System for Artificial Intelligence

Parallelization and Training III

Siyuan Feng
Shanghai Innovation Institute

OUTLINE

01 MoE and Expert Parallelism

02 Context Parallelism

03 Activation Checkpoint

01

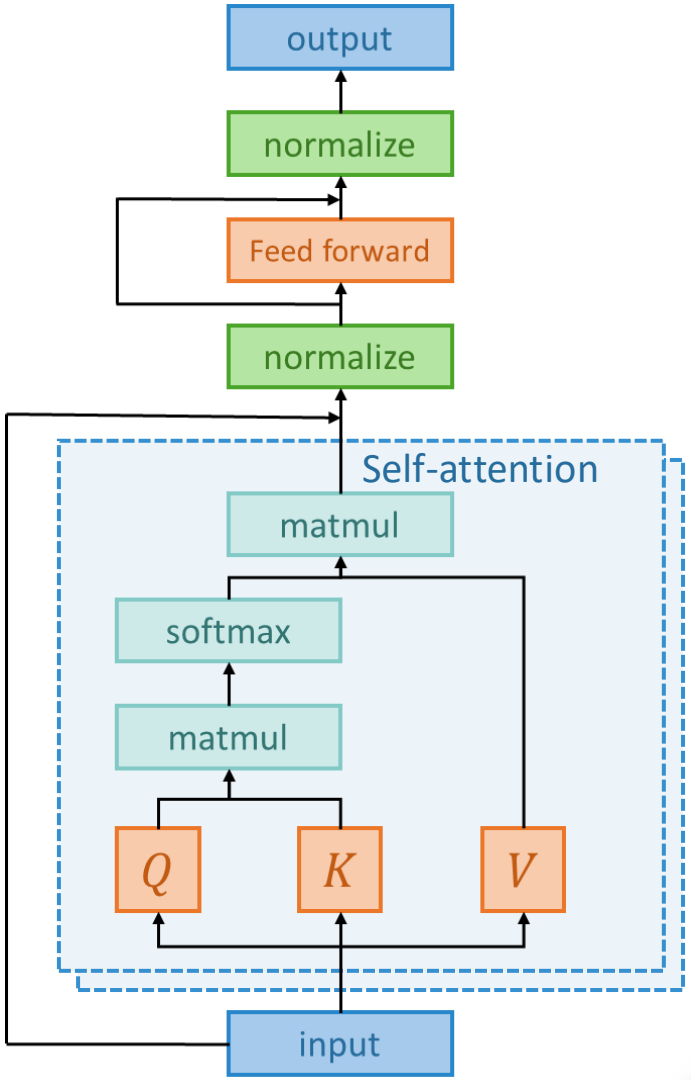
MoE and Expert Parallelism

Recap: Transformer Block

- A typical transformer block :

$$Z = \text{SelfAttention}\left(XW_Q, XW_K, XW_V\right)$$
$$Z = \text{LayerNorm}(X + Z)$$
$$H = \text{LayerNorm}((\text{GeLU}(ZW_1) \odot ZW_2)W_3 + Z)$$

- (multi-head) self-attention, followed by a linear layer and ReLU and some additional residual connections and normalization



