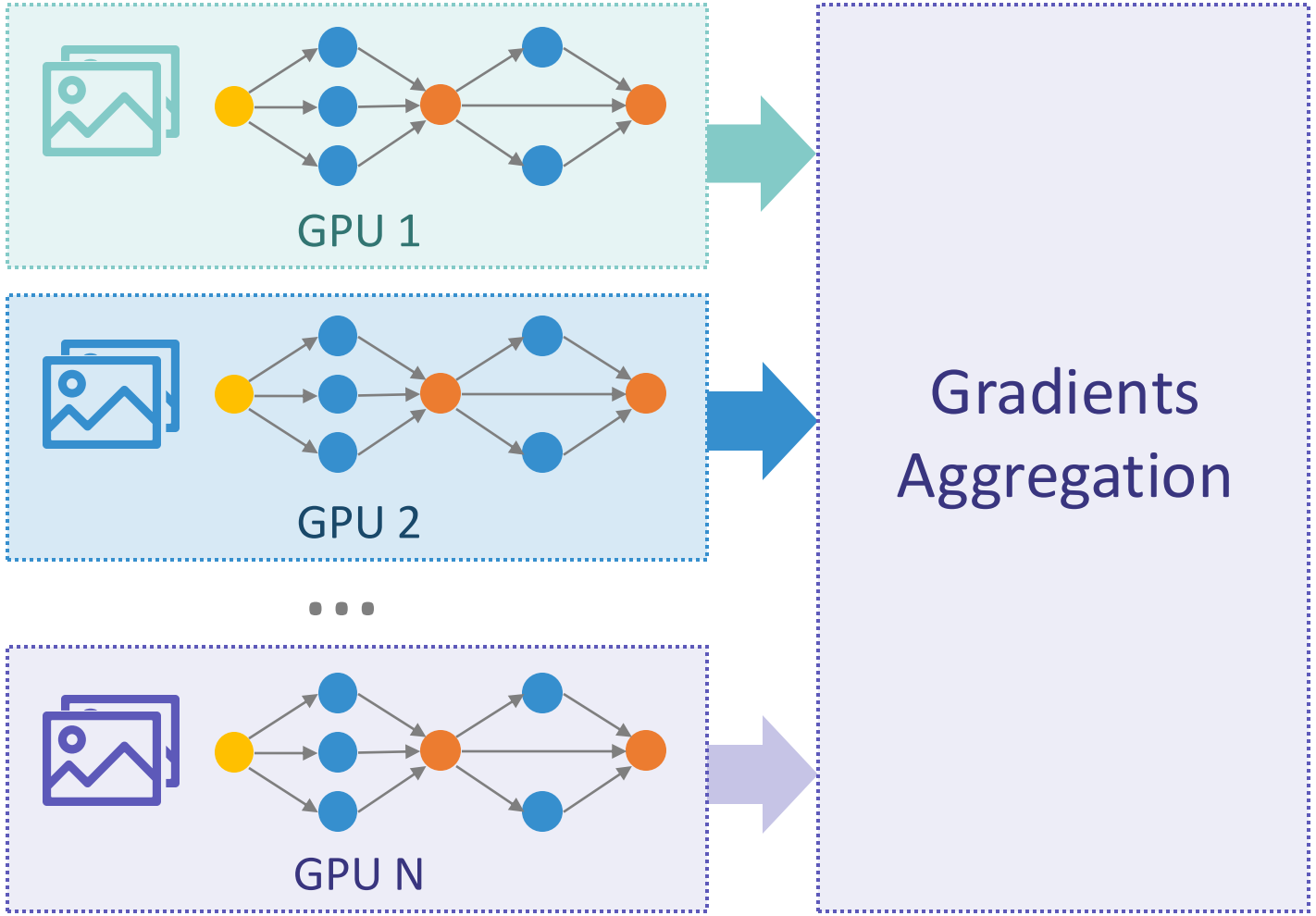
Recap: Data Parallelism



ML Model

Training Dataset

$$w_i := w_i - \gamma \nabla L(w_i) = w_i - \frac{\gamma}{n} \sum_{j=1}^{n} \nabla L_j(w_i)$$



1. Partition training data into batches

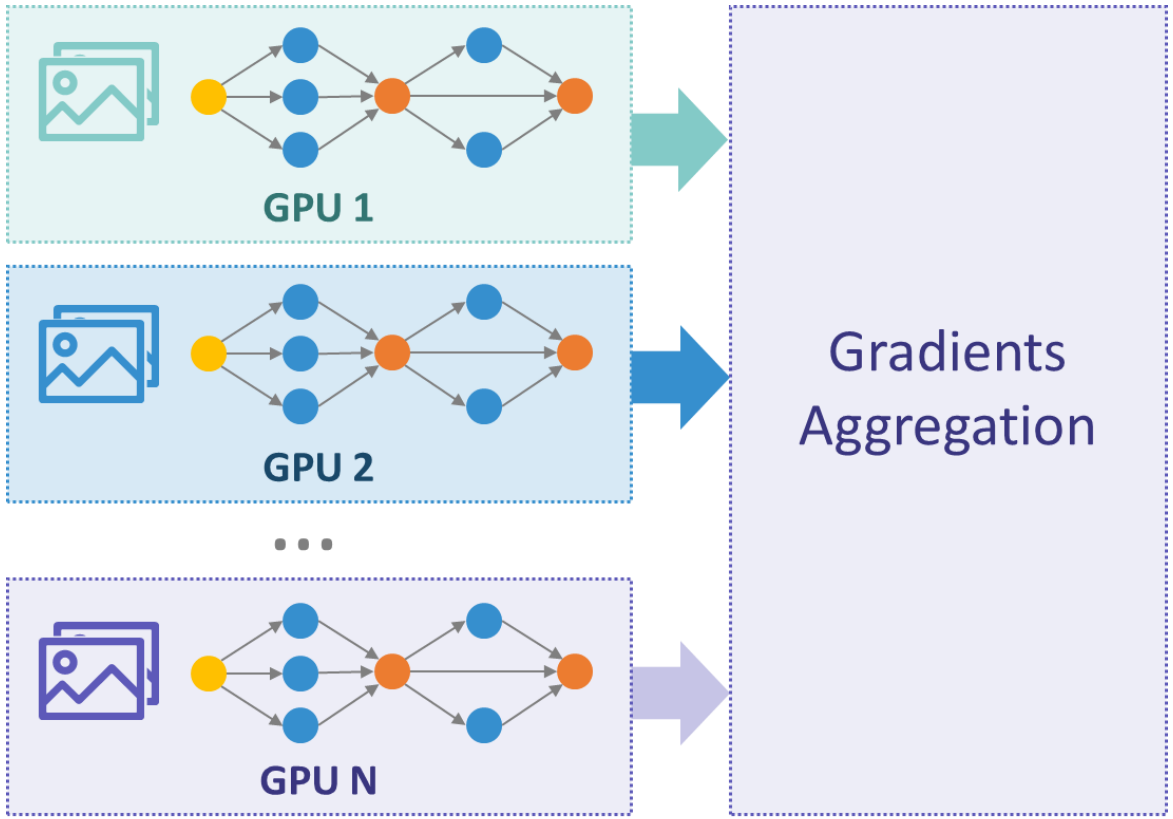
2. Compute the gradients of each batch on a GPU

3. Aggregate gradients across GPUs

Recap: An Issue with Data Parallelism

上海创智学院
Shanghai Innovation Institute

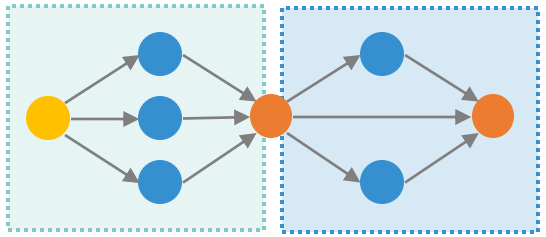
- Each GPU saves a replica of the entire model
- Cannot train large models that exceed GPU device memory



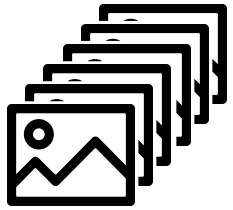
Model Parallelism

上海创智学院
Shanghai Innovation Institute

- Split a model into multiple subgraphs and assign them to different devices

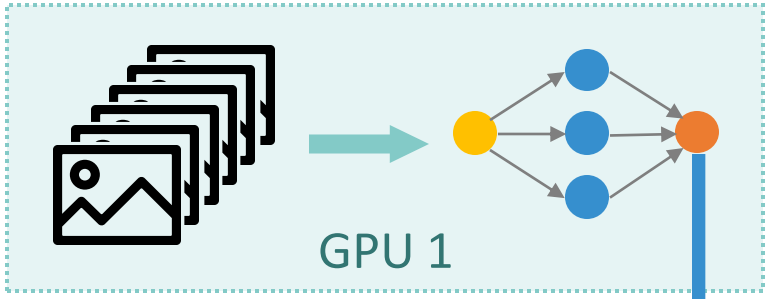


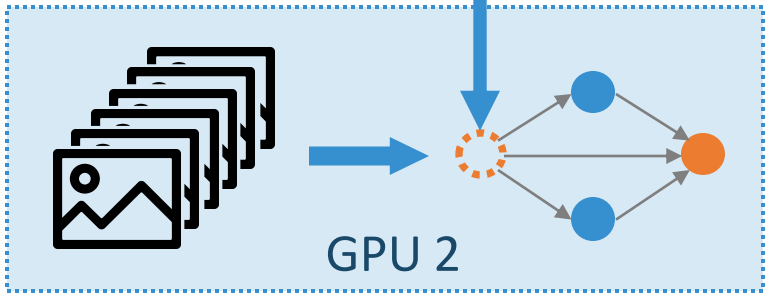
ML Model



Training Dataset

Model Parallelism





Transfer intermediate results between devices

$$w_i := w_i - \gamma \nabla L(w_i) = w_i - \frac{\gamma}{n} \sum_{j=1}^{n} \nabla L_j(w_i)$$

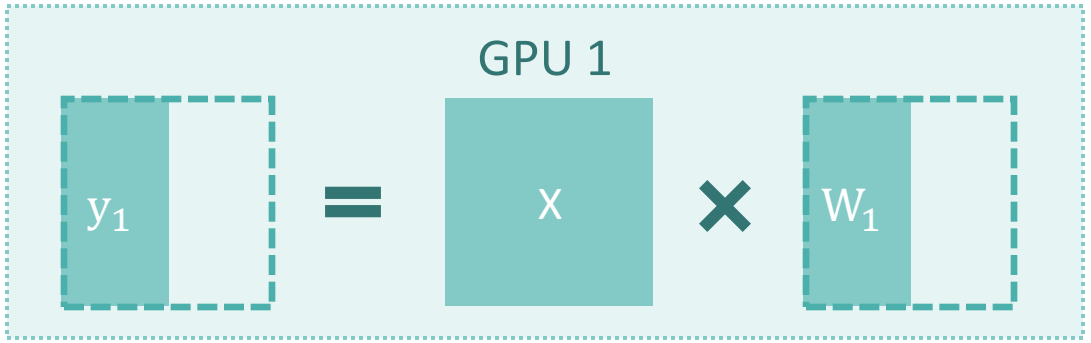
How to parallelize DNN Training?

