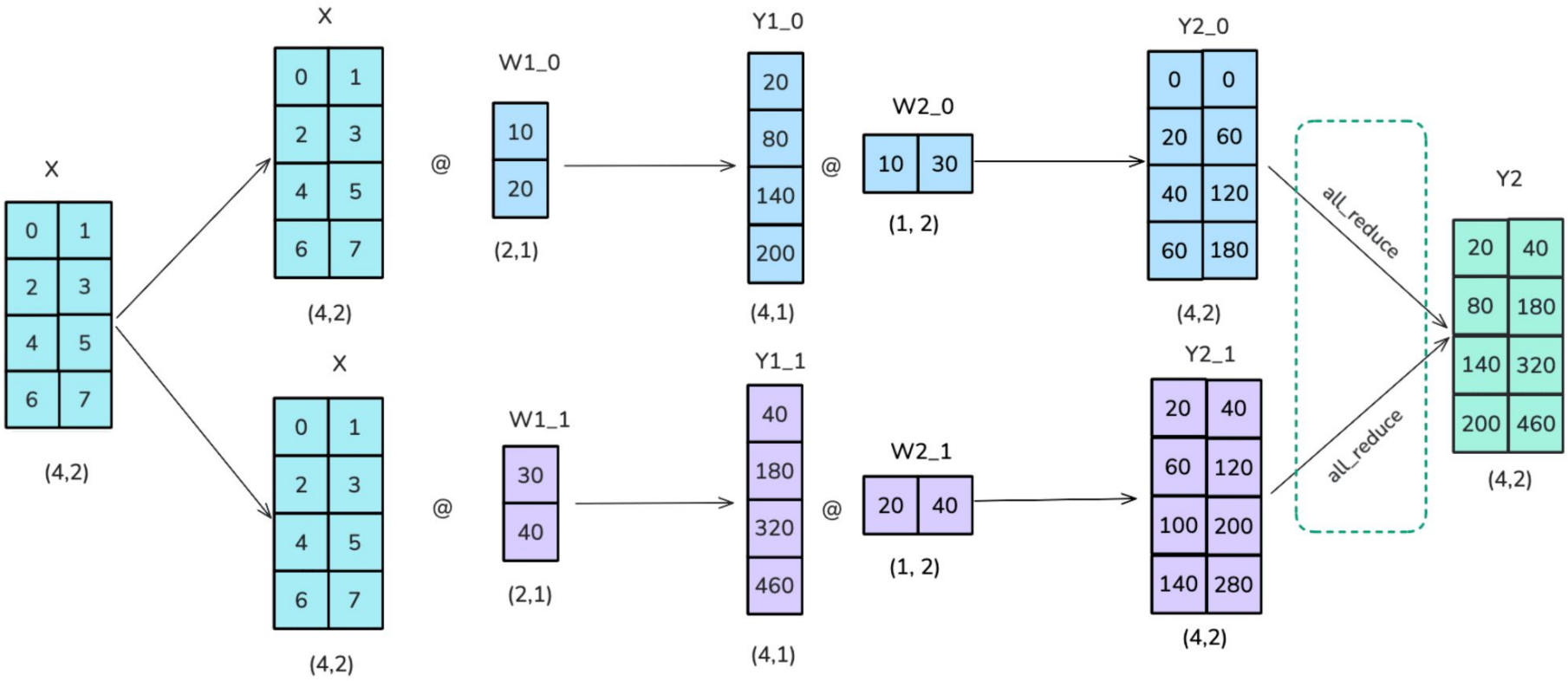
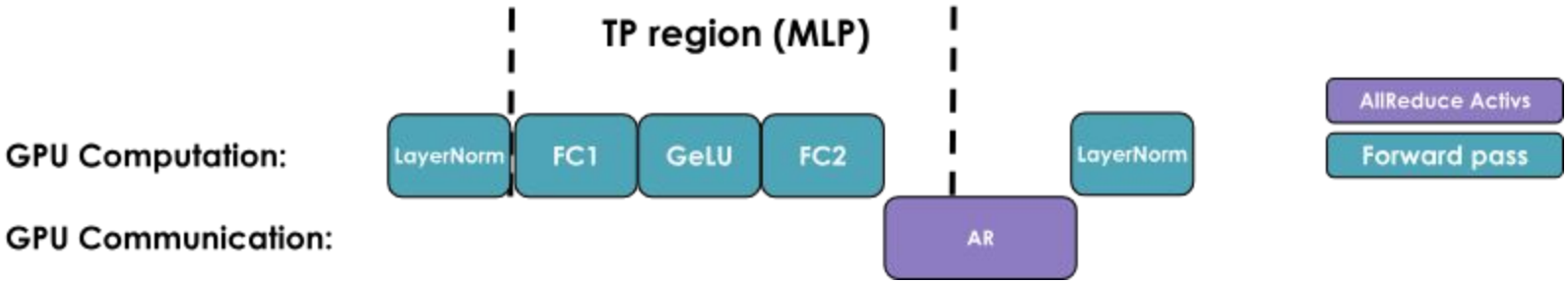
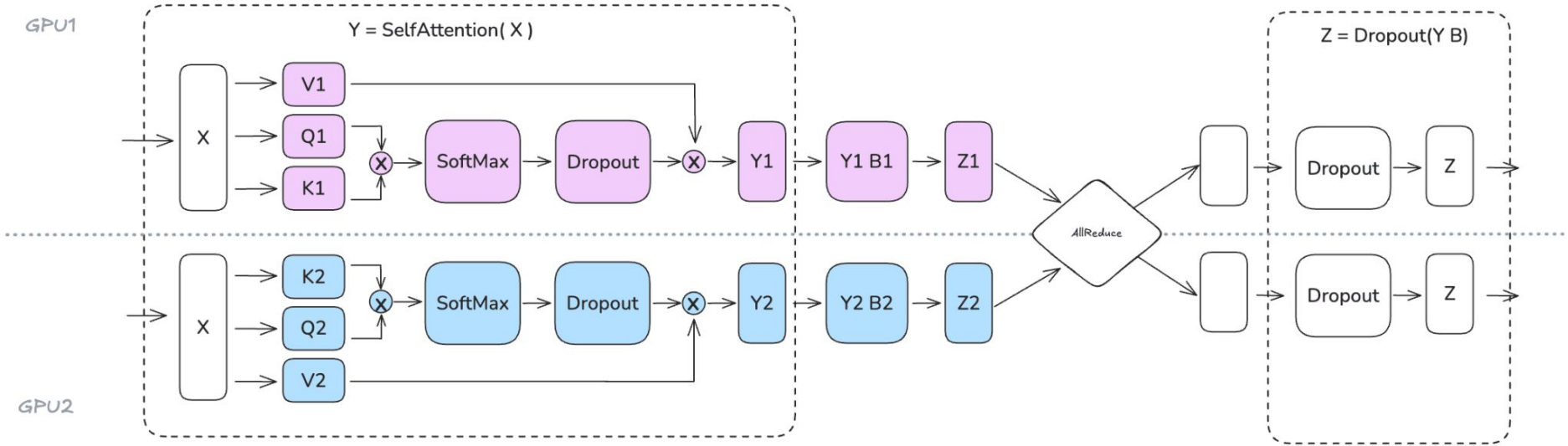
Sequence Parallelism

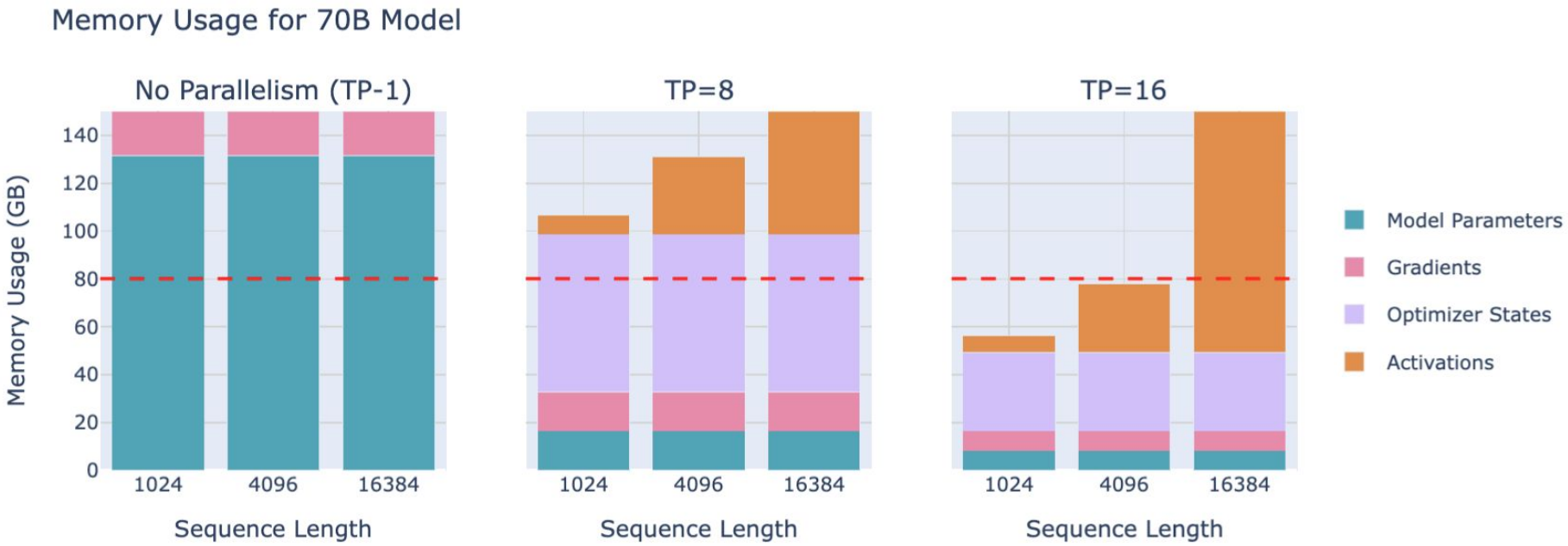
made with ❤️ for “Little ML book club”