Standard Feed Forward Layer

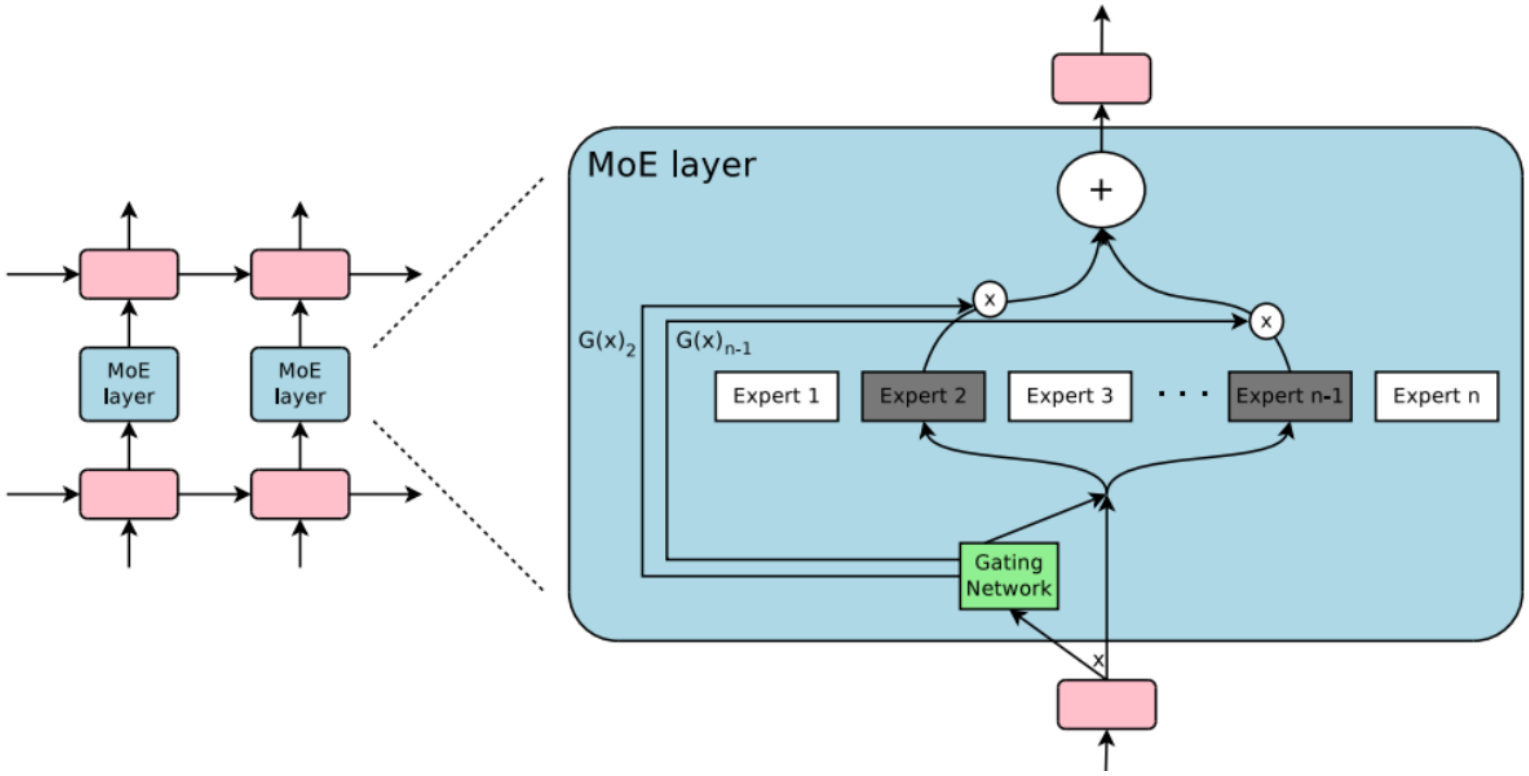
上海创智学院
Shanghai Innovation Institute

- $H = LayerNorm\big((GeLU(ZW_1) \odot ZW_2)W_3 + Z\big)$
- $W_1, W_2 \in R^{n \times m}, W_3 \in R^{m \times n}$

- Increasing feature size will increase compute quadratically
- Everything is mixed together in the FFN(feed forward network) layer

Mixture-of-Experts

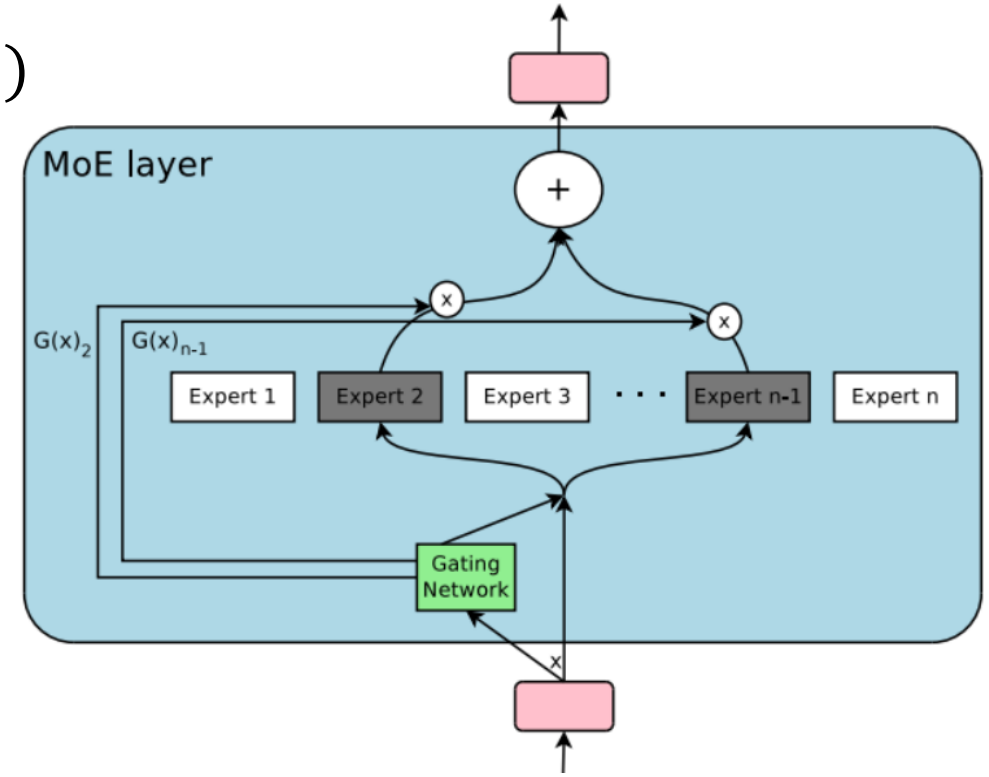
- Key idea: make each expert focus on predicting the right answer for a subset of cases
- **Actual: a kind of model-level sparsity.**