上海创智学院
Shanghai Innovation Institute

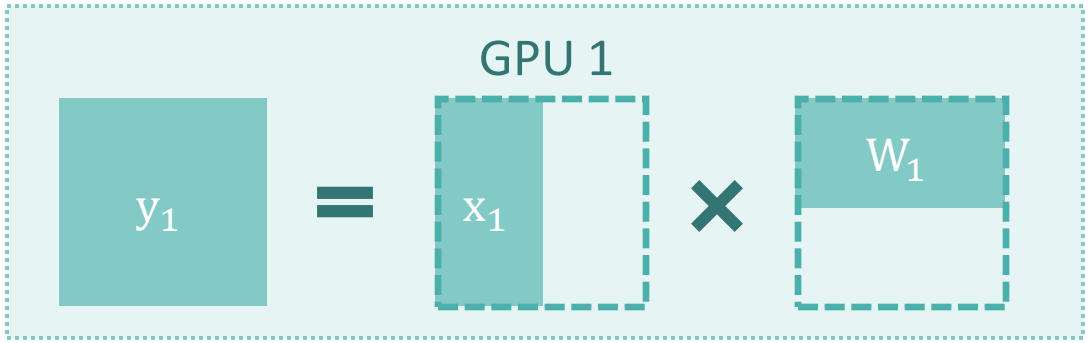
- Data parallelism
- Model parallelism
  - Tensor model parallelism
  - Pipeline model parallelism

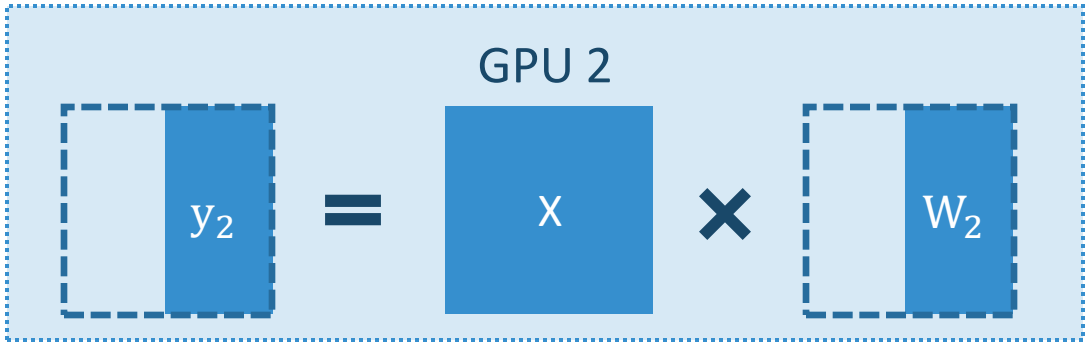
Tensor Model Parallelism

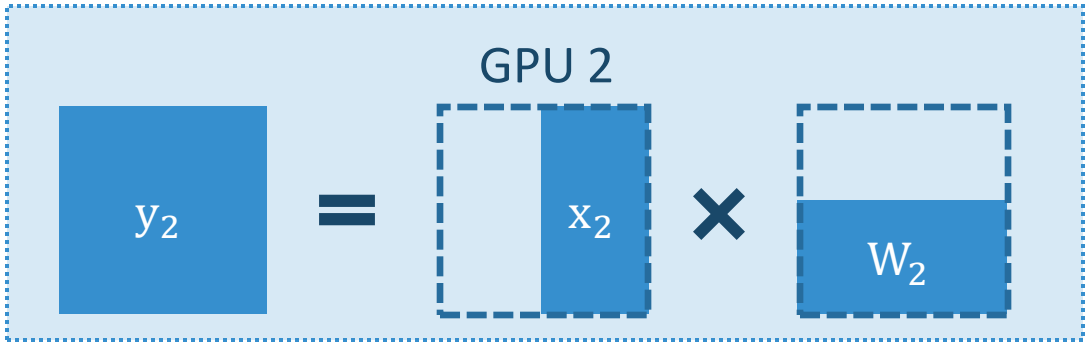
- Partition parameters/gradients ***within*** a layer



Tensor Model Parallelism (partition output)



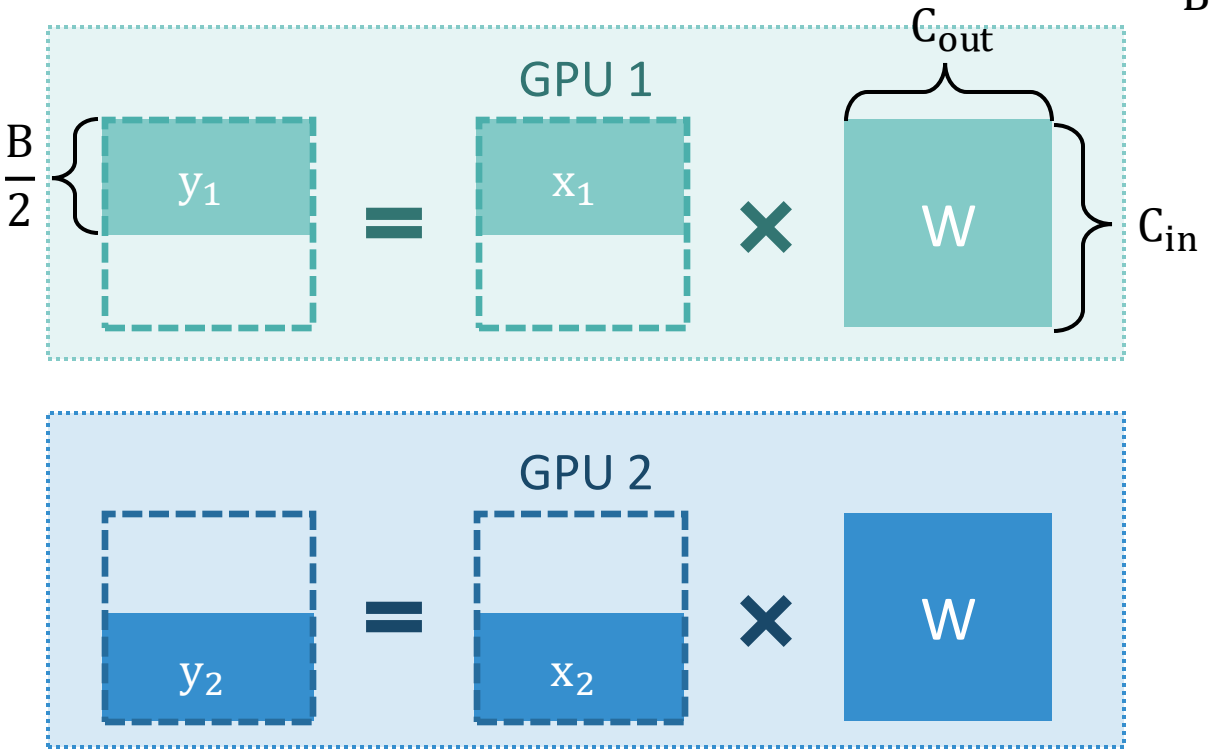




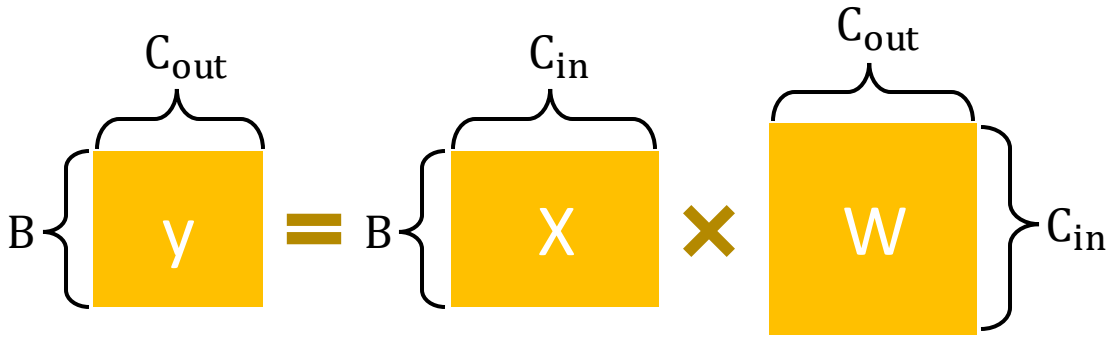
Tensor Model Parallelism (reduce output)

$y = y1 + y2$

Comparing Data and Tensor Model Parallelism



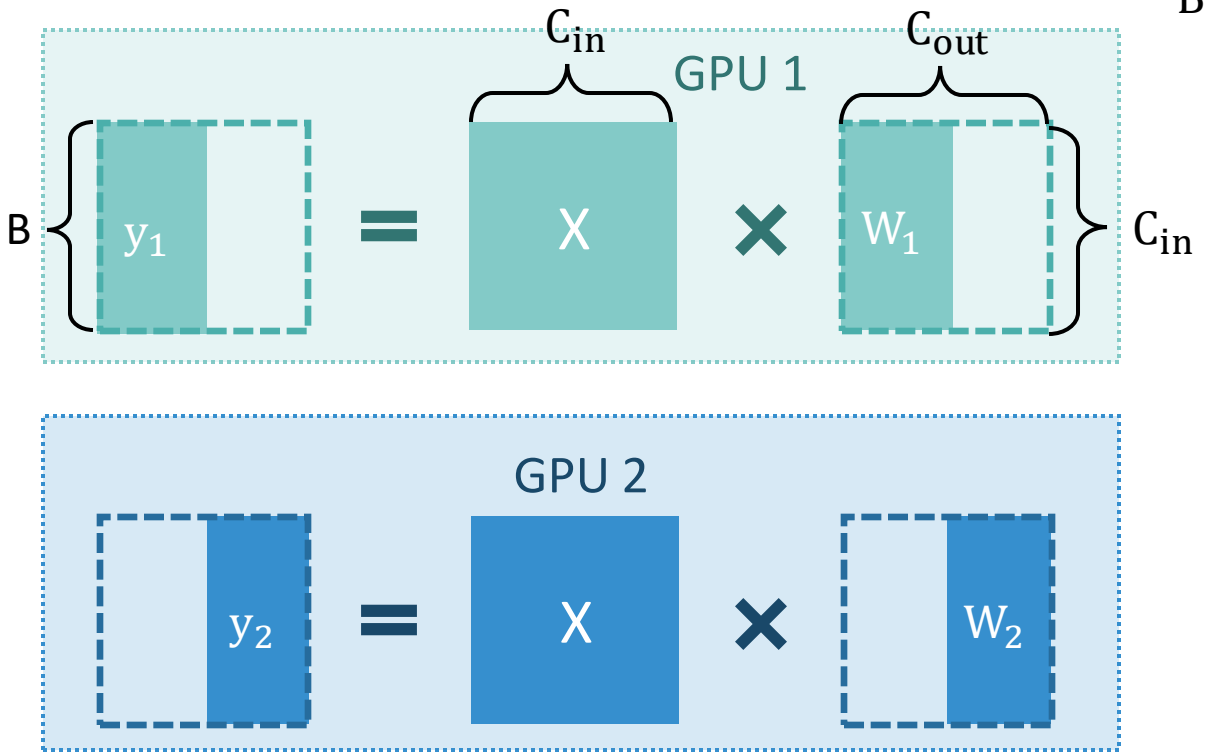
Data parallelism $y = Wx$



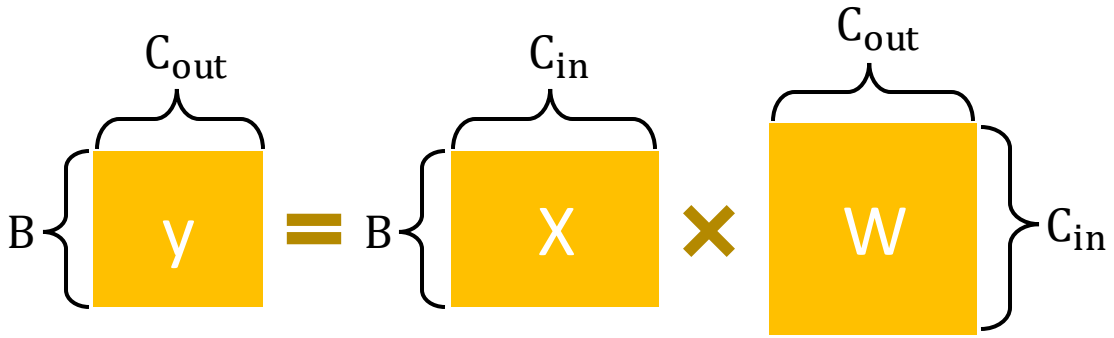
Forward Processing	Backward Propagation	Gradients Sync
0	0	$O(Cout * Cin)$

Communication Cost of Data Parallelism

Comparing Data and Tensor Model Parallelism



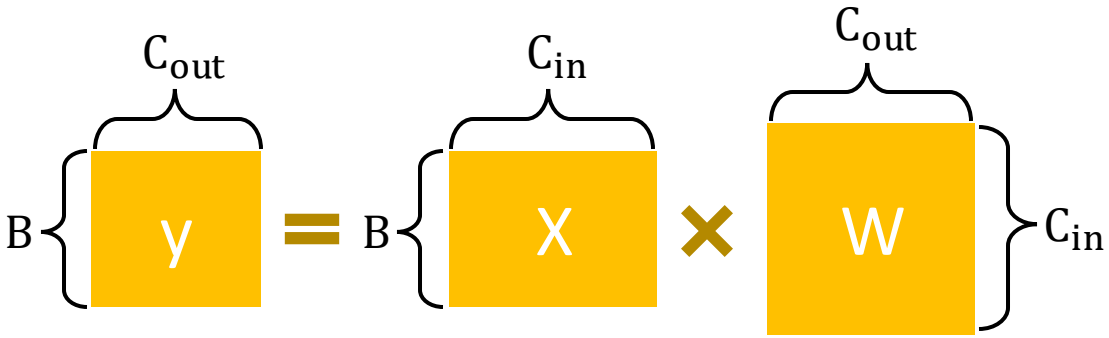
Tensor Model Parallelism (partition output)