Tensor parallelism with column linear + row Linear

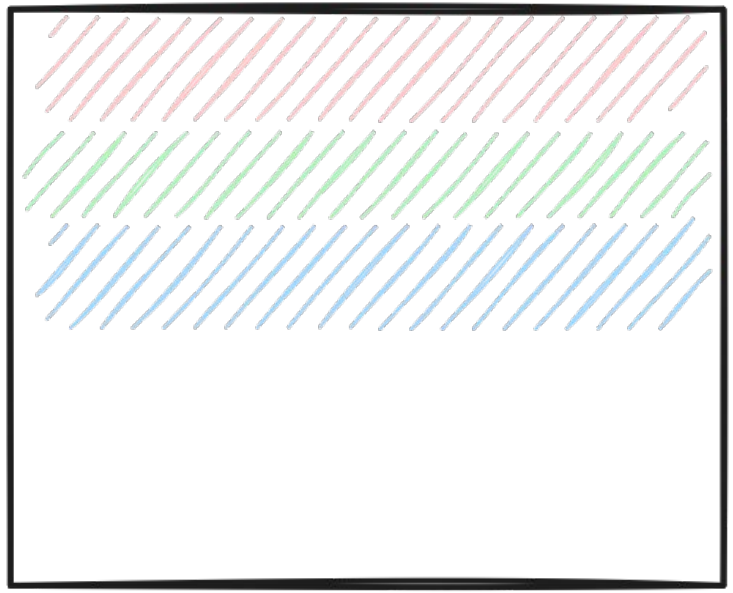

TP region (MLP)
GPU Computation:
GPU Communication:
LayerNorm
FC1
GeLU
FC2
AR
LayerNorm
AllReduce Activs
Forward pass

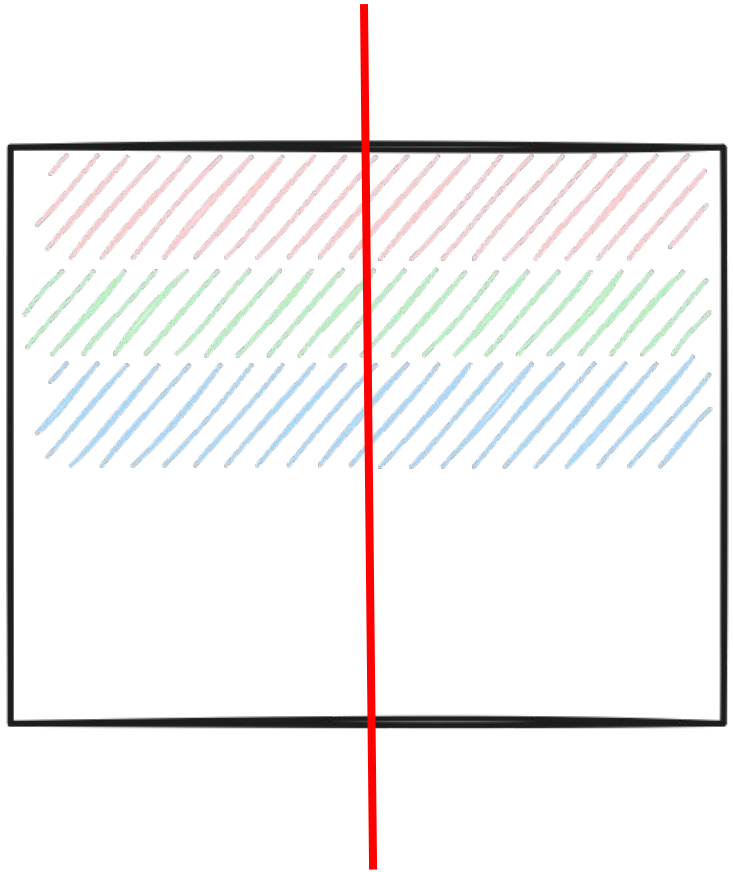

GPU1
Y = SelfAttention(X)
X
V1
Q1
K1
SoftMax
Dropout
Y1
Y1 B1
Z1
AllReduce
Z = Dropout(Y B)
Dropout
Z
GPU2
X
K2
Q2
V2
SoftMax
Dropout
Y2
Y2 B2
Z2
Dropout
Z

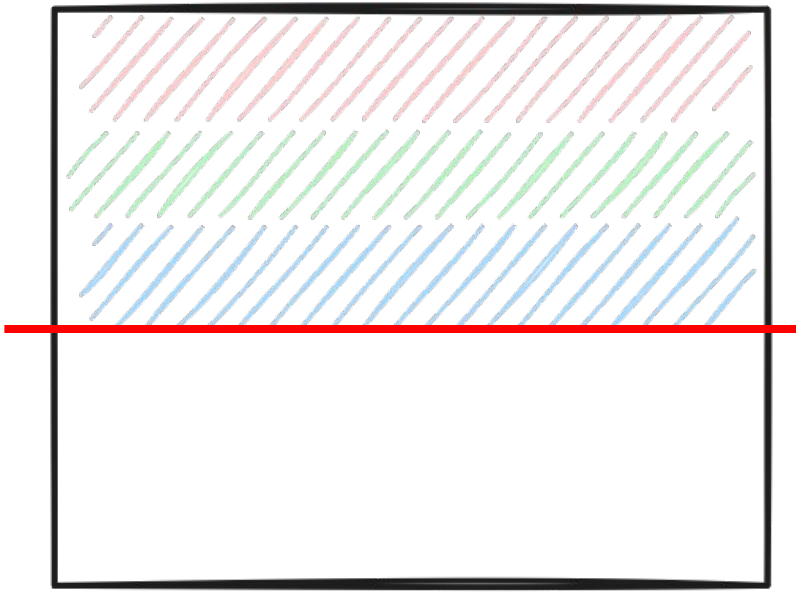

Memory Usage for 70B Model
No Parallelism (TP-1)
TP=8
TP=16
Memory Usage (GB)
0
20
40
60
80
100
120
140
1024
4096
16384
Sequence Length
Model Parameters
Gradients
Optimizer States
Activations

- layer norm
- dropout

- layer norm
- dropout ???