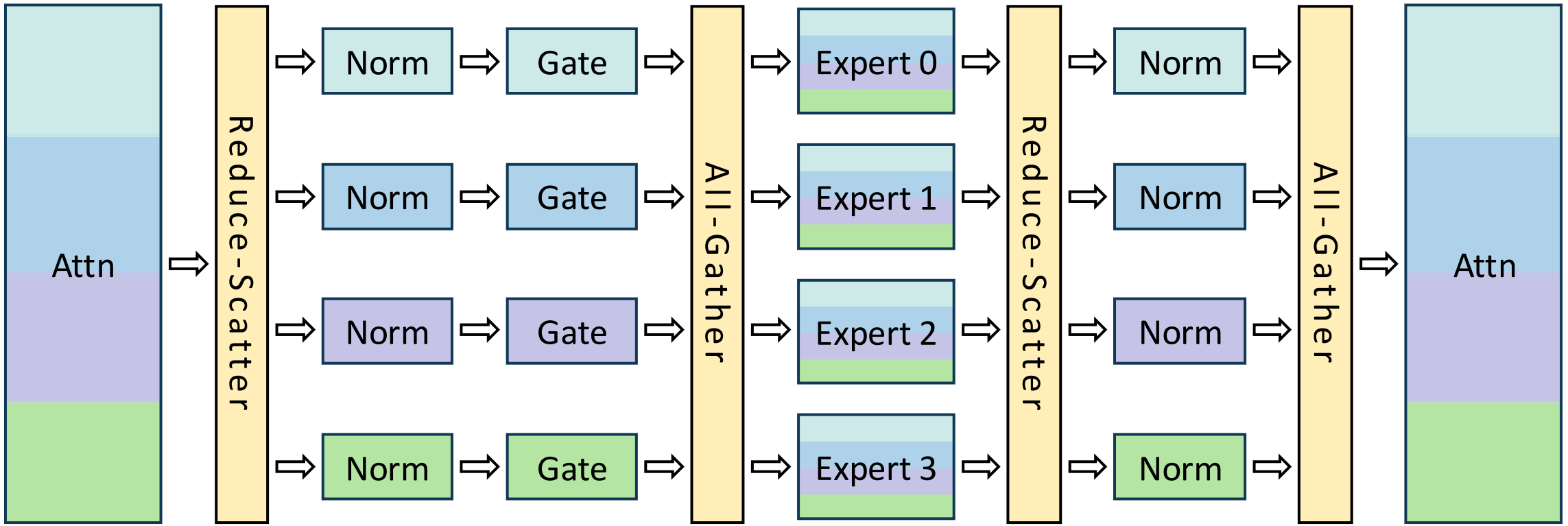
A Closer Look at Mixture-of-Experts

上海创智学院 Shanghai Innovation Institute

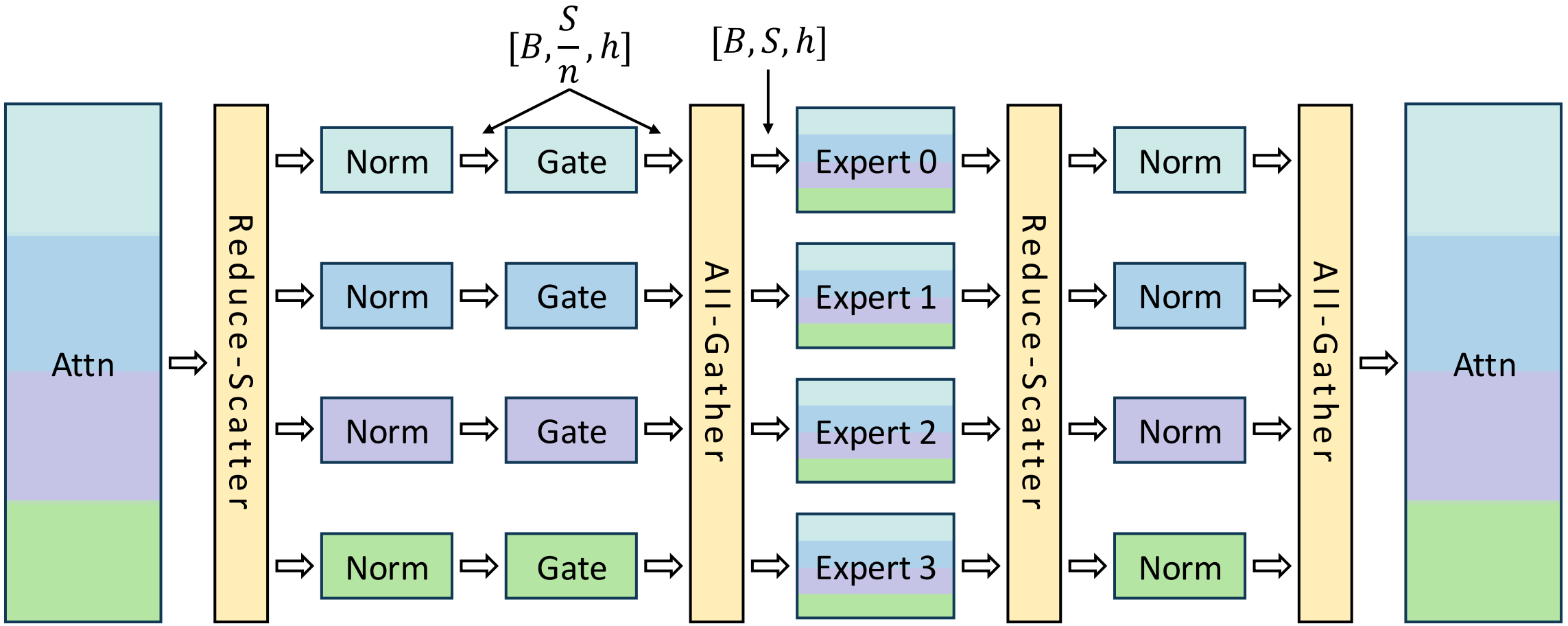
- A typical MoE layer (assume single instance and activate two experts)
  - Gating: $G = \mathrm{softmax}(W_G X)$
  - Expert indices: $I = \{i_0, i_1\} = \mathrm{TopK}(\mathrm{G}, \mathrm{k} = 2)$
  - Output weight: $s_0 = \frac{G_{i_0}}{G_{i_0} + G_{i_1}}, s_1 = \frac{G_{i_1}}{G_{i_0} + G_{i_1}}$
  - Output: $Y = s_0 FFN_{i_0}(X) + s_1 FFN_{i_1}(X)$
- Greedily select top-K experts among N