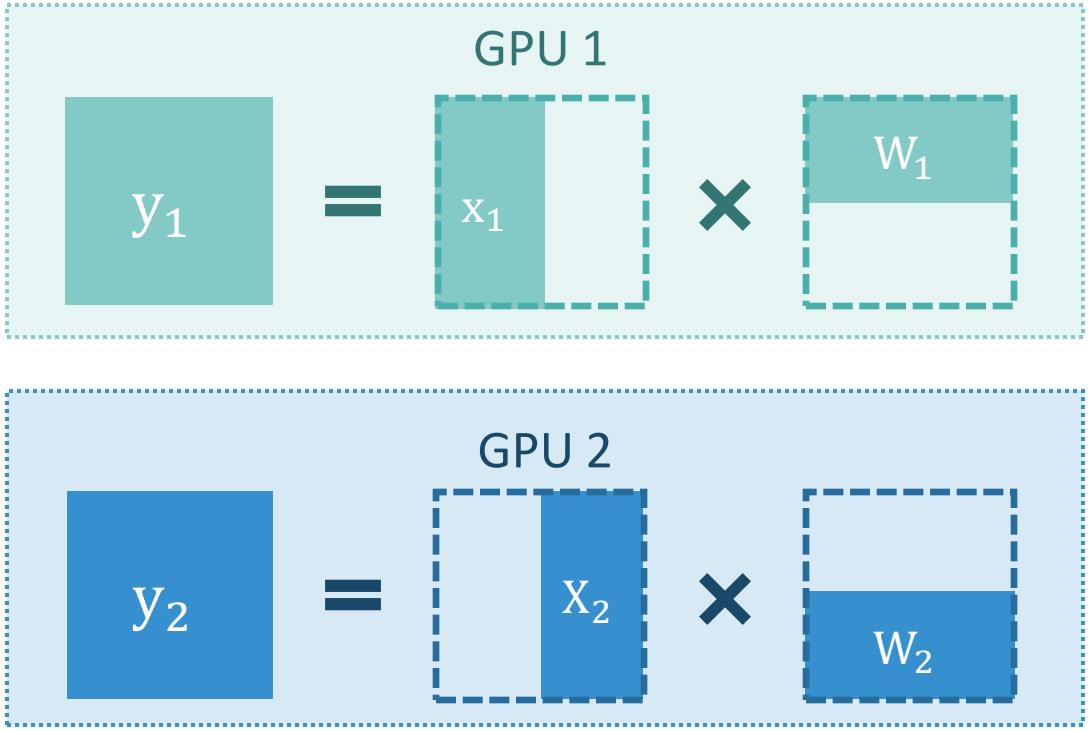


Forward Processing	Backward Propagation	Gradients Sync
$O(B * Cin)$	$O(B * Cin)$	0

Communication Cost of Tensor Model Parallelism

Comparing Data and Tensor Model Parallelism





Tensor Model Parallelism (Reduce output)

$y = y1 + y2$

Forward Processing	Backward Propagation	Gradients Sync
$O(B * Cout)$	$O(B * Cout)$	0

Communication Cost of Tensor Model Parallelism

Comparing Data and Tensor Model Parallelism

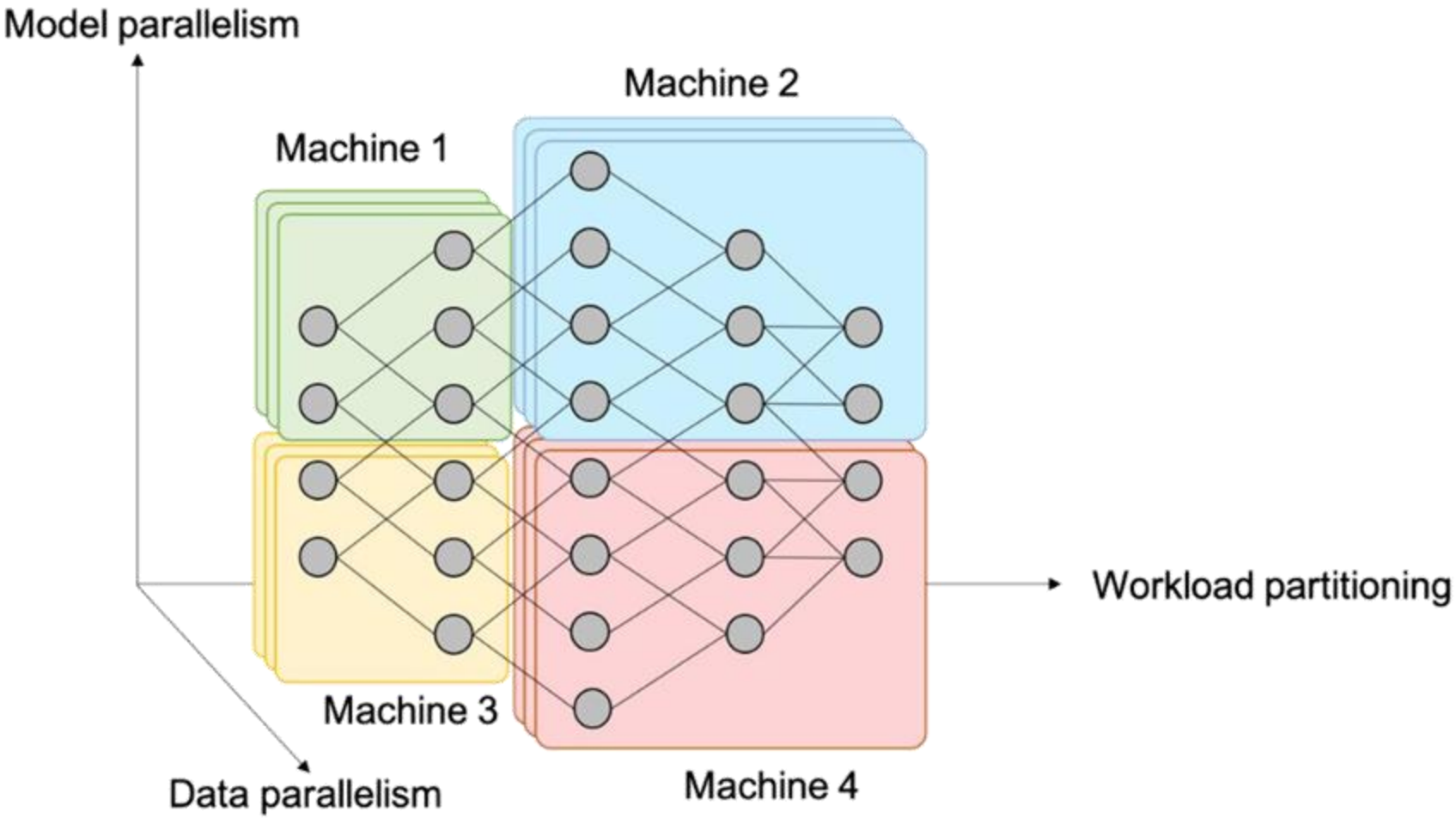
上海创智学院
Shanghai Innovation Institute

- Data parallelism: $O(Cout * Cin)$
- Tensor model parallelism (partition output): $O(B * Cin)$
- Tensor model parallelism (reduce output): $O(B * Cout)$

- **The best strategy depends on the model and underlying machine**

Combine Data and Model Parallelism

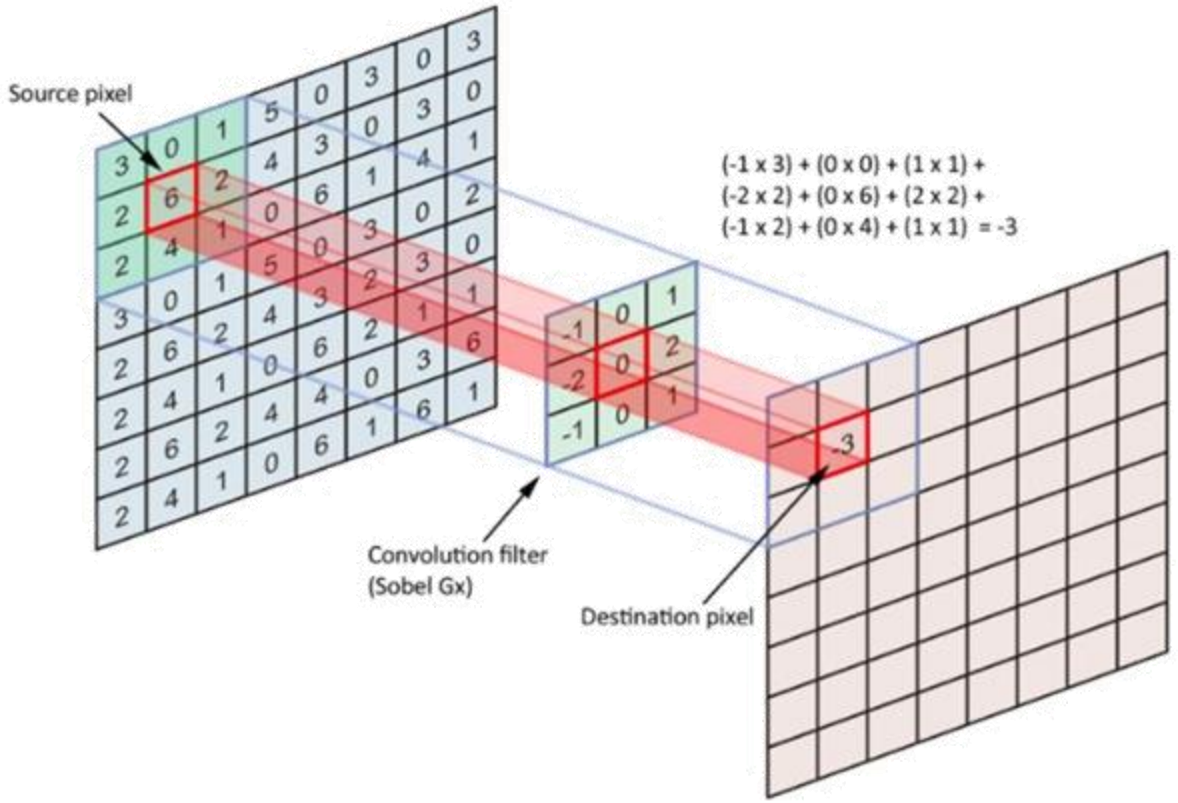
上海创智学院
Shanghai Innovation Institute



Convolution

上海创智学院
Shanghai Innovation Institute

- Convolve the filter with the image: slide over the image spatially and compute dot products



Parallelizing Convolutional Neural Networks

上海创智学院
Shanghai Innovation Institute

- Convolutional layers
  - 90-95% of the computation
  - 5% of the parameters
  - Very large intermediate activations
- Fully-connected layers
  - 5-10% of the computation
  - 95% of the parameters
  - Small intermediate activations
- **Discussion: how to parallelize CNNs?**