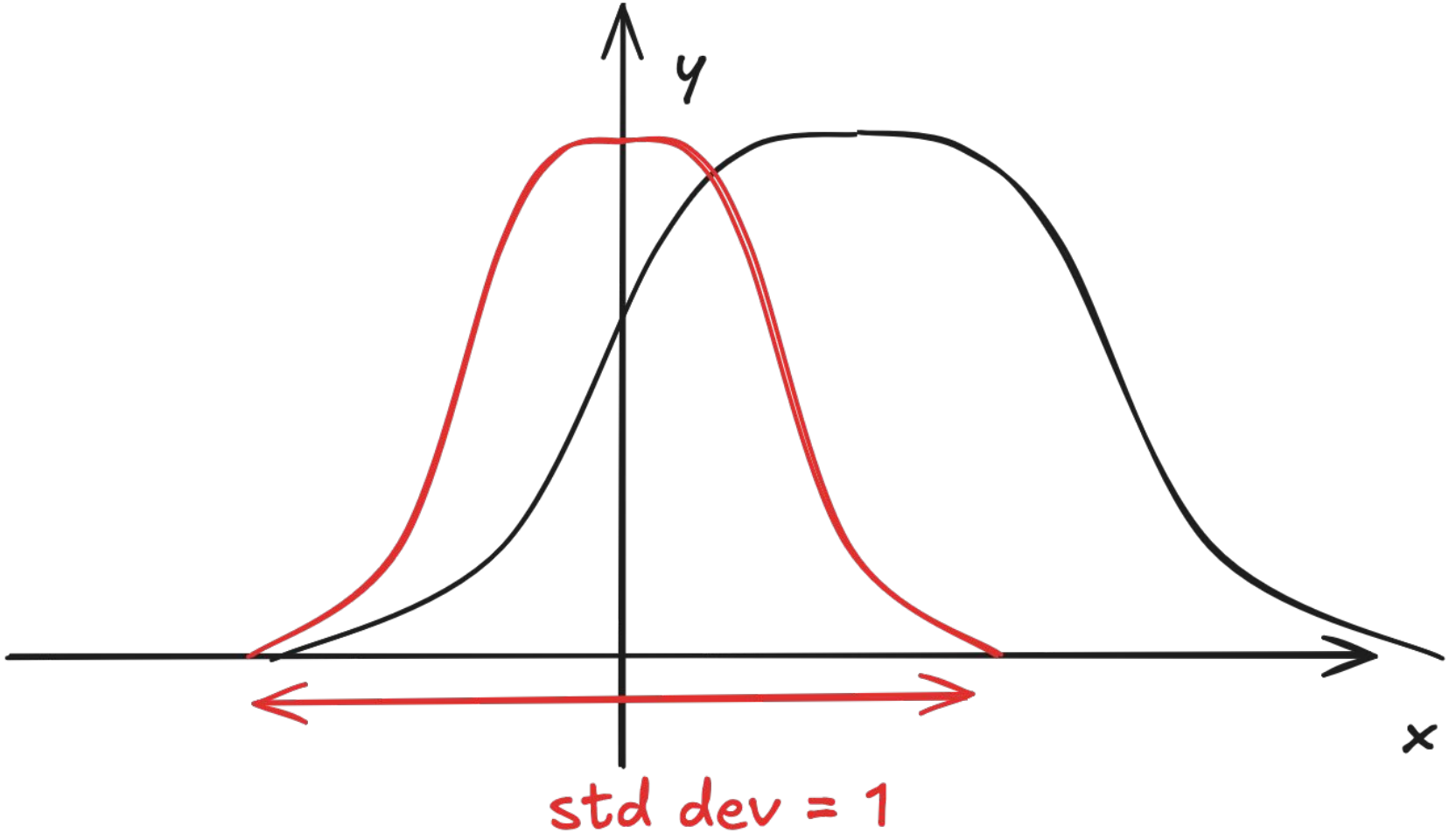




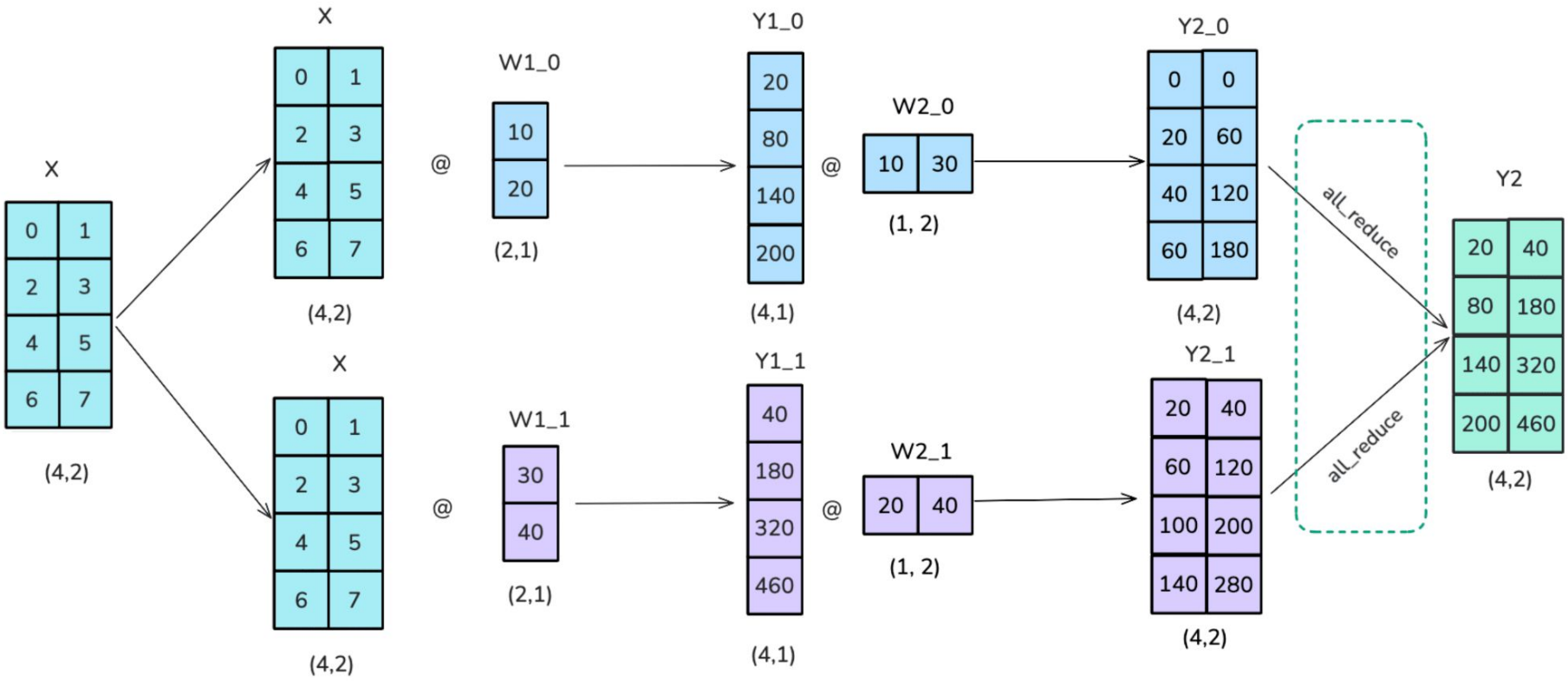


$$\text{LayerNorm}(x) = \gamma \cdot \frac{x - \mu}{\sqrt{\sigma^2 + \epsilon}} + \beta$$

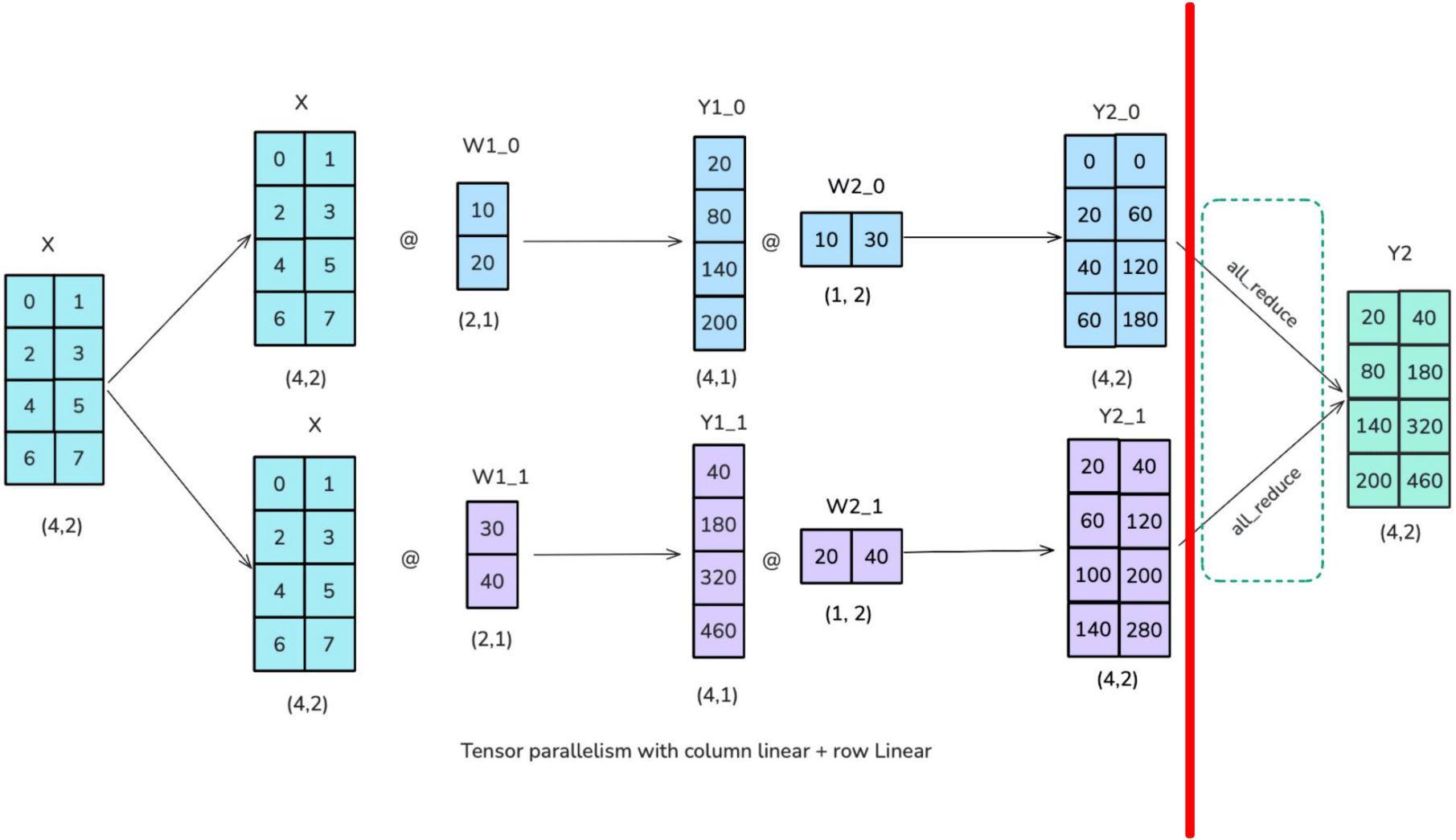
where $\mu = \text{mean}(x)$ and $\sigma^2 = \text{var}(x)$ are computed across hidden dimension h.