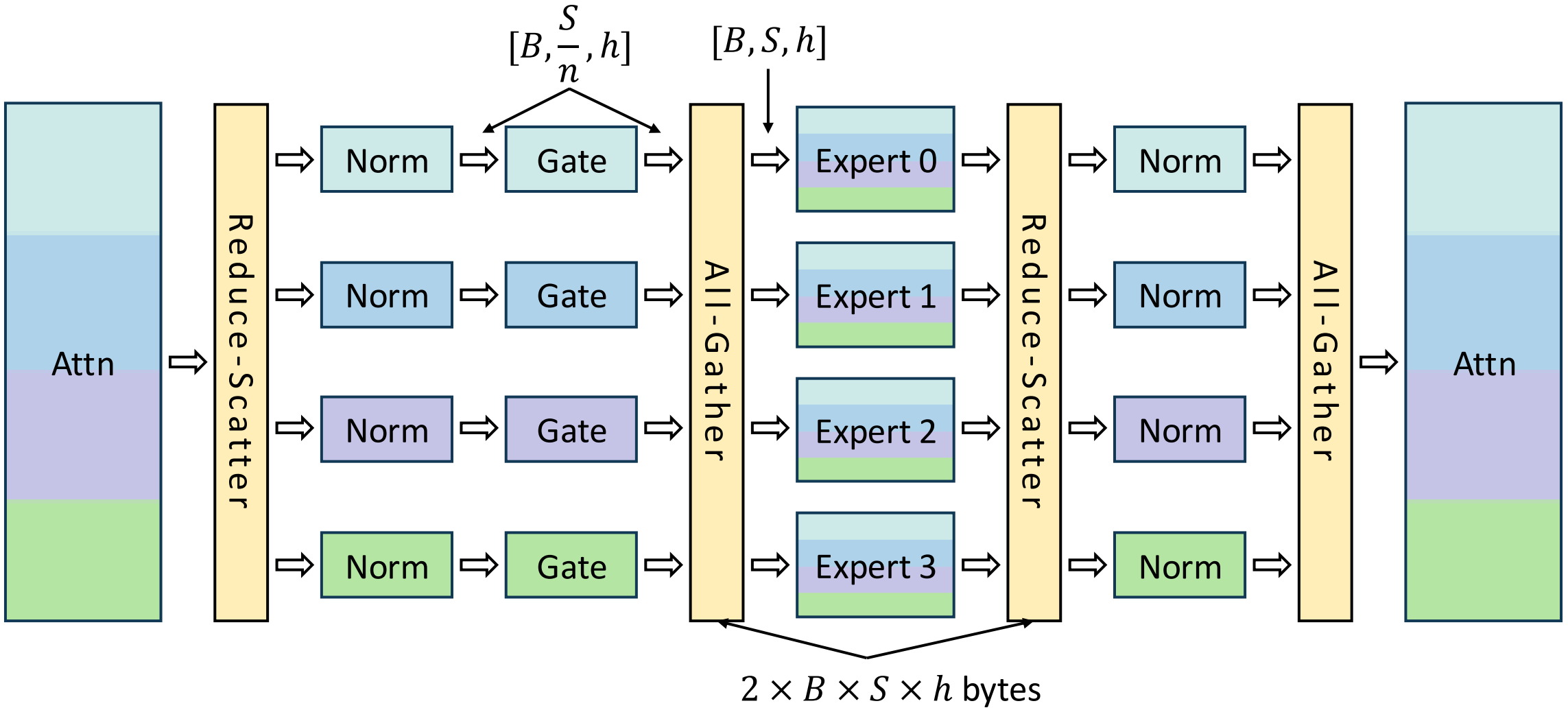
MoE Training w/ TP+SP



MoE Training w/ TP+SP

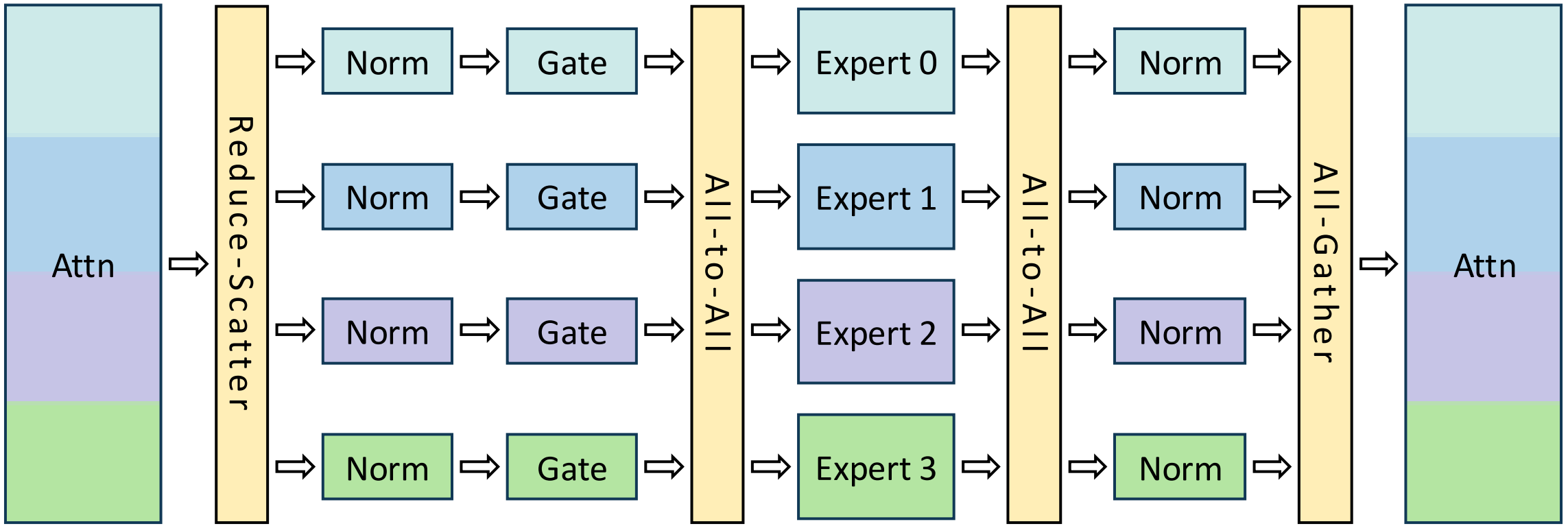


MoE Training w/ TP+SP

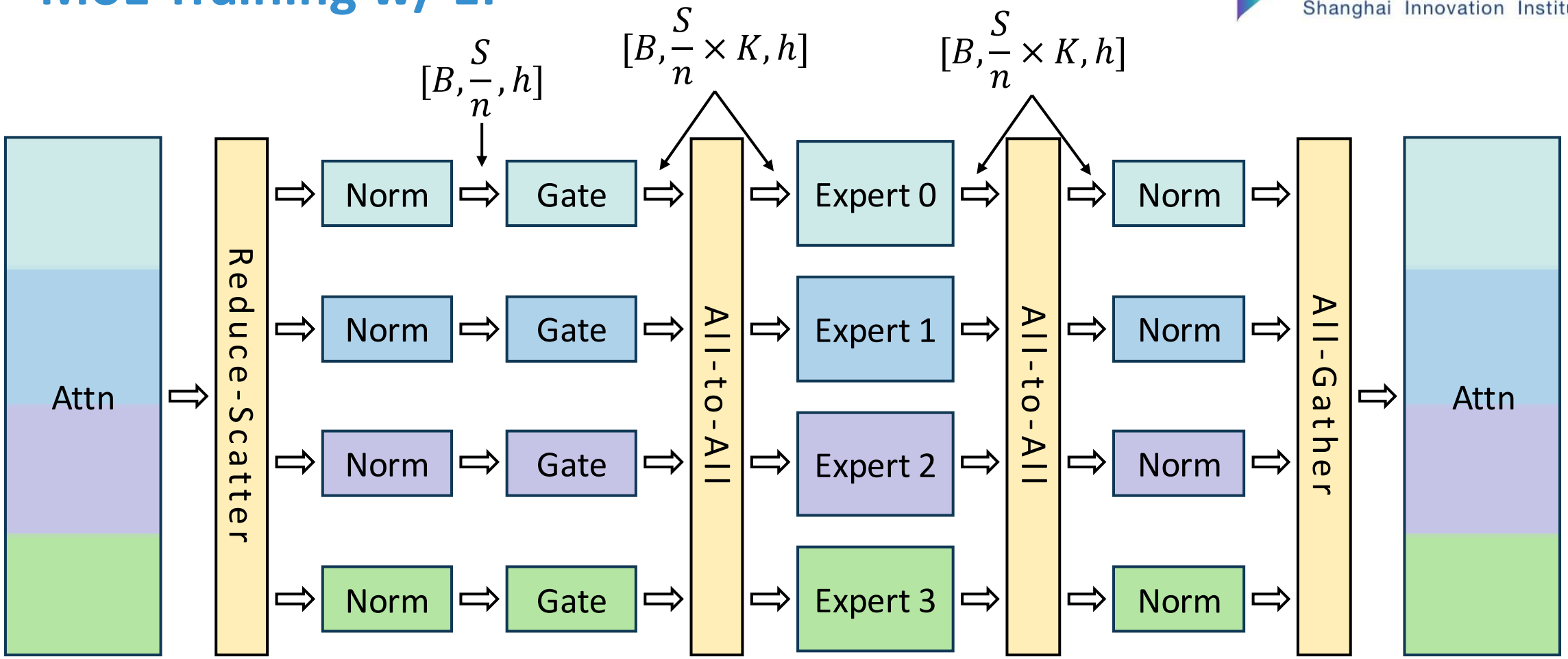


Total communication: $4 \times B \times S \times h$ bytes for bfloat16