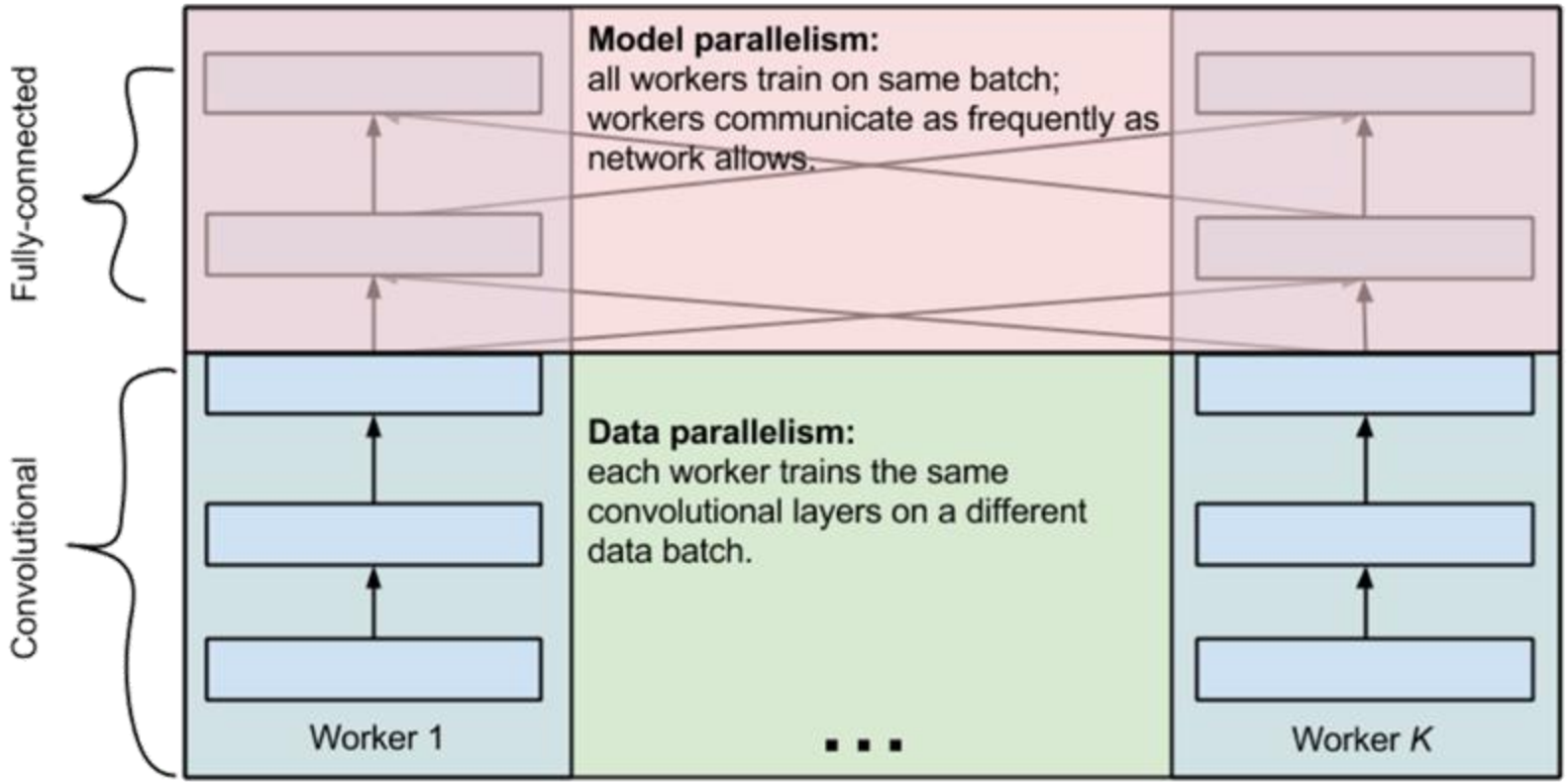
Data parallelism

Tensor model parallelism

Parallelizing Convolutional Neural Networks

上海创智学院
Shanghai Innovation Institute

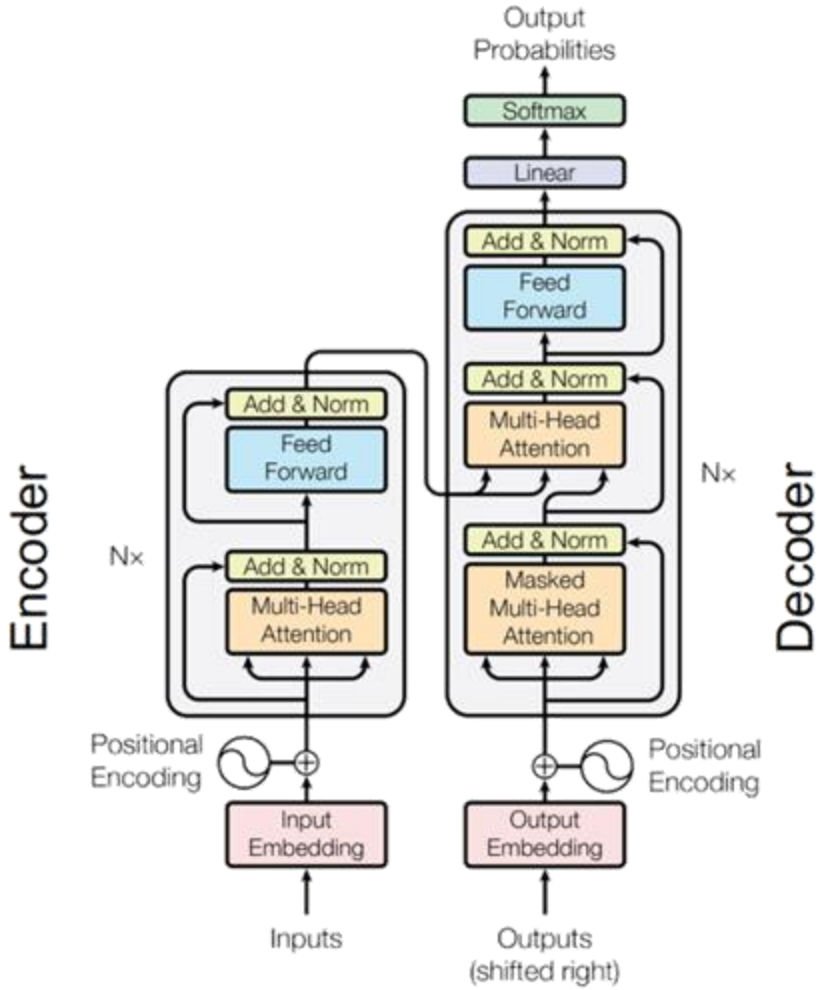
- Data parallelism for convolutional layers
- Tensor model parallelism for fully-connected layers

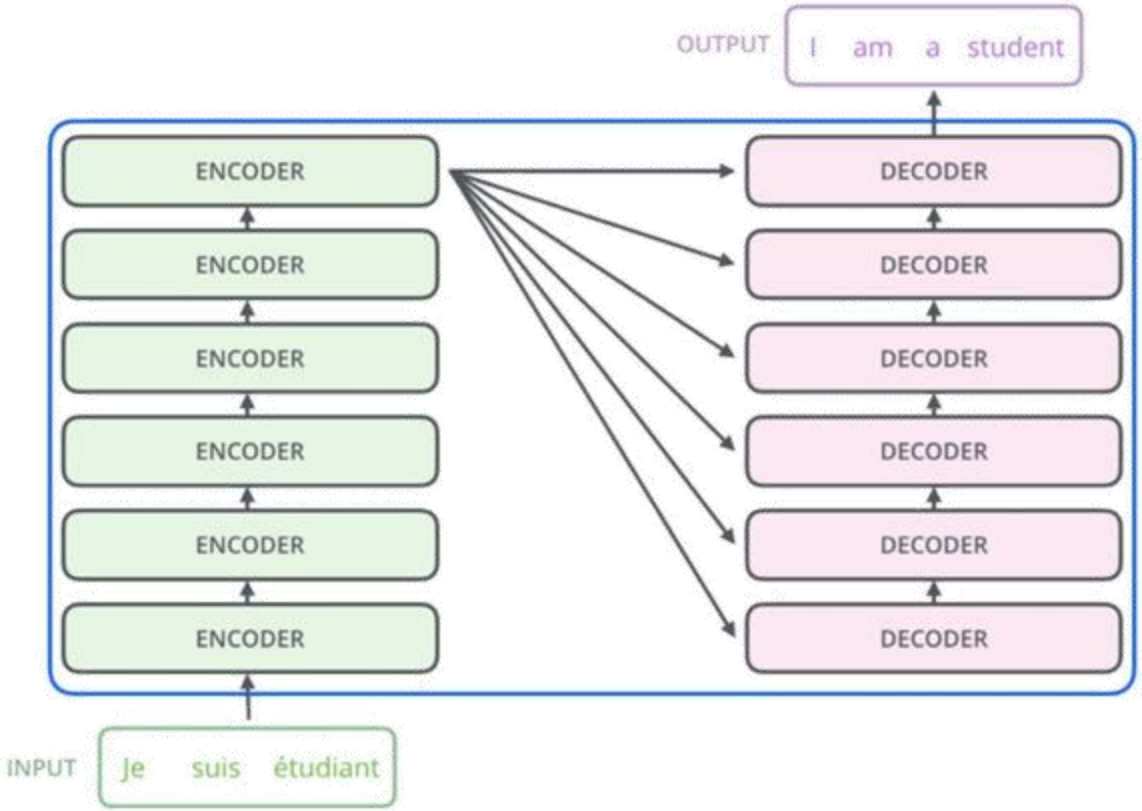


Example: Parallelizing Transformers

上海创智学院
Shanghai Innovation Institute

- Transformer: attention mechanism for language understanding





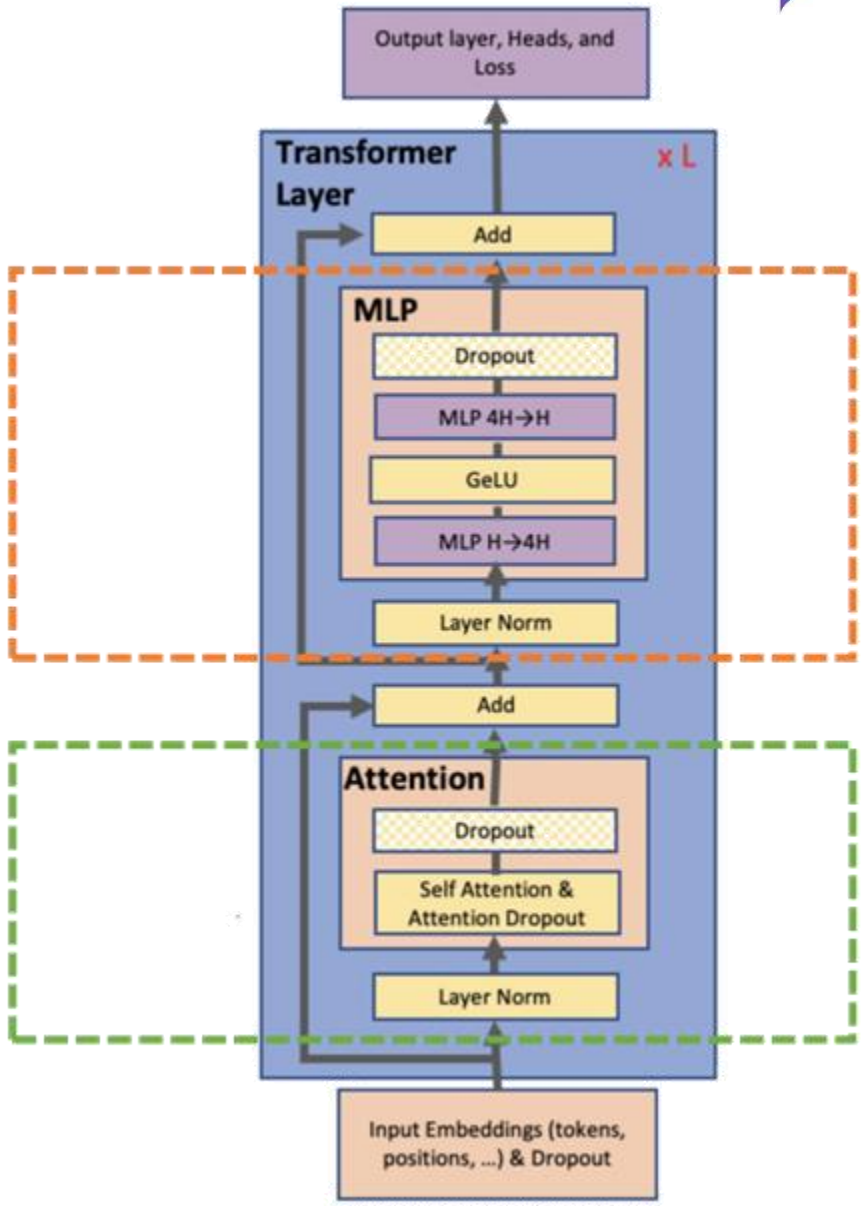
Ashish Vaswani et. al. Attention is all you need.

A Single Transformer Layer

上海创智学院
Shanghai Innovation Institute

Fully-Connected Layers

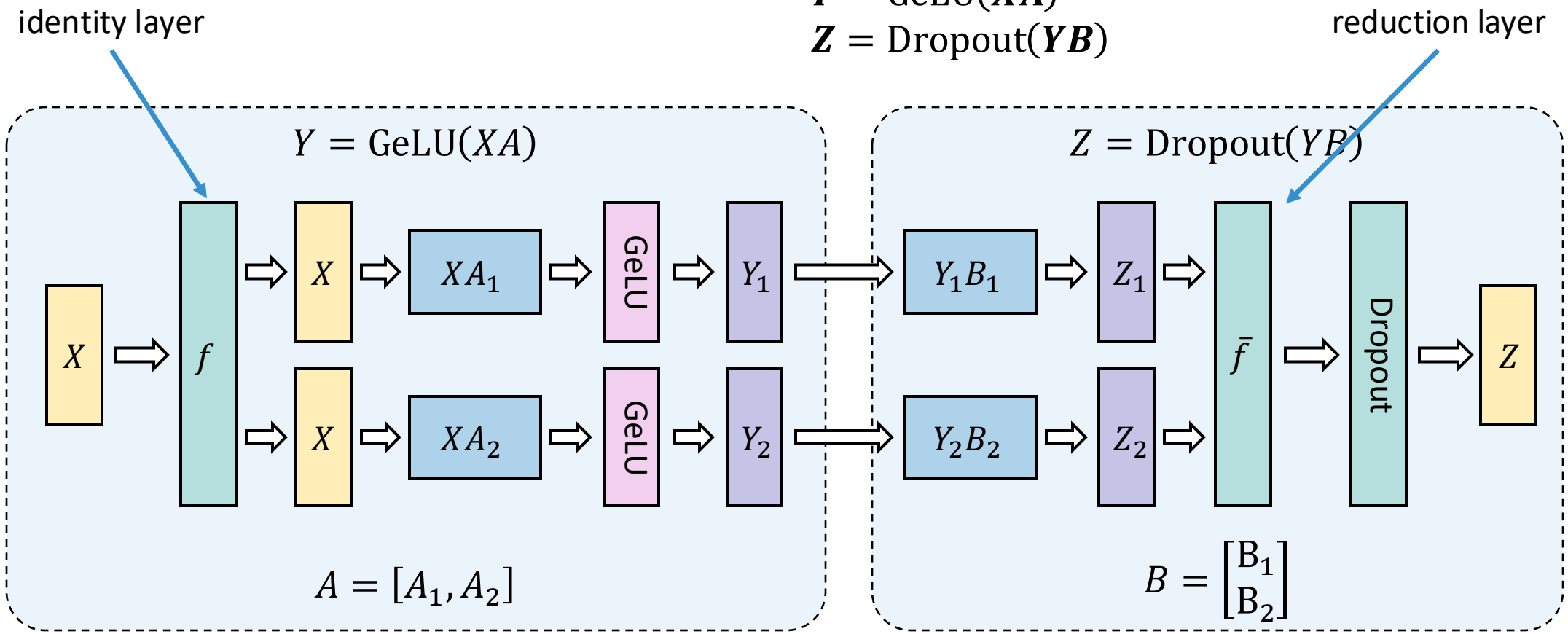
Self-Attention Layers



Parallelizing Fully-Connected Layers in Transformers

上海创智学院 Shanghai Innovation Institute

$$\boldsymbol{Y} = \mathrm{GeLU}(\boldsymbol{XA})$$
$$\boldsymbol{Z} = \mathrm{Dropout}(\boldsymbol{YB})$$