y
x
std dev = 1


(batch, seq_len, d_model)



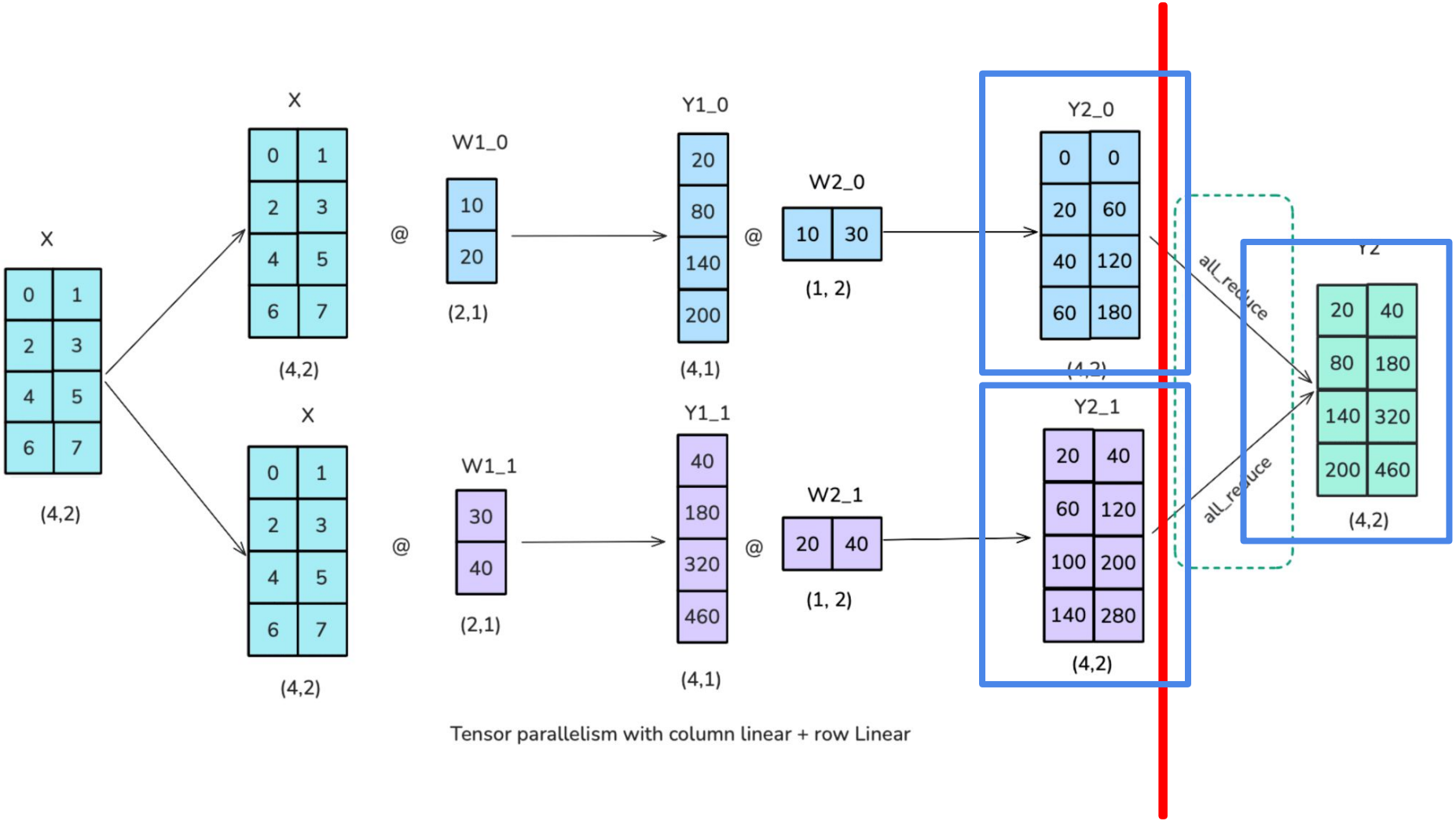
Tensor parallelism with column linear + row Linear



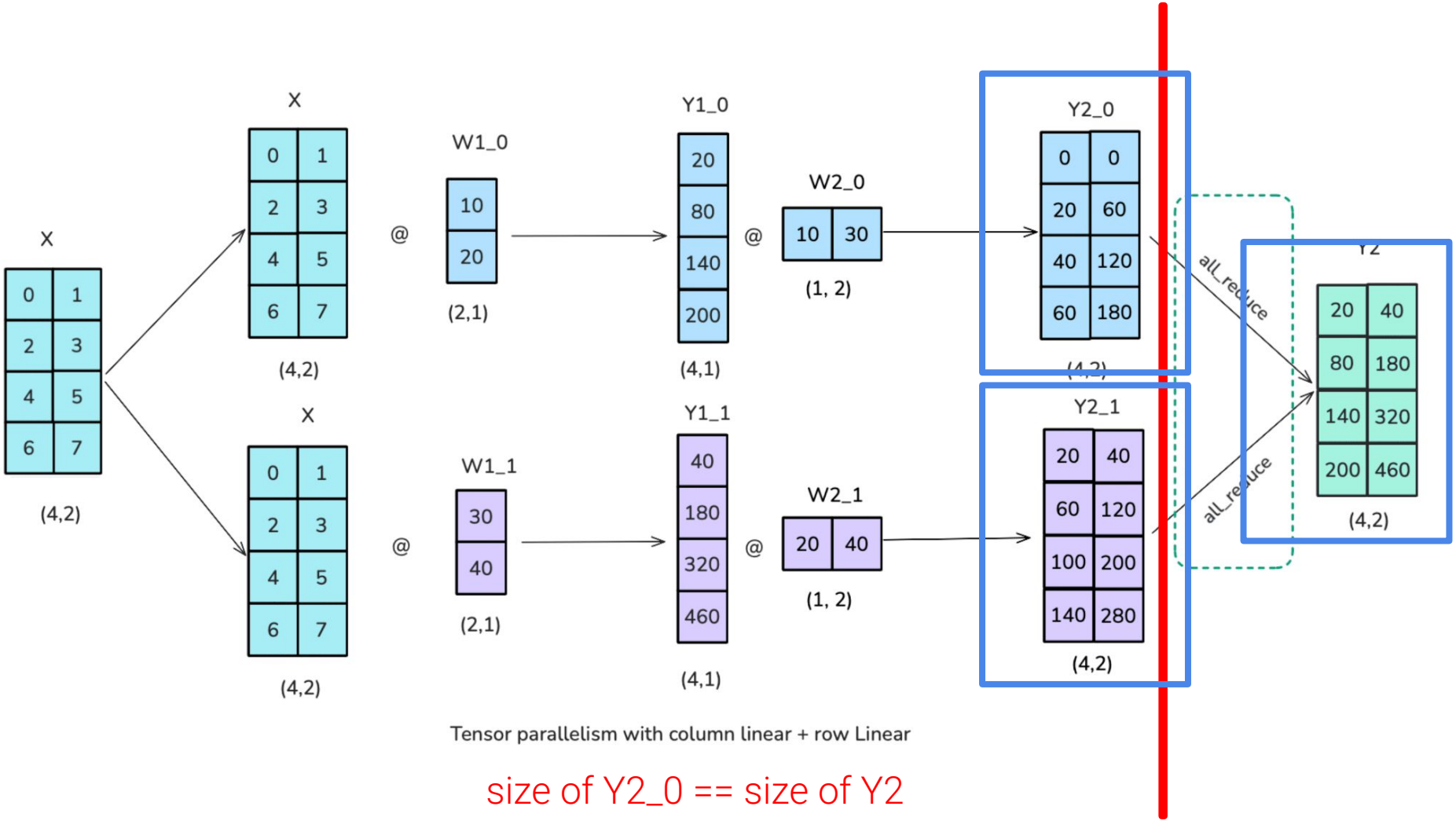
Tensor parallelism with column linear + row Linear


Full tensor: (batch, seq_len, d_model)
↑
SHARDED across GPUs
GPU 0: x[:, :, 0:D//2] # all tokens, first half of features
GPU 1: x[:, :, D//2:D] # all tokens, second half of features

hmmm....