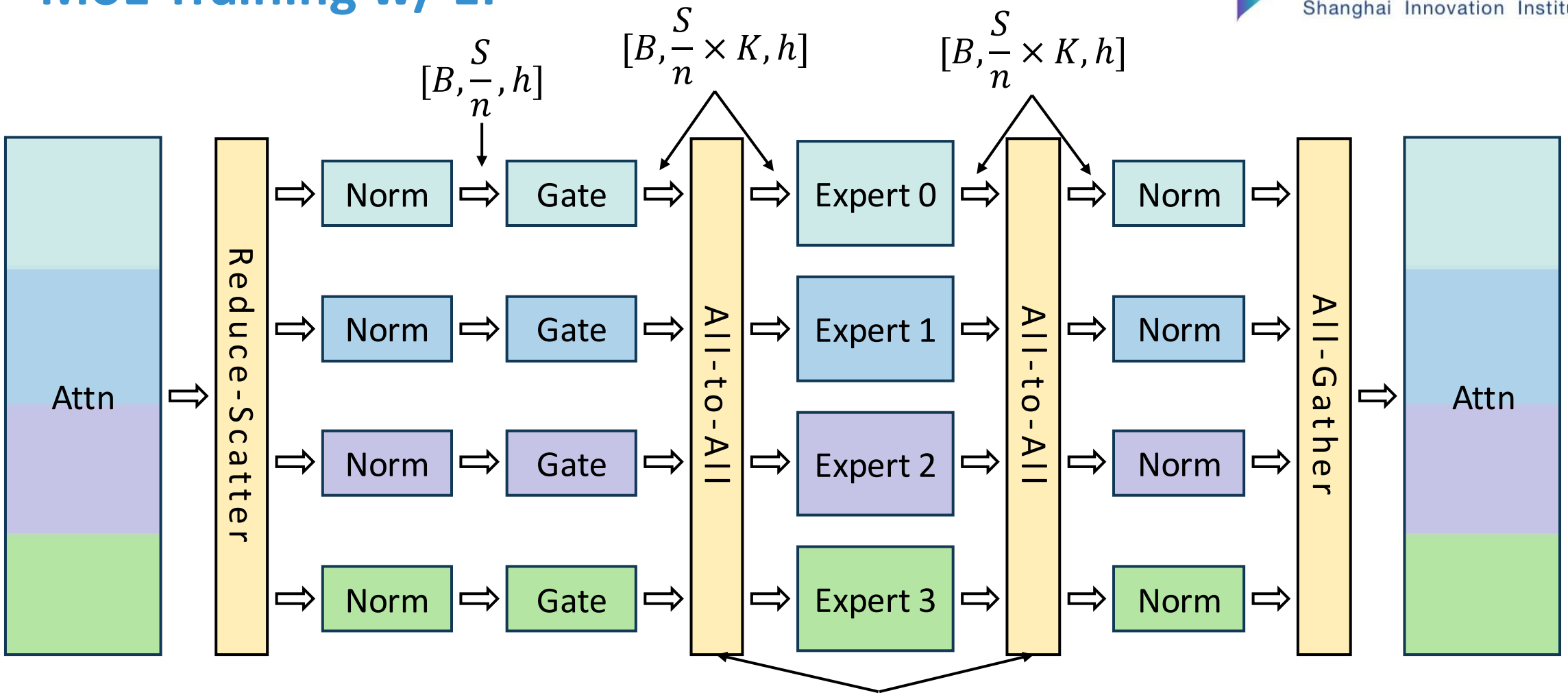
MoE Training w/ EP



MoE Training w/ EP



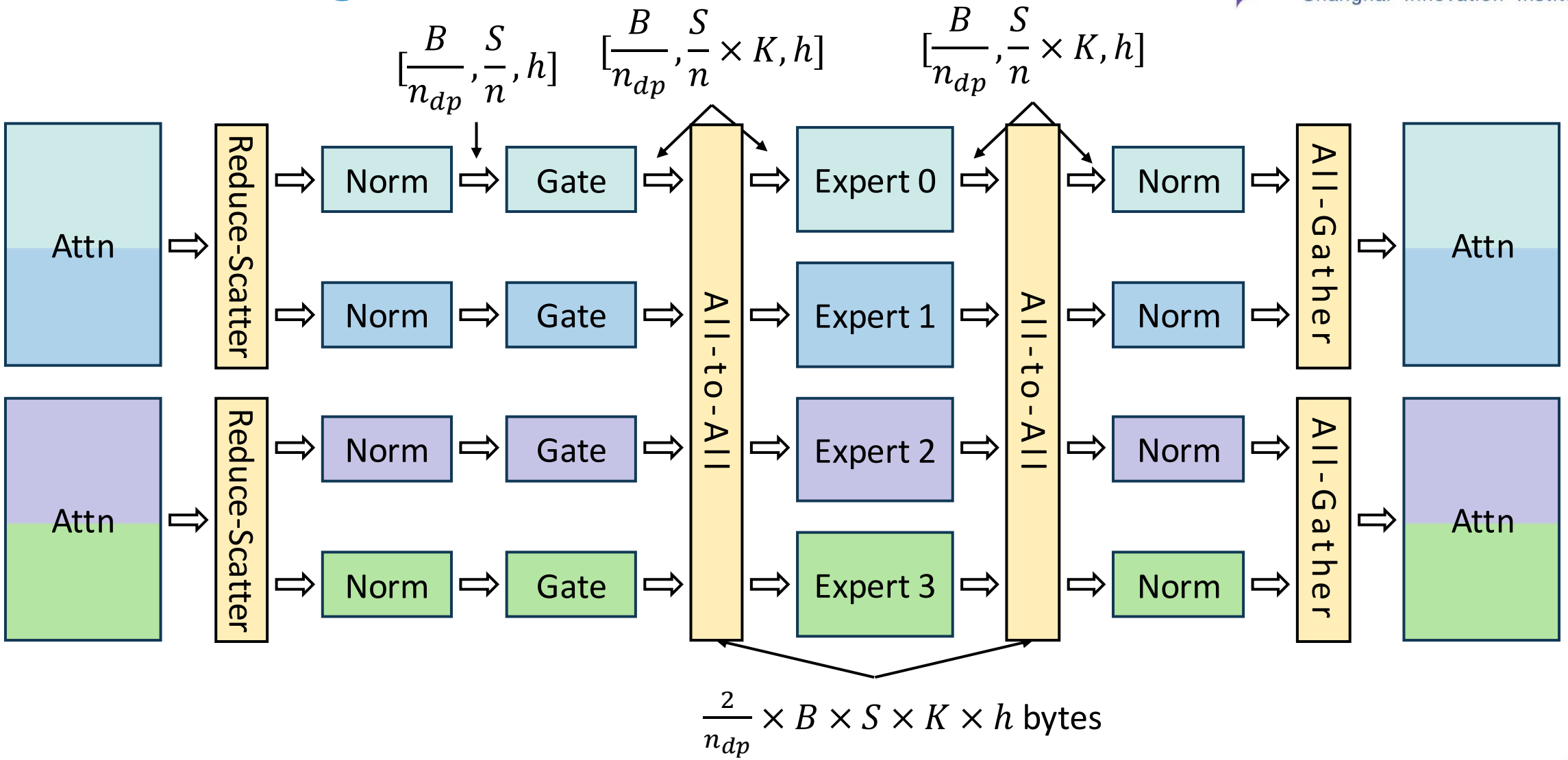
MoE Training w/ EP



$$B \times \frac{S}{n} \times K \times h \times \frac{n-1}{n} \times n \times 2 \approx 2 \times B \times S \times K \times h \text{ bytes}$$

Total communication: $4 \times B \times S \times K \times h$ bytes for bfloat16

MoE Training w/ EP + DP



Total communication: $\frac{4}{n_{dp}} \times B \times S \times K \times h$ bytes for bfloat16