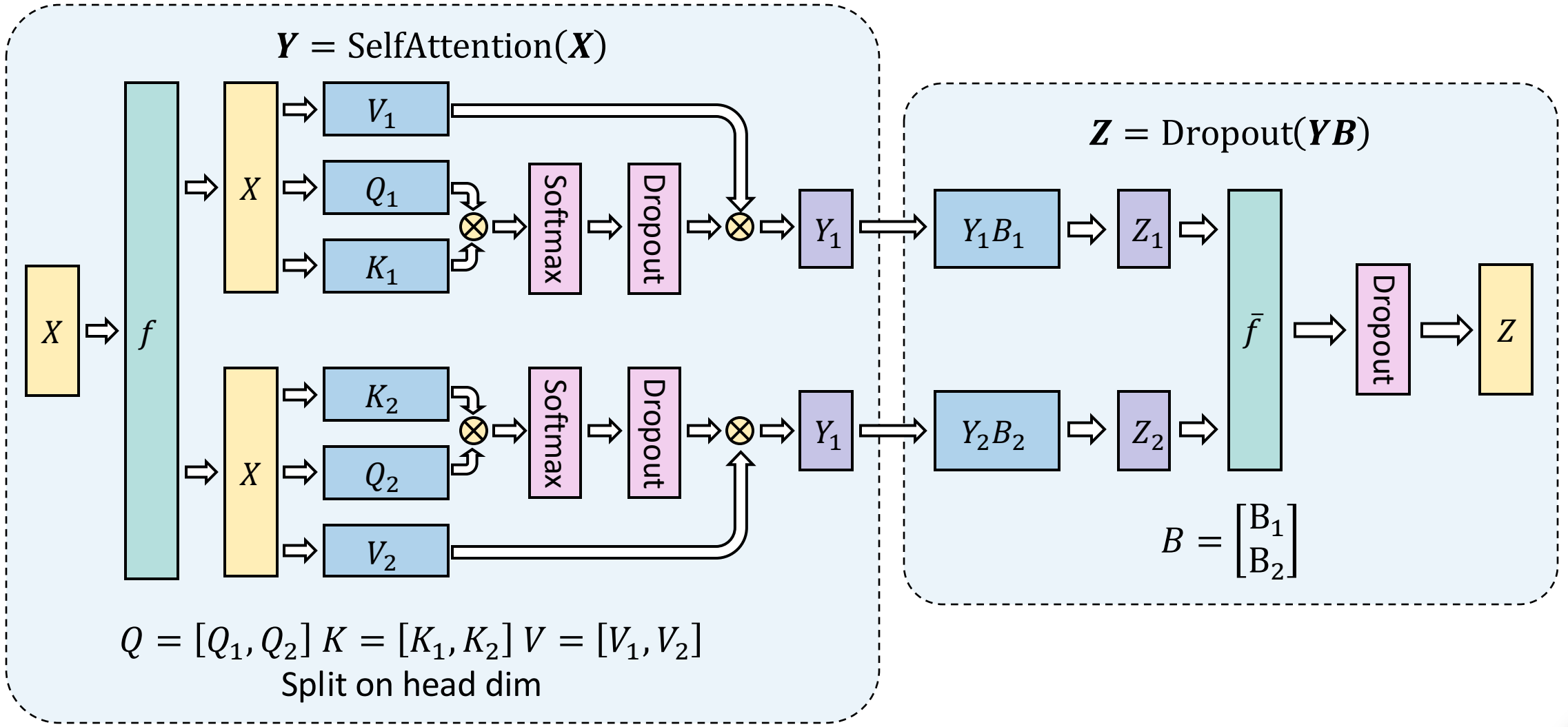


Tensor model parallelism
(partition output)

Tensor model parallelism
(reduce output)

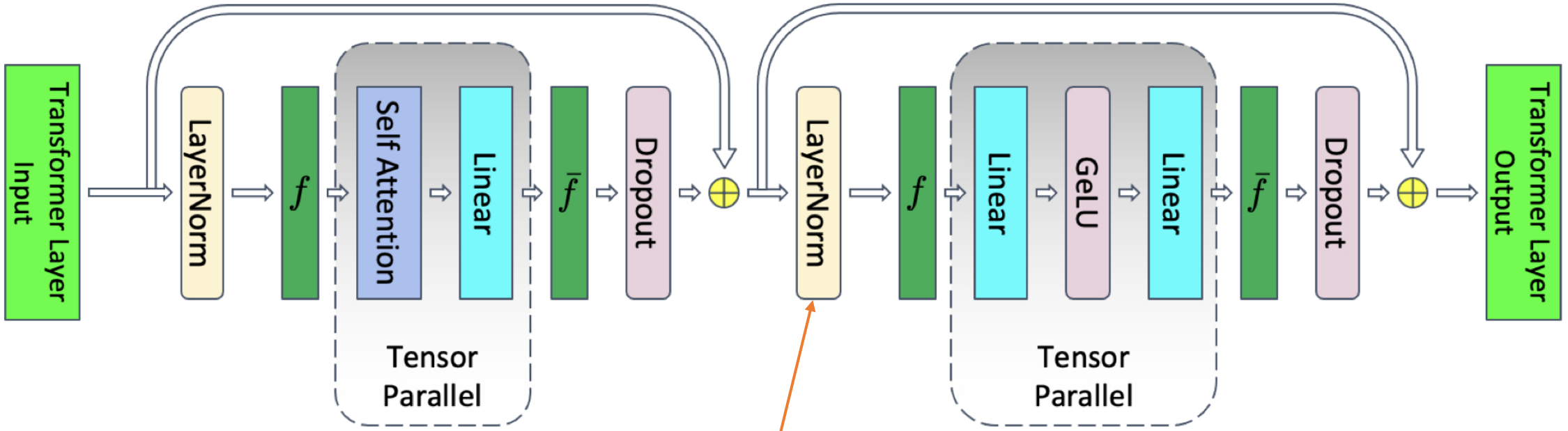
Megatron-LM: Training Multi-Billion Parameter Language Models Using Model Parallelism.

Parallelizing Self-Attn Layers in Transformers



Megatron-LM: Training Multi-Billion Parameter Language Models Using Model Parallelism.

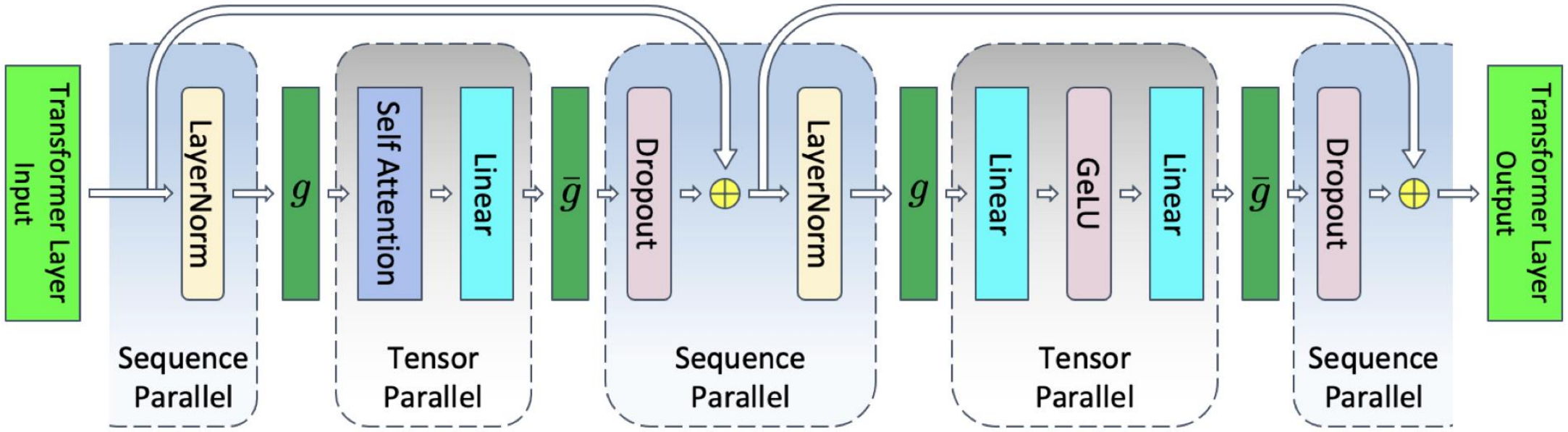
Tensor Parallelism for Transformer Layer



Where f is identity operation while $\bar{f}$ (g in prev slides) is all-reduce

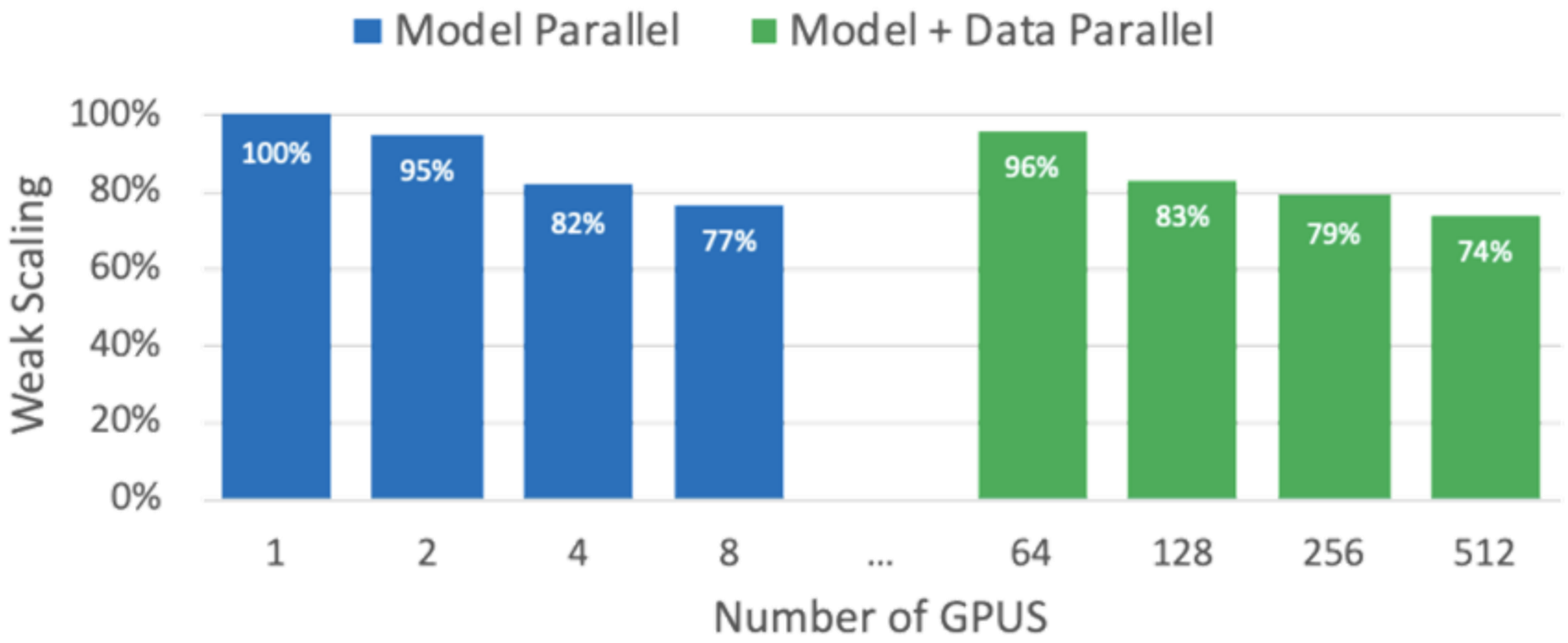
Norm is not parallelized

Sequence Parallelism



Where g is all-gather in forward pass while $\bar{g}$ is reduce-scatter

Parallelizing Transformers



Scale to 512 GPUs by combining data and model parallelism

Megatron-LM: Training Multi-Billion Parameter Language Models Using Model Parallelism.

How to parallelize DNN Training?