Tensor parallelism with column linear + row Linear

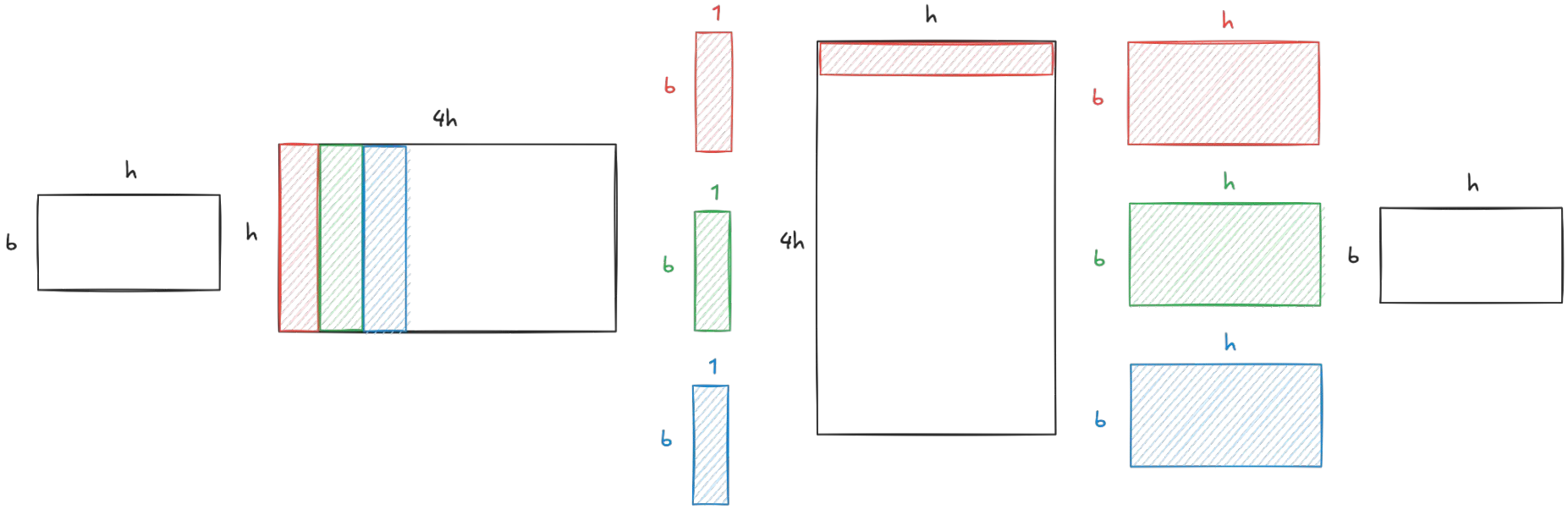


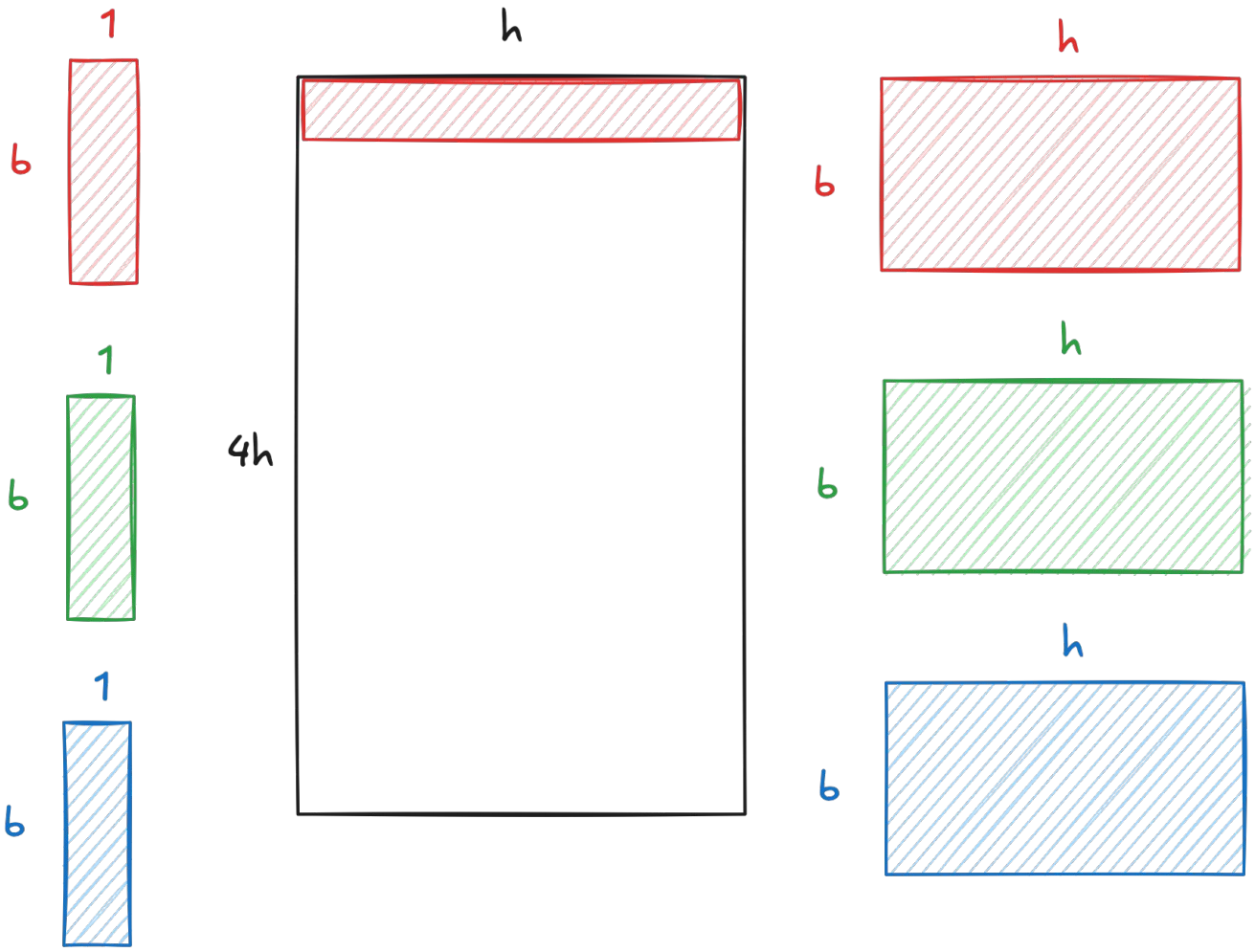
Tensor parallelism with column linear + row Linear