Best Practice for MoE Training

上海创智学院
Shanghai Innovation Institute

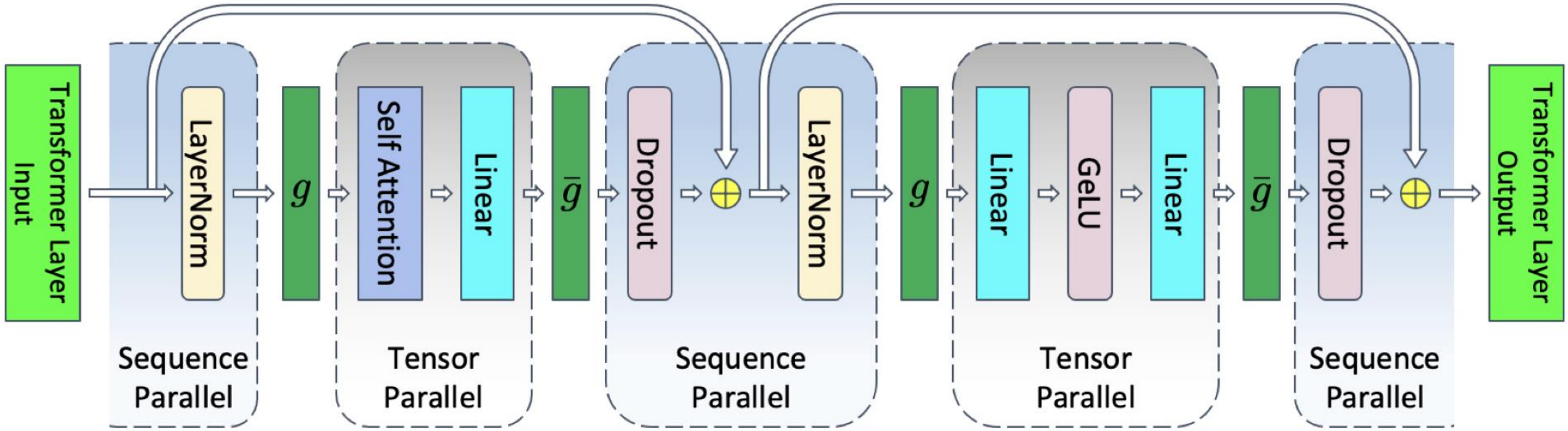
- Communication (for Bfloat16):
  - Pure TP: $4 \times B \times S \times h$ bytes
  - Pure EP (TP size = EP size): $4 \times B \times S \times K \times h$ bytes
  - EP + DP Attention: $\frac{4}{n_{dp}} \times B \times S \times K \times h$ bytes

- When $n_{dp} > K$, choose EP will be better than TP.
- For example of DS-V3, $K = 8$, and training with DP 1024, TP 1 and EP 256.

02

Context Parallelism

Recap: Sequence Parallelism



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

Activation Memory of Sequence Parallelism

上海创智学院 Shanghai Innovation Institute

- Attention Block Activations
  - Input to LayerNorm (before Attention): $B \cdot \frac{S}{t} \cdot H \cdot P$
  - Input to FlashAttention (Q, K, V): $B \cdot S \cdot \frac{H+2H_{KV}}{t} \cdot P$
  - Input to Linear: $B \cdot S \cdot \frac{H}{t} \cdot P$
- MLP Block Activations
  - Input to LayerNorm (before MLP): $B \cdot \frac{S}{t} \cdot H \cdot P$
  - Input to MLP Linear Layers (Up/Gate Projections): $B \cdot S \cdot \frac{H}{t} \cdot P$
  - Input to Down Projection: $B \cdot S \cdot \frac{2H_{inter}}{t} \cdot P$
- Total Memory per Layer: $\frac{1}{t} \cdot B \cdot S \cdot (4H + 2H_{KV} + 2H_{inter}) \cdot P$

B: batch size

S: Sequence Length

H: Hidden Size

H_{KV}: GQA hidden size for KV

H_{inter}: MLP Intermediate Size

t: TP / SP world size

P: Precision (2 for bf16)

Taking Llama 70B as example

上海创智学院
Shanghai Innovation Institute

- Memory per Layer: $\frac{1}{t} \cdot B \cdot S \cdot (4H + 2H_{KV} + 2H_{inter}) \cdot P$
  - $B = 1, S = 128k$
  - $H = 8192, H_{KV} = 1024, H_{inter} = 28672$
  - $P = 2, t = 8, num_{layer} = 80$
- Totally memory usage is around 225GB

Can we use larger t to reduce activation memory?

Answer is NO, as GQA KV head is only 8.