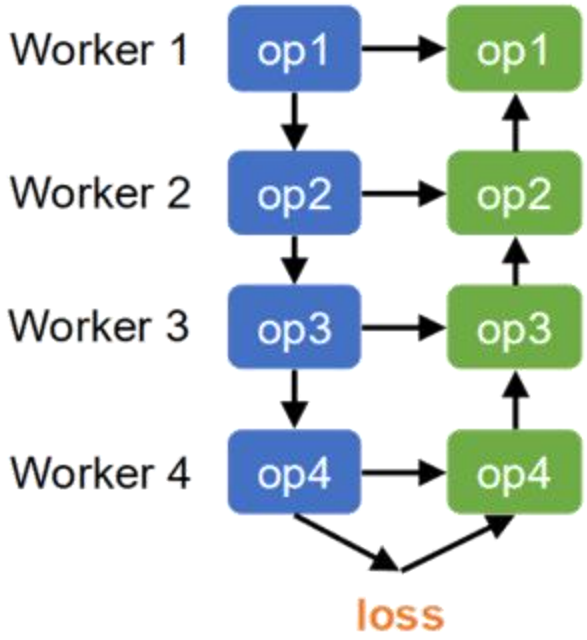
上海创智学院
Shanghai Innovation Institute

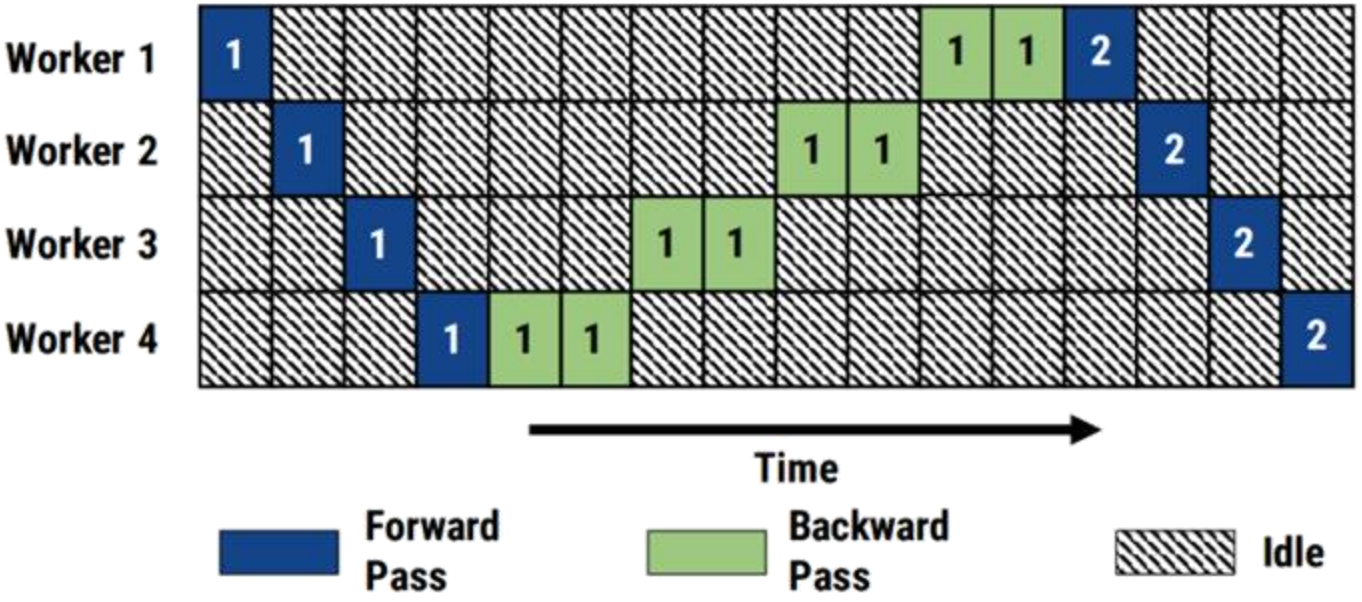
- Data parallelism
- Model parallelism
  - Tensor model parallelism
  - **Pipeline model parallelism**

An Issue with Model Parallelism

上海创智学院 Shanghai Innovation Institute

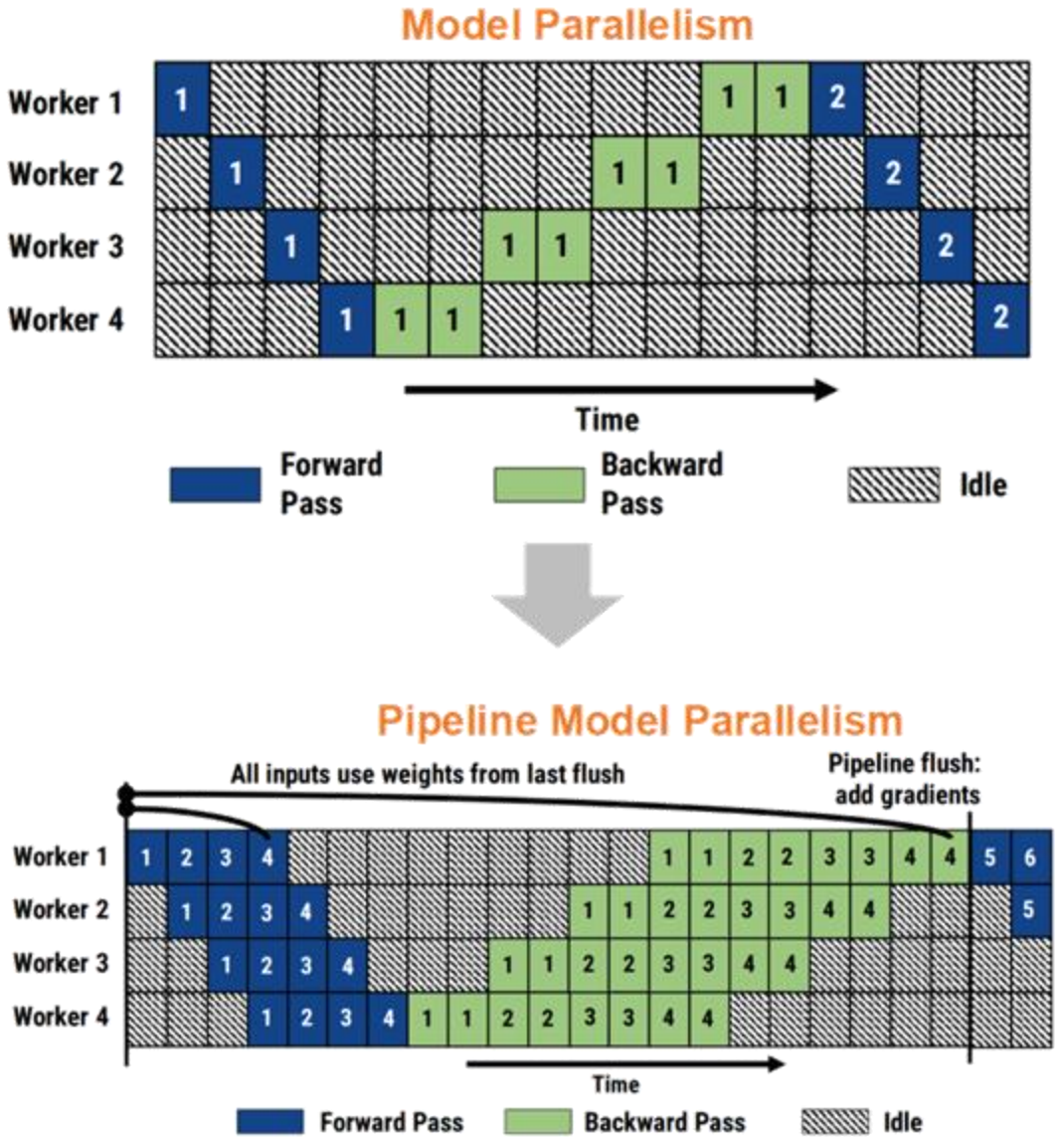
- Under-utilization of compute resources
- Low overall throughput due to resource utilization





Pipeline Model Parallelism

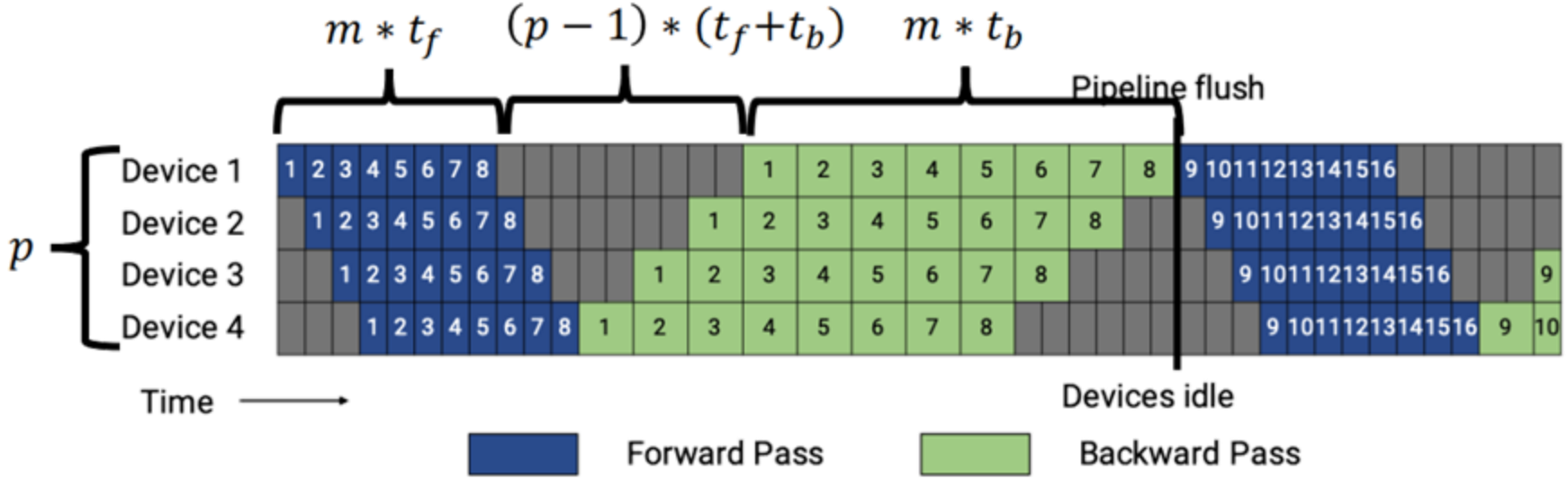
- **Mini-batch**: the number of samples processed in each iteration
- Divide a mini-batch into multiple **micro-batches**
- Pipeline the forward and backward computations across micro-batches



Pipeline Model Parallelism: Device Utilization

上海创智学院 Shanghai Innovation Institute

- m : micro-batches in a mini-batch
- p: number of pipeline stages
- All stages take t_f/ t_b to process a forward (backward) micro-batch

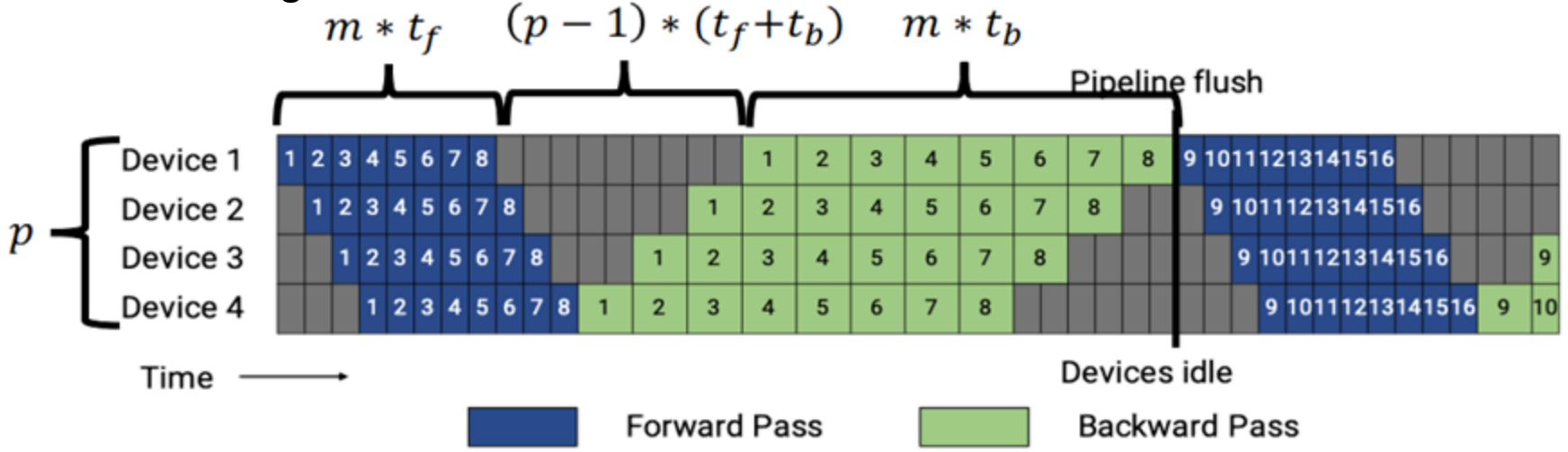


$$\boldsymbol{BubbleFraction} = \frac{(\boldsymbol{p}-\boldsymbol{1}) * (\boldsymbol{t_f}+\boldsymbol{t_b})}{\boldsymbol{m} * \boldsymbol{t_f} + \boldsymbol{m} * \boldsymbol{t_b}} = \frac{\boldsymbol{p}-\boldsymbol{1}}{\boldsymbol{m}}$$