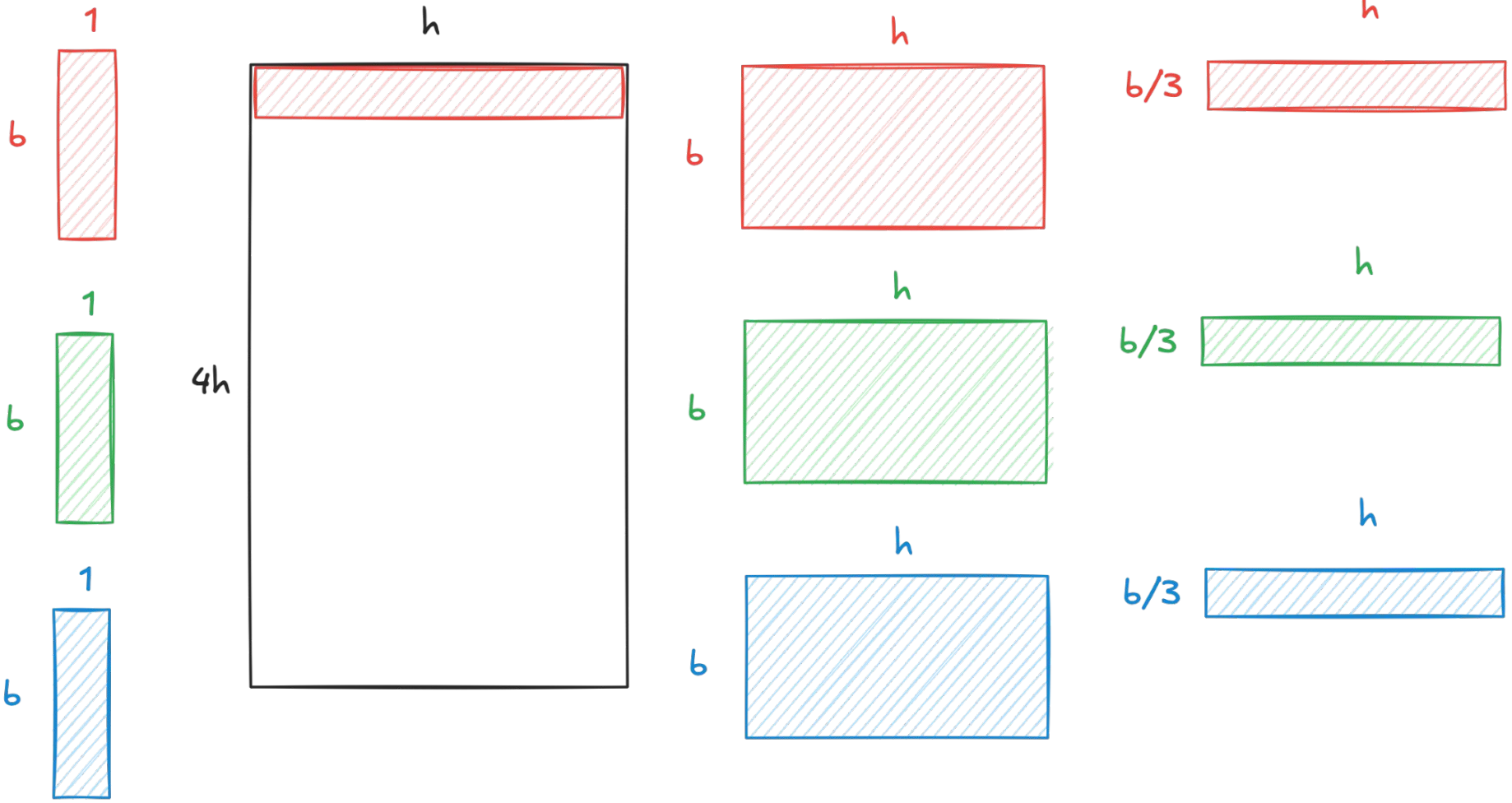
size of Y2_0 == size of Y2

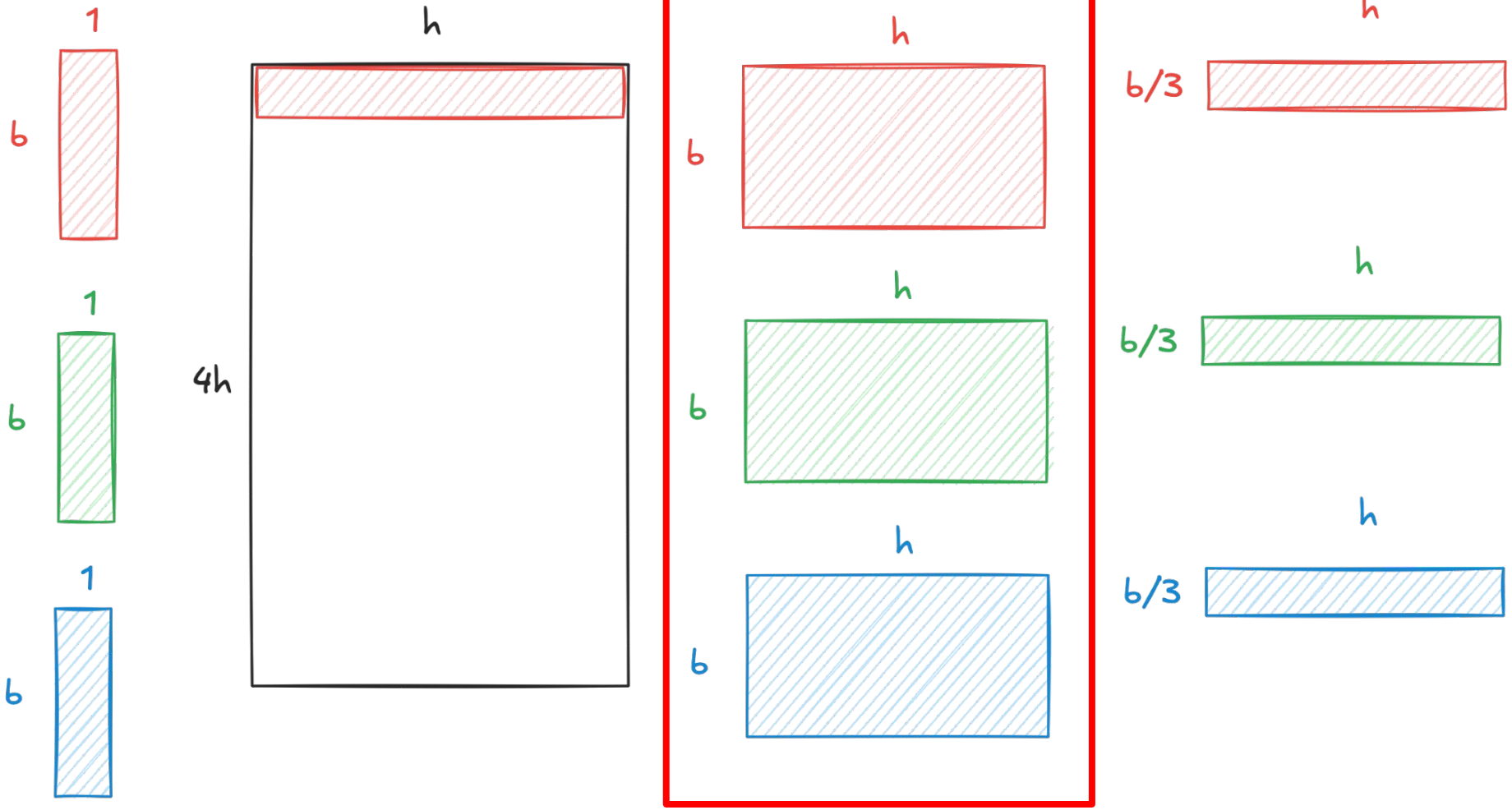
	Standard TP (The Diagram)	Sequence Parallelism
Operation	`All-Reduce`	`Reduce-Scatter`
GPU 0 Output	$\begin{bmatrix} 20 & 40 \\ 80 & 180 \\ 140 & 320 \\ 200 & 460 \end{bmatrix}$	$\begin{bmatrix} 20 & 40 \\ 80 & 180 \end{bmatrix}$
GPU 1 Output	$\begin{bmatrix} 20 & 40 \\ 80 & 180 \\ 140 & 320 \\ 200 & 460 \end{bmatrix}$	$\begin{bmatrix} 140 & 320 \\ 200 & 460 \end{bmatrix}$
Memory Used	**100% (Full Matrix)**	**50% (Sharded Matrix)**