B: batch size

S: Sequence Length

H: Hidden Size

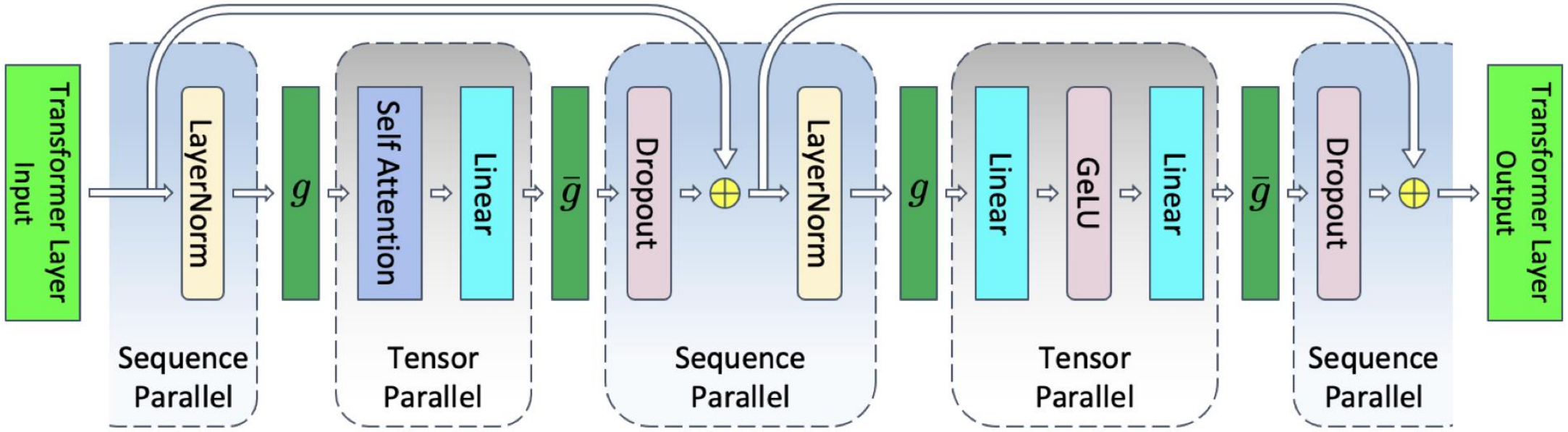
H_{KV}: GQA hidden size for KV

H_{inter}: MLP Intermediate Size

t: TP / SP world size

P: Precision (2 for bf16)

Recap: Sequence Parallelism



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

Activation X is not sharded during attention field

Recap: Recursive Attention

上海创智学院
Shanghai Innovation Institute

- Pre-softmax "attention score" $s_i = \frac{1}{\sqrt{d}} q k_i^T$

$$s(I) = \log\left(\sum_{i \in I} \exp(s_i)\right), \mathrm{v}(I) = \sum_{i \in I} \mathrm{softmax}(s)_i v_i = \frac{\sum_{i \in I} \exp(s_i) v_i}{\exp(s(I))}$$

- When index set $I = \{i\}$, $\mathrm{s}(\{i\}) = s_i$, $\mathrm{v}(\{i\}) = v_i$
- When index set $I = \{1, 2 \ldots t\}$, $\mathrm{v}(I)$ is the final output of the attention

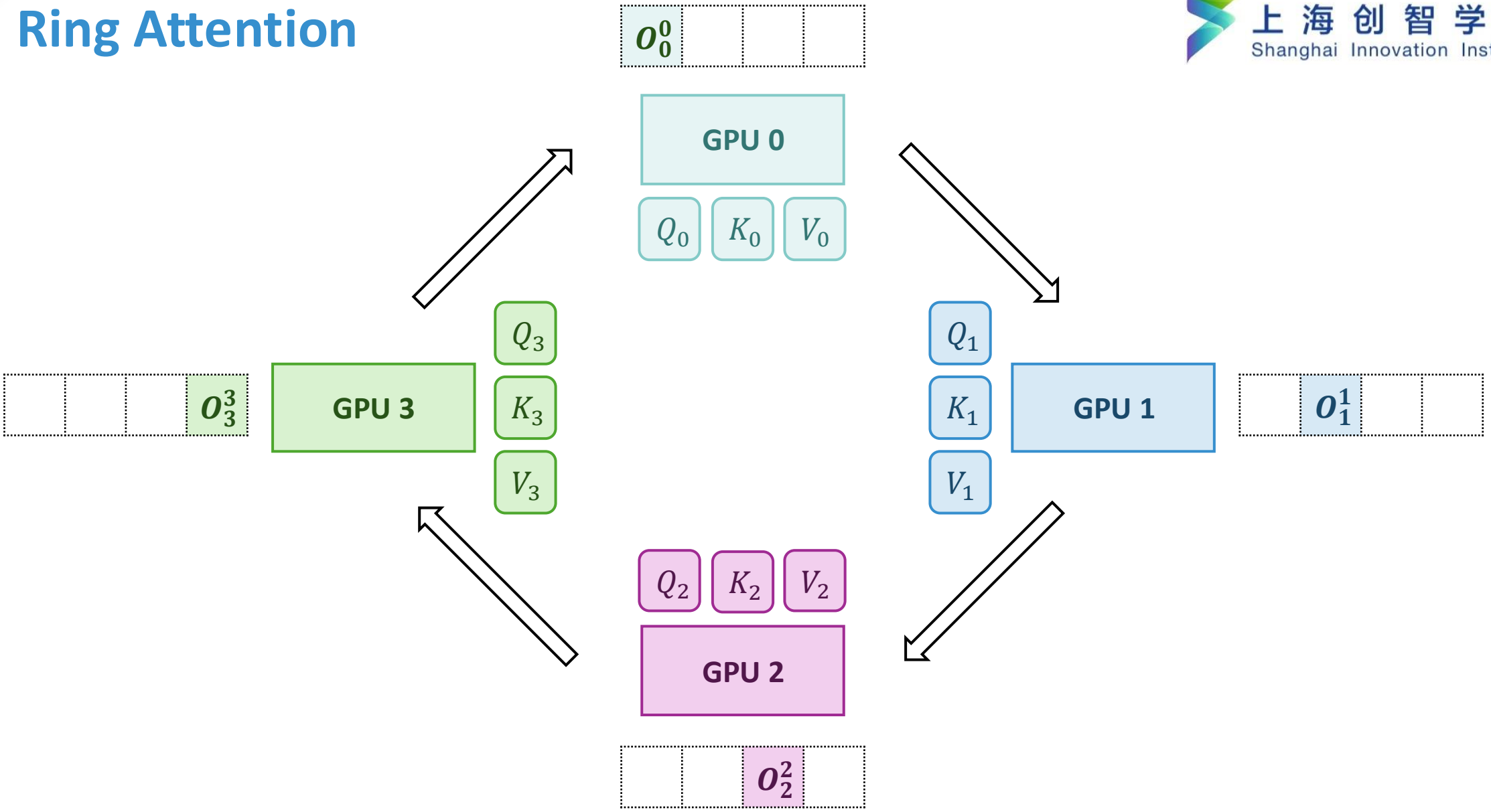
Recap: Recursive Attention

上海创智学院
Shanghai Innovation Institute