GPipe: Efficient Training of Giant Neural Networks using Pipeline Parallelism

Improving Pipeline Parallelism Efficiency

- m : number of micro-batches in a mini-batch
  - Increase mini-batch size or reduce micro-batch size
  - Caveat: large mini-batch sizes can lead to accuracy loss; small micro-batch sizes reduce GPU utilization
- p: number of pipeline stages
  - Decrease pipeline depth
  - Caveat: increase stage size



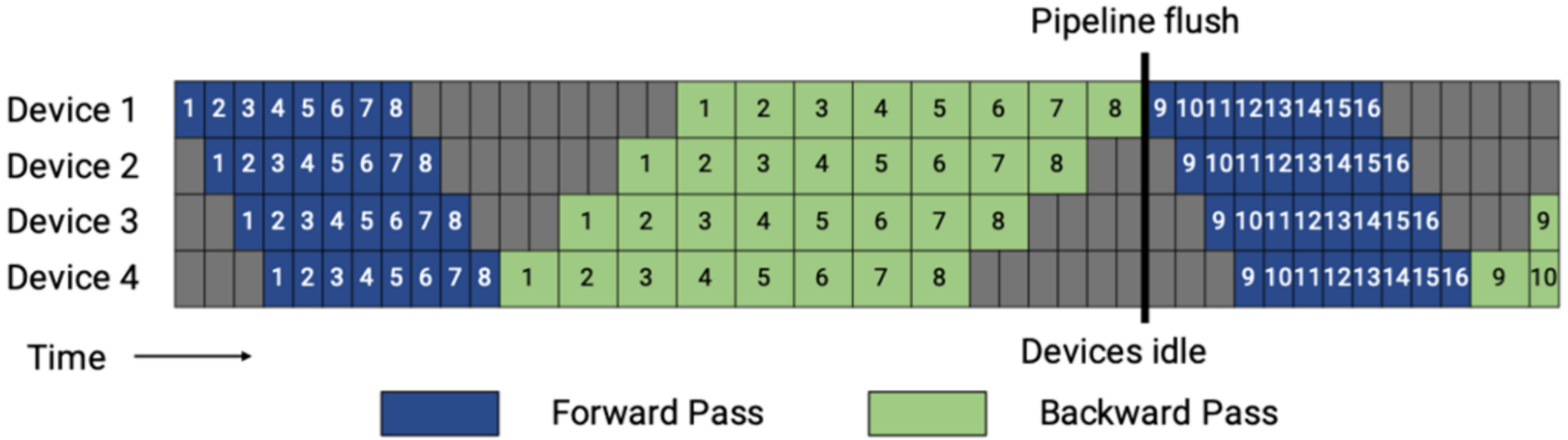
$$\boldsymbol{BubbleFraction} = \frac{(\boldsymbol{p}-\boldsymbol{1}) * (\boldsymbol{t_f}+\boldsymbol{t_b})}{\boldsymbol{m} * \boldsymbol{t_f} + \boldsymbol{m} * \boldsymbol{t_b}} = \frac{\boldsymbol{p}-\boldsymbol{1}}{\boldsymbol{m}}$$

GPipe: Efficient Training of Giant Neural Networks using Pipeline Parallelism

Pipeline Model Parallelism: Memory Requirement

上海创智学院 Shanghai Innovation Institute

- An issue: we need to keep the intermediate activations of **all microbatches** before back propagation



Can we improve the pipeline schedule to reduce memory requirement?

GPipe: Efficient Training of Giant Neural Networks using Pipeline Parallelism

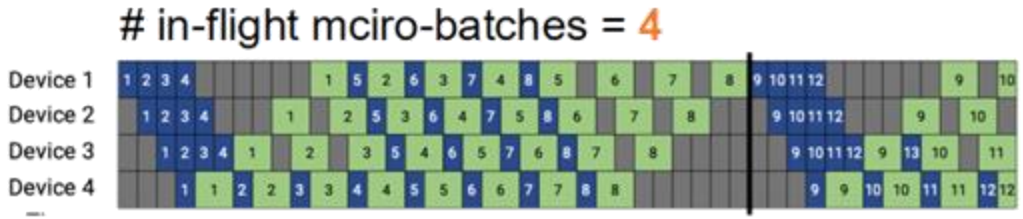
Pipeline Parallelism with 1F1B Schedule

上海创智学院
Shanghai Innovation Institute

- One-Forward-One-Backward in the steady state
- Limit the number of in-flight micro-batches to the pipeline depth
- **Reduce memory footprint of pipeline parallelism**
- **Doesn't reduce pipeline bubble**

Can we reduce pipeline bubble?

Pipeline parallelism with GPipe's schedule



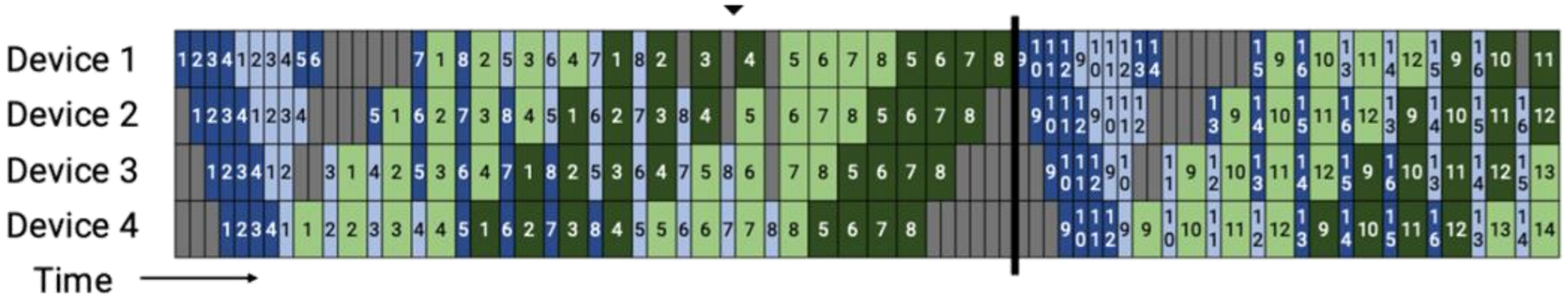
Pipeline parallelism with 1F1B schedule

GPipe: Efficient Training of Giant Neural Networks using Pipeline Parallelism

Pipeline Parallelism with Interleaved 1F1B Schedule

上海创智学院 Shanghai Innovation Institute

- Further divide each stage into v sub-stages
- The forward (backward) time of each sub-stage is $\frac{t_f}{v}\left(\frac{t_b}{v}\right)$



Each device is assigned two chunks. Dark colors show the first chunk and light colors show the second chunk.

$$\boldsymbol{BubbleFraction} = \frac{(\boldsymbol{p}-\boldsymbol{1}) * \frac{(\boldsymbol{t_f}+\boldsymbol{t_b})}{\boldsymbol{v}}}{\boldsymbol{m} * \boldsymbol{t_f} + \boldsymbol{m} * \boldsymbol{t_b}} = \frac{\boldsymbol{1}}{\boldsymbol{v}} * \frac{\boldsymbol{p}-\boldsymbol{1}}{\boldsymbol{m}}$$