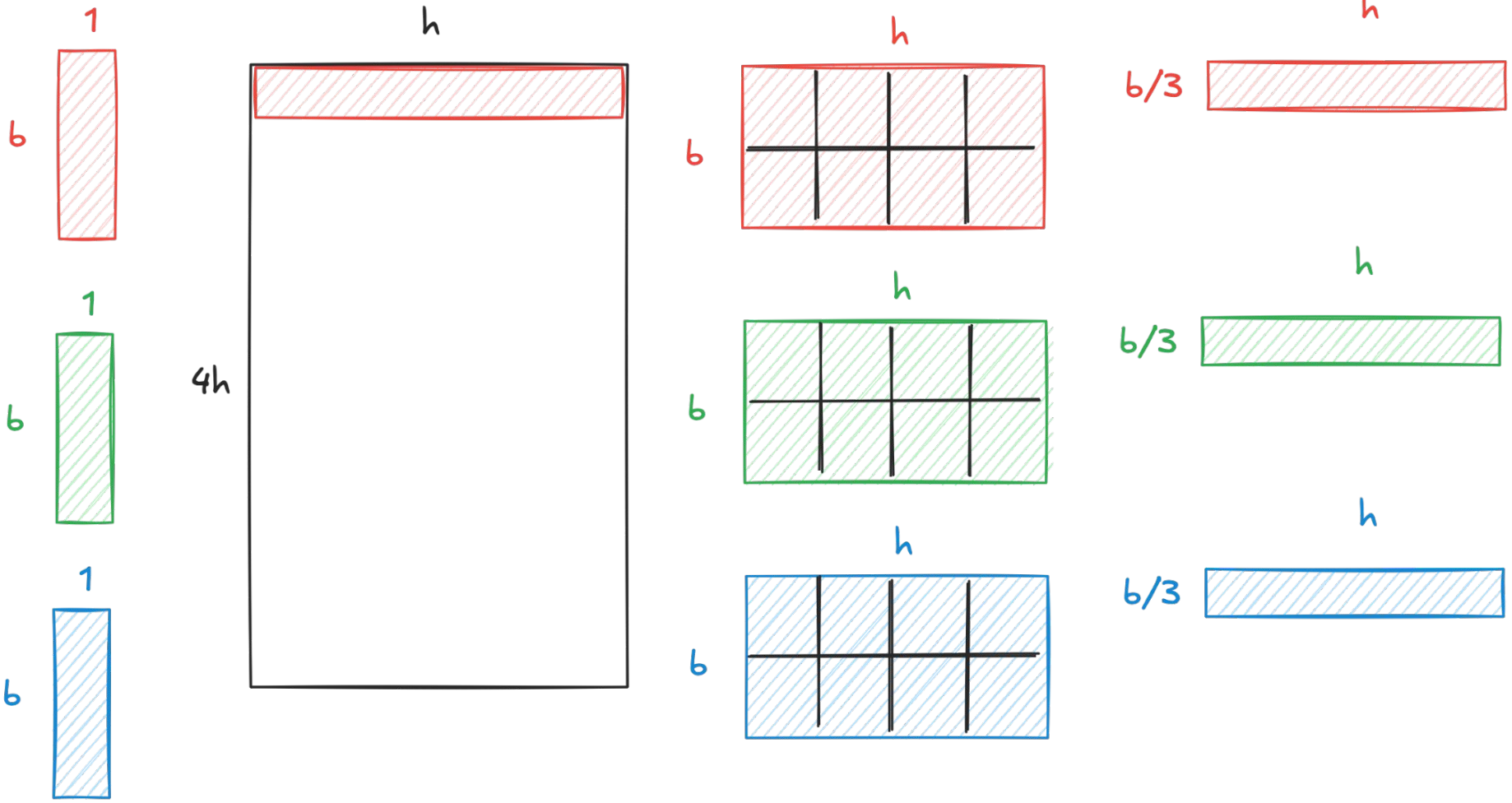

Full tensor: (batch, seq_len, d_model)
↑
SHARDED across GPUs
GPU 0: x[:, :, 0:2D] # all tokens, partials of all features
GPU 1: x[:, :, 0:2D] # all tokens, partials of all features


h
b
h
4h
1
b
1
b
1
b
h
4h
h
b
h
b
h
b
h
b


1
b
1
b
1
b
h
4h
h
b
h
b
h
b


1
b
1
b
1
b
h
4h
h
b
h
b
h
b
h
b/3
h
b/3
h
b/3

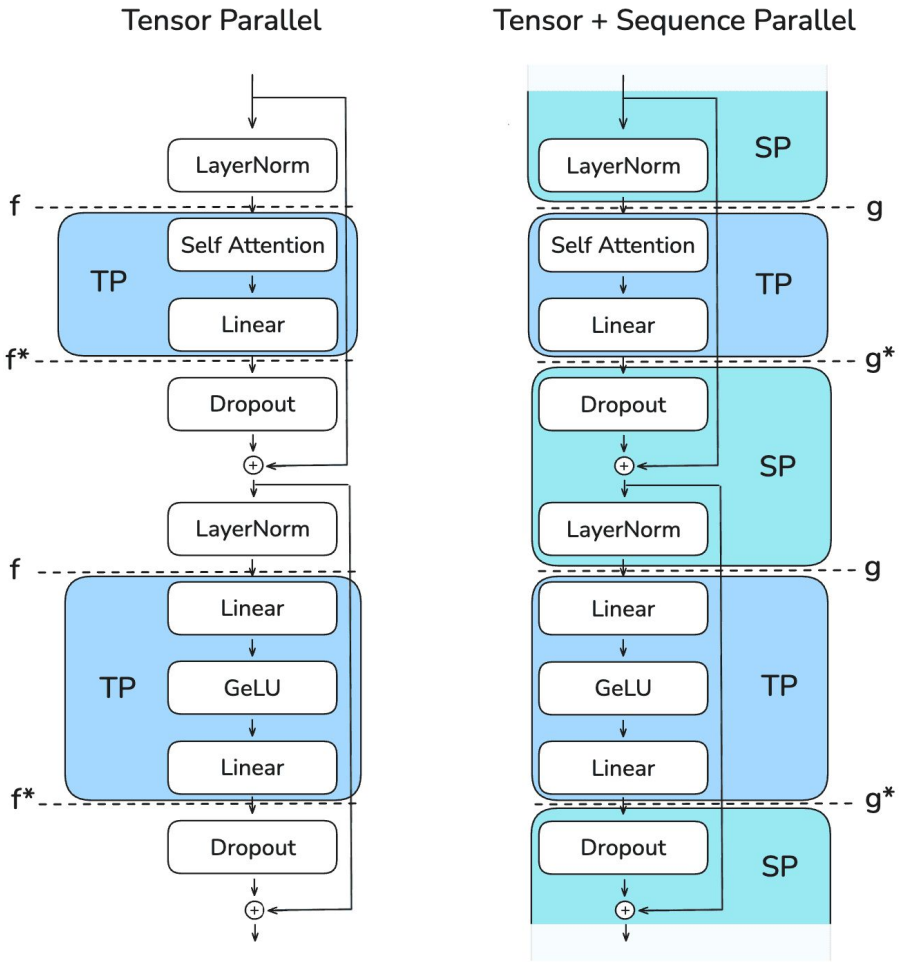

1
b
1
b
1
b
h
4h
h
b
h
b
h
b
h
b/3
h
b/3
h
b/3


1
b
1
b
1
b
h
4h
h
b
h
b
h
b
h
b/3
h
b/3
h
b/3

dropout ???

$$y = x \cdot \mathrm{RandomMask}$$


$y = x \cdot \mathrm{RandomMask}$


Tensor Parallel
Tensor + Sequence Parallel
LayerNorm
f
Self Attention
TP
Linear
f*
Dropout
LayerNorm
f
Linear
TP
GeLU
Linear
f*
Dropout
SP
g
TP
g*
SP
g
TP
g*
SP

Initial LayerNorm layer (SP region)

- Input tensors *X1** and *X2** $(b, s/2, h)$ enter, already split across the sequence dimension.
- Each GPU computes LayerNorm independently on its sequence chunk, giving *Y1** and *Y2**.

First transition (SP → TP)

- *g* operation (all-gather) combines *Y1* and *Y2* back to full sequence length.
- Restores *Y* (b, s, h) since column-linear layers need the full hidden dimension h.

First linear layer (TP region)

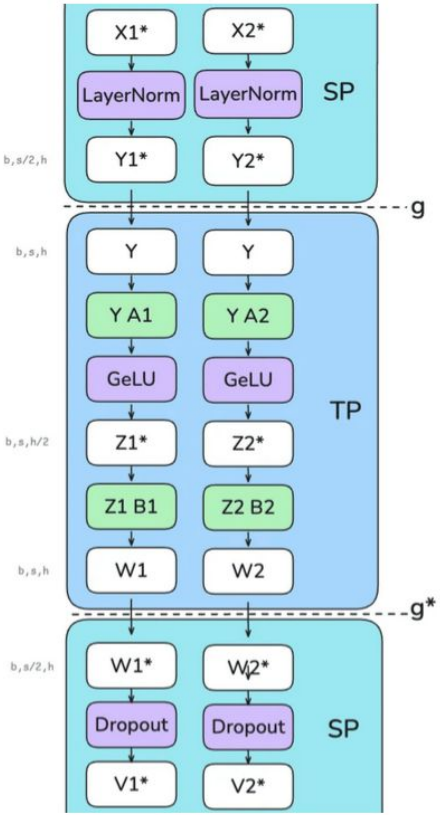
- *A1* and *A2* are column-linear layers, so they split *Y* along the hidden dimension.
- GELU is applied independently on each GPU.
- *Z1** and *Z2** are $(b, s, h/2)$.