Reduce bubble time at the cost increased communication

Pipeline Parallelism with Interleaved 1F1B Schedule

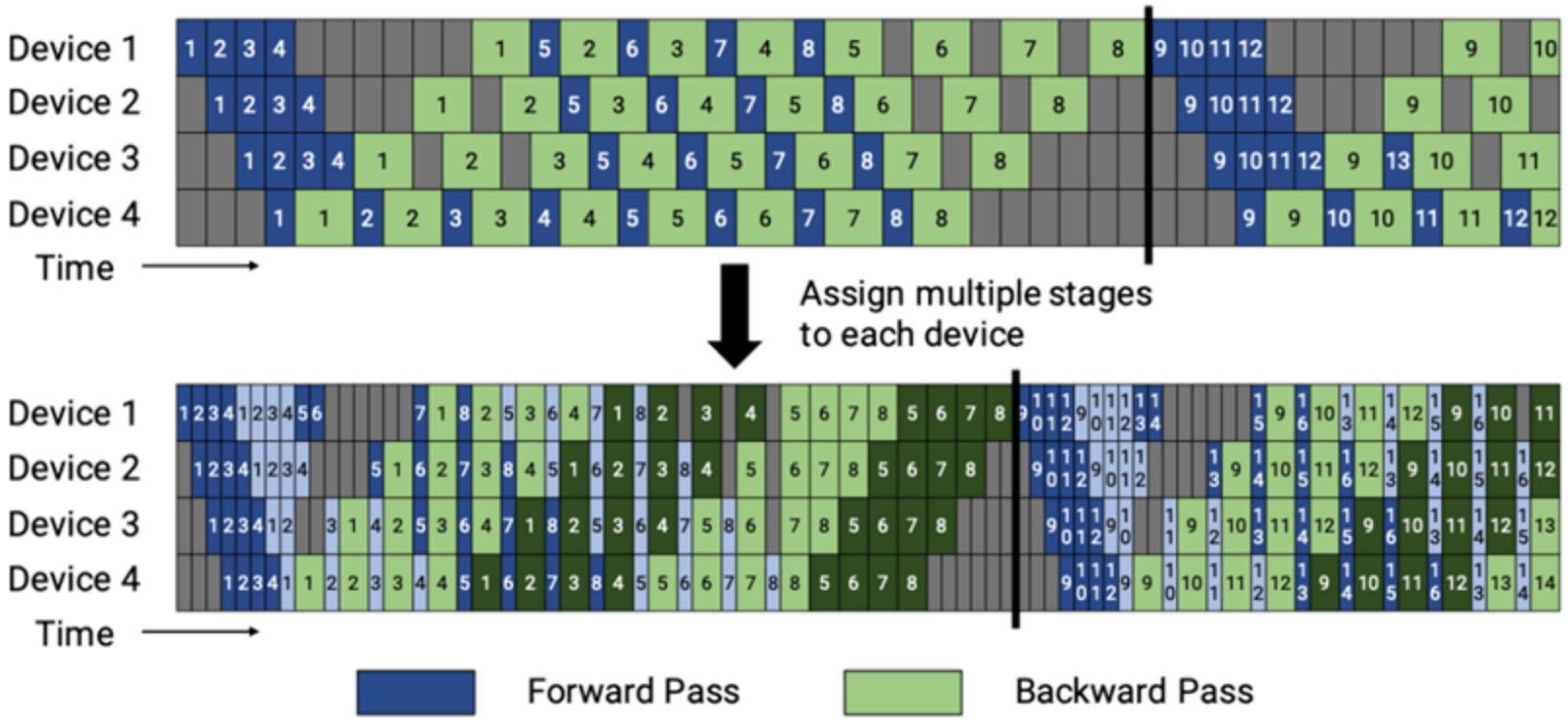
上海创智学院
Shanghai Innovation Institute

Pipeline parallelism with 1F1B Schedule

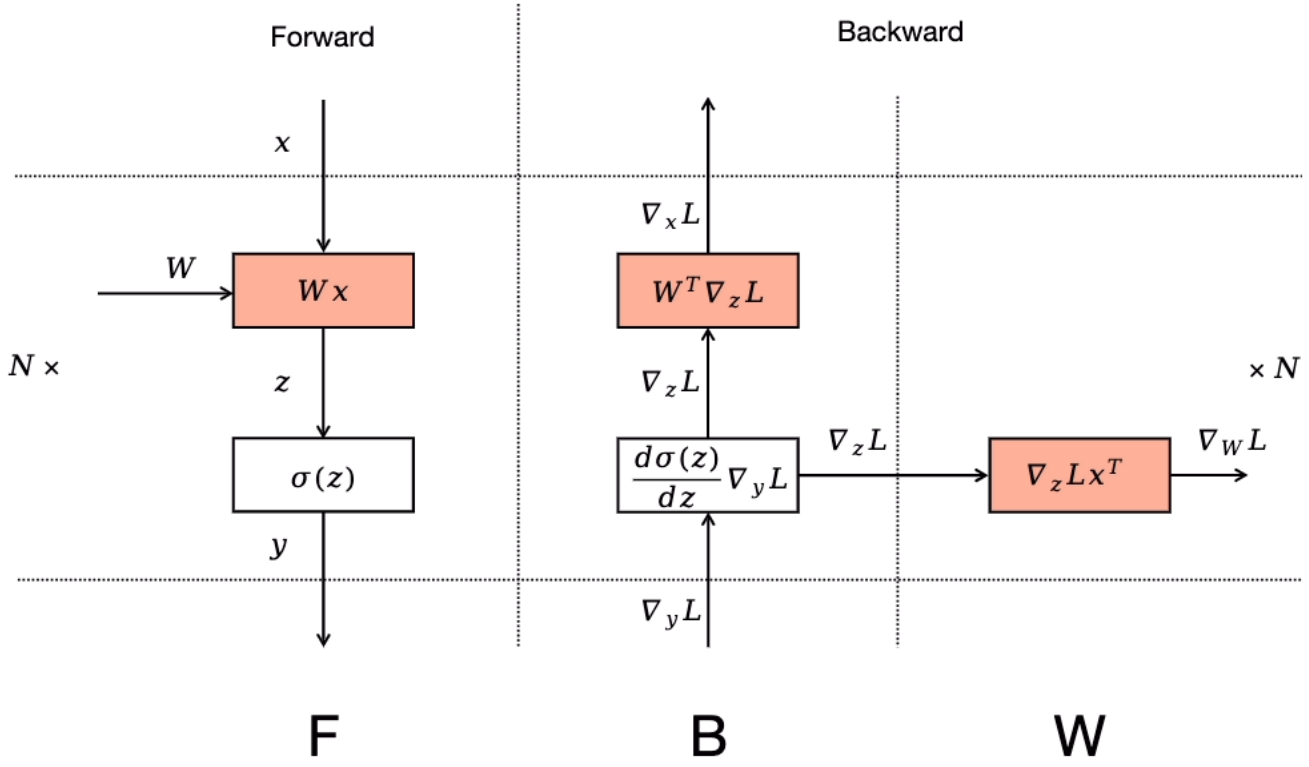
$$BubbleFraction = \frac{p-1}{m}$$

Pipeline parallelism with interleaved 1F1B Schedule

$$BubbleFraction = \frac{1}{v} * \frac{p-1}{m}$$



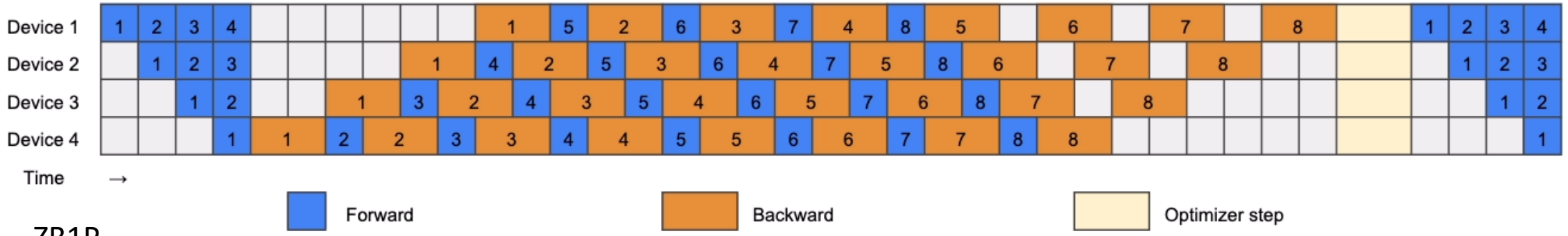
Zero Bubble Pipeline Parallelism



Zero Bubble Pipeline Parallelism

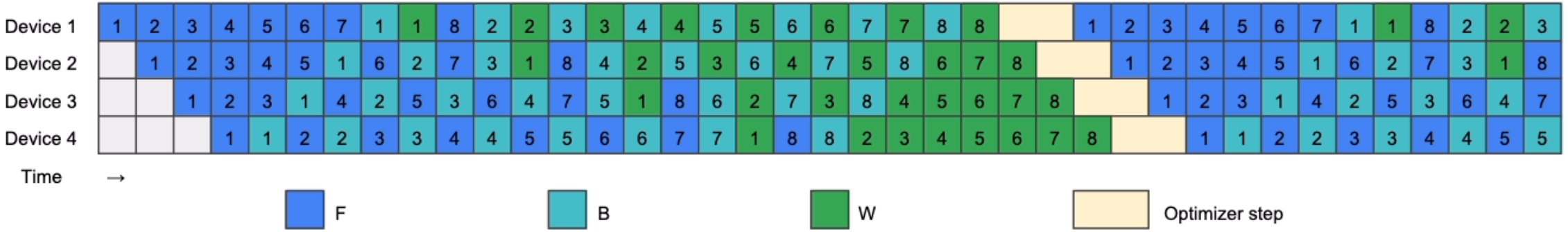
上海创智学院
Shanghai Innovation Institute

1F1B



ZB1P

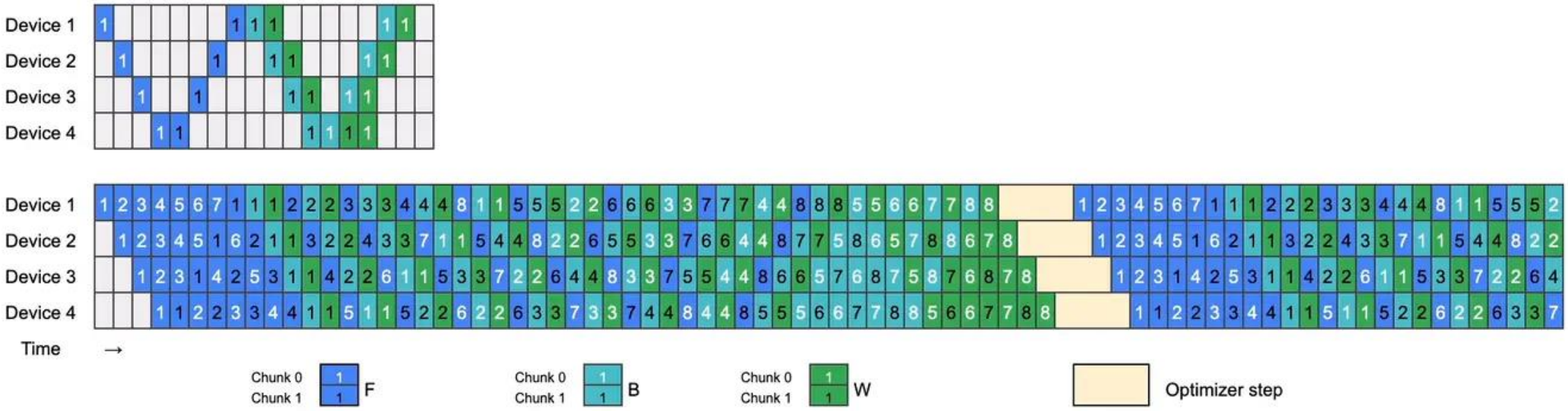
ZB2P



Zero Bubble Pipeline Parallelism

上海创智学院
Shanghai Innovation Institute

ZBV



Combine Zero Bubble and Interleaved 1F1B

Zero Bubble Summary

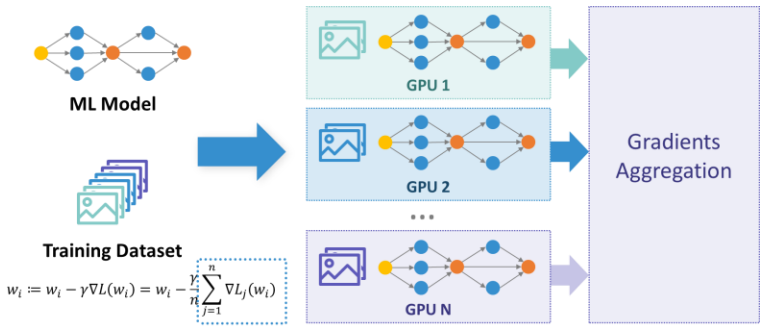
上海创智学院
Shanghai Innovation Institute

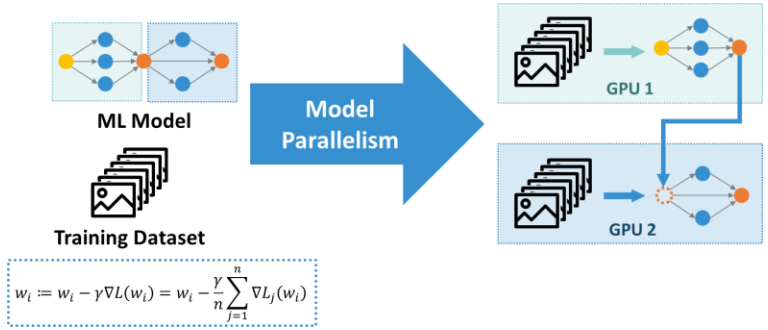
	1F1B	ZB1P	ZB2P	ZBV
Bubble Rate	$\frac{p-1}{m+p-1} = B$	$\frac{B}{3}$	0	0
Activation Memory (Compared to 1F1B)	1x	1x	2x	1x
Pipeline Communication Volume (Compared to 1F1B)	1x	1x	1x	2x

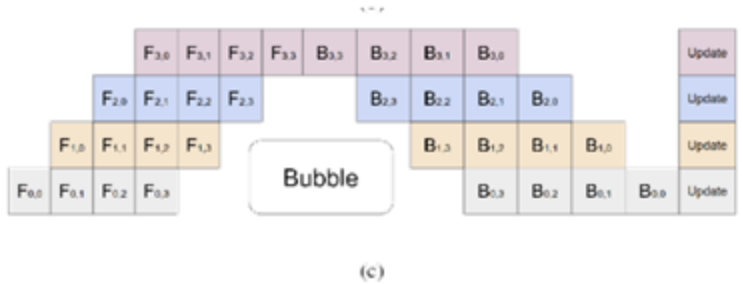
- p: number of pipeline stages
- m: number of microbatches
- Assuming $T_F = TB = TW$

Summary: Comparing Data/Tensor Model/Pipeline Model Parallelism

上海创智学院
Shanghai Innovation Institute



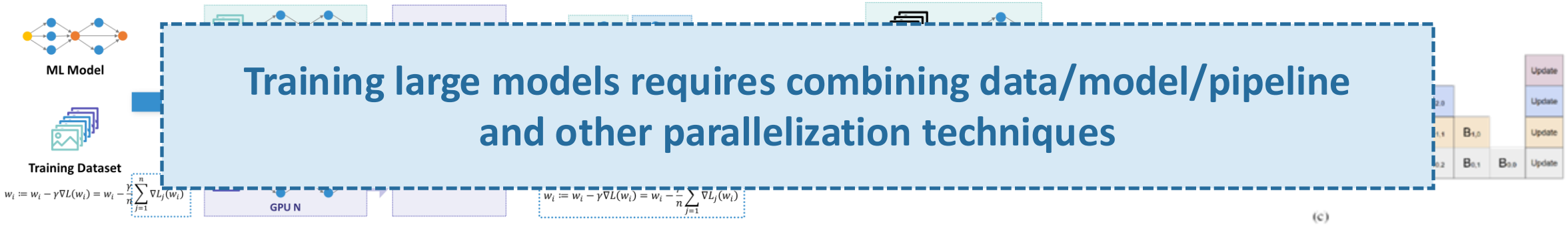




	Data Parallelism	Tensor Model Parallelism	Pipeline Model Parallelism
Pros	◆ Massively parallelizable ◆ Require no communication during forward/backward	◆ Support training large models ◆ Efficient for models with large numbers of parameters	◆ Support large-batch training ◆ Efficient for deep models
Cons	■ Do not work for models that cannot fit on a GPU ■ Do not scale for models with large numbers of parameters	■ Limited parallelizability; cannot scale to large numbers of GPUs ■ Need to transfer intermediate results in forward/backward	■ Limited utilization: bubbles in forward/backward

Summary: Comparing Data/Tensor Model/Pipeline Model Parallelism

上海创智学院
Shanghai Innovation Institute



Training large models requires combining data/model/pipeline and other parallelization techniques

	Data Parallelism	Tensor Model Parallelism	Pipeline Model Parallelism
Pros	◆ Massively parallelizable ◆ Require no communication during forward/backward	◆ Support training large models ◆ Efficient for models with large numbers of parameters	◆ Support large-batch training ◆ Efficient for deep models
Cons	■ Do not work for models that cannot fit on a GPU ■ Do not scale for models with large numbers of parameters	■ Limited parallelizability; cannot scale to large numbers of GPUs ■ Need to transfer intermediate results in forward/backward	■ Limited utilization: bubbles in forward/backward

Acknowledgement

The development of this course, including its structure, content, and accompanying presentation slides, has been significantly influenced and inspired by the excellent work of instructors and institutions who have shared their materials openly. We wish to extend our sincere acknowledgement and gratitude to the following courses, which served as invaluable references and a source of pedagogical inspiration:

- Machine Learning Systems[15-442/15-642], by **Tianqi Chen** and **Zhihao Jia** at **CMU**.
- Advanced Topics in Machine Learning (Systems)[CS6216], by **Yao Lu** at **NUS**

While these materials provided a foundational blueprint and a wealth of insightful examples, all content herein has been adapted, modified, and curated to meet the specific learning objectives of our curriculum. Any errors, omissions, or shortcomings found in these course materials are entirely our own responsibility. We are profoundly grateful for the contributions of the educators listed above, whose dedication to teaching and knowledge-sharing has made the creation of this course possible.

System for Artificial Intelligence

Thanks

Siyuan Feng💩💩💩
Shanghai Innovation Institute