Second linear layer (TP region)

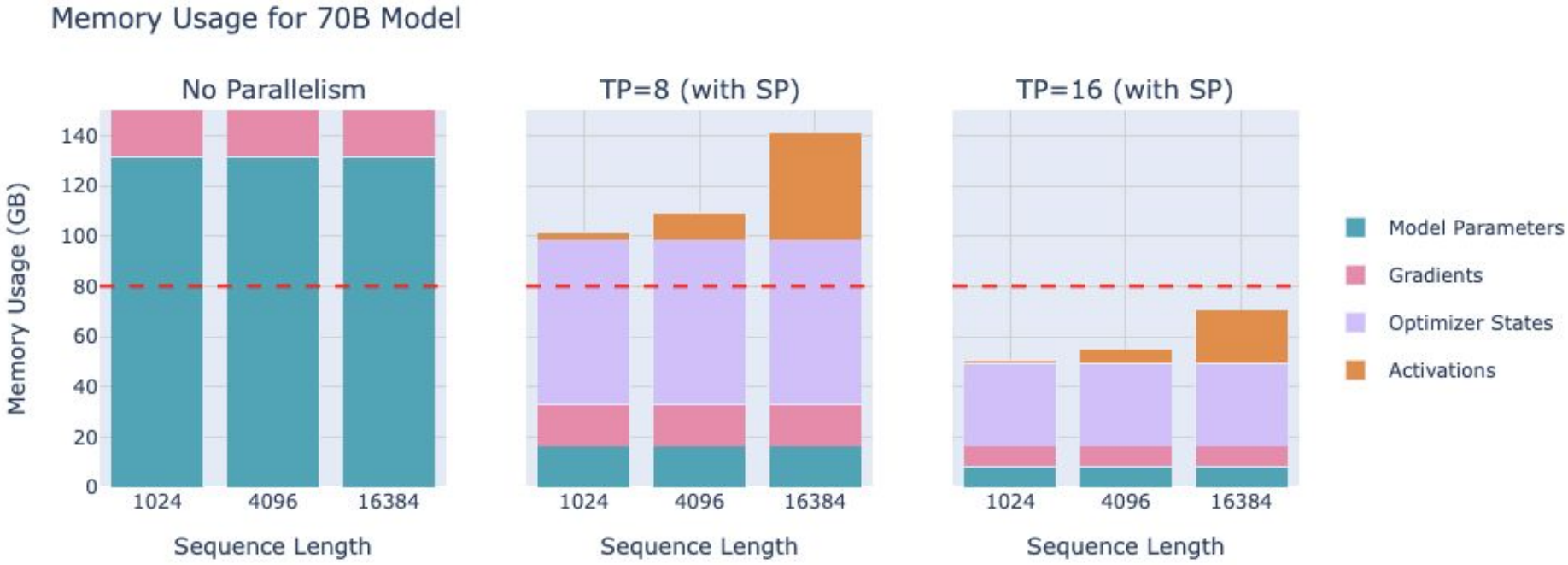
- *B1* and *B2* are row-linear layers, so they restore the hidden dimension.
- *W1* and *W2* are (b, s, h) that need to be summed together.

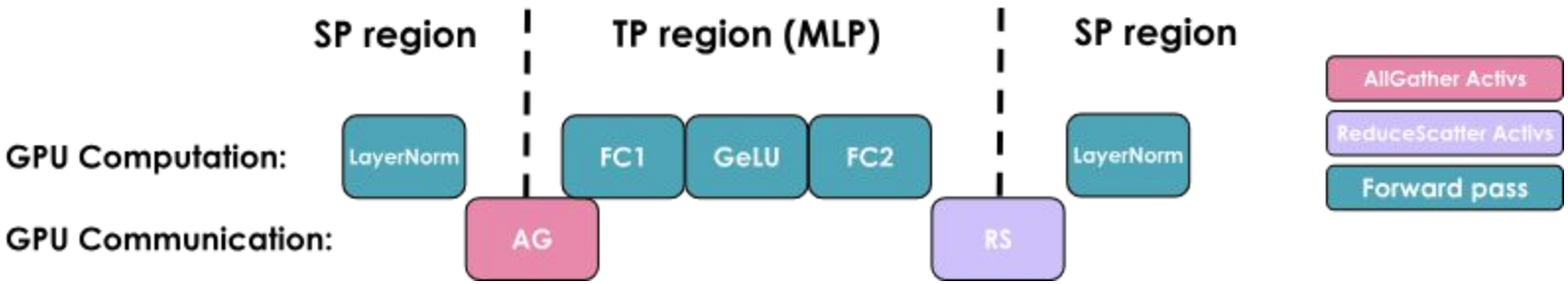
Final transition (TP → SP)

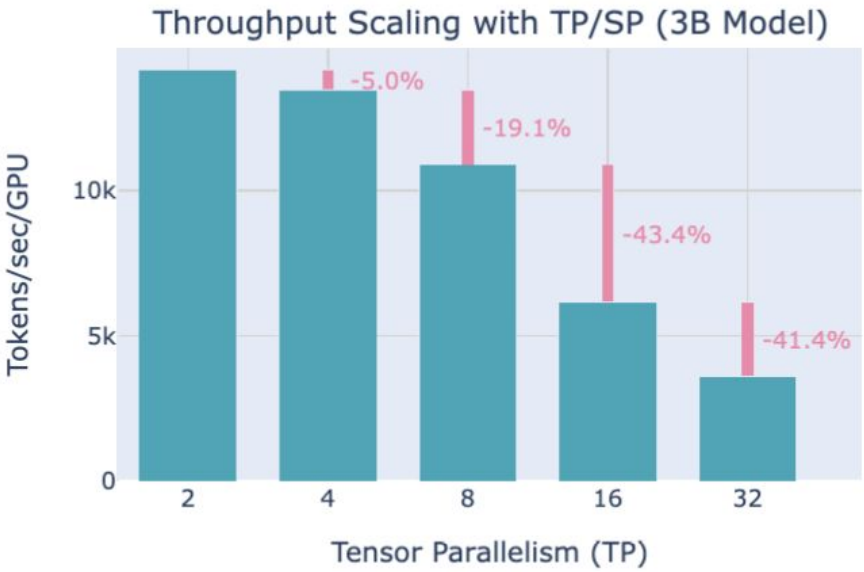
- *g** operation (reduce-scatter) reduces for previous row-linear correctness while scattering along the sequence dimension.
- *W1** and *W2** are $(b, s/2, h)$.

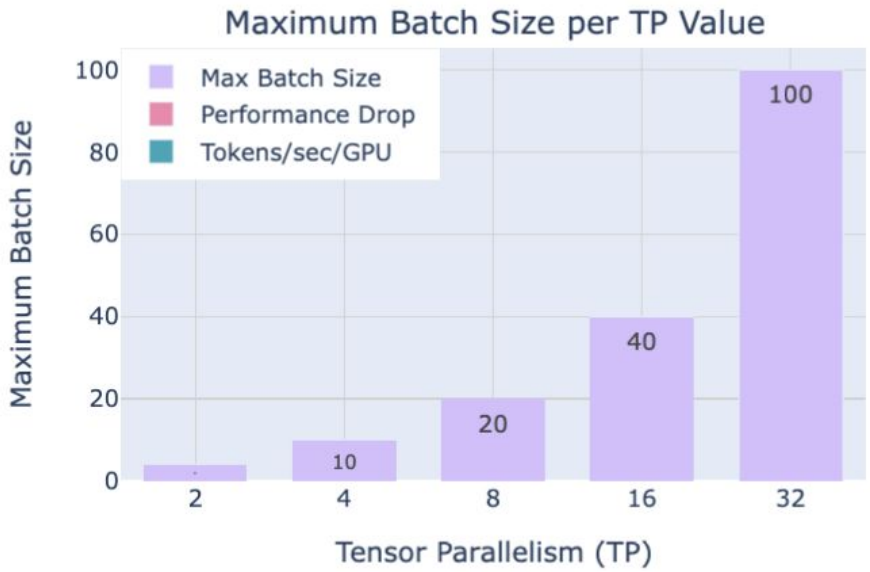


Region	TP only	TP with SP
Enter TP (column-linear)	h: sharded (`weight_out` is sharded) s: full	h: sharded (`weight_out` is sharded) s: **all-gather** to full
TP region	h: sharded s: full	h: sharded s: full
Exit TP (row-linear)	h: full (`weight_out` is full + **all-reduce** for correctness) s: full	h: full (`weight_out` is full + **reduce-scatter** for correctness) s: **reduce-scatter** to sharded
SP region	h: full s: full	h: full s: sharded


Memory Usage for 70B Model
No Parallelism
TP=8 (with SP)
TP=16 (with SP)
Memory Usage (GB)
0
20
40
60
80
100
120
140
1024
4096
16384
Sequence Length
Model Parameters
Gradients
Optimizer States
Activations


SP region
TP region (MLP)
SP region
GPU Computation:
GPU Communication:
LayerNorm
FC1
GeLU
FC2
LayerNorm
AG
RS
AllGather Activs
ReduceScatter Activs
Forward pass


Throughput Scaling with TP/SP (3B Model)
Tokens/sec/GPU
10k
5k
0
-5.0%
-19.1%
-43.4%
-41.4%
2
4
8
16
32
Tensor Parallelism (TP)


Maximum Batch Size per TP Value
Maximum Batch Size
100
80
60
40
20
0
Max Batch Size
Performance Drop
Tokens/sec/GPU
10
20
40
100
2
4
8
16
32
Tensor Parallelism (TP)

limits

- veeery large sequences blow up in TP
- inter-node connectivity tax for big models

see you next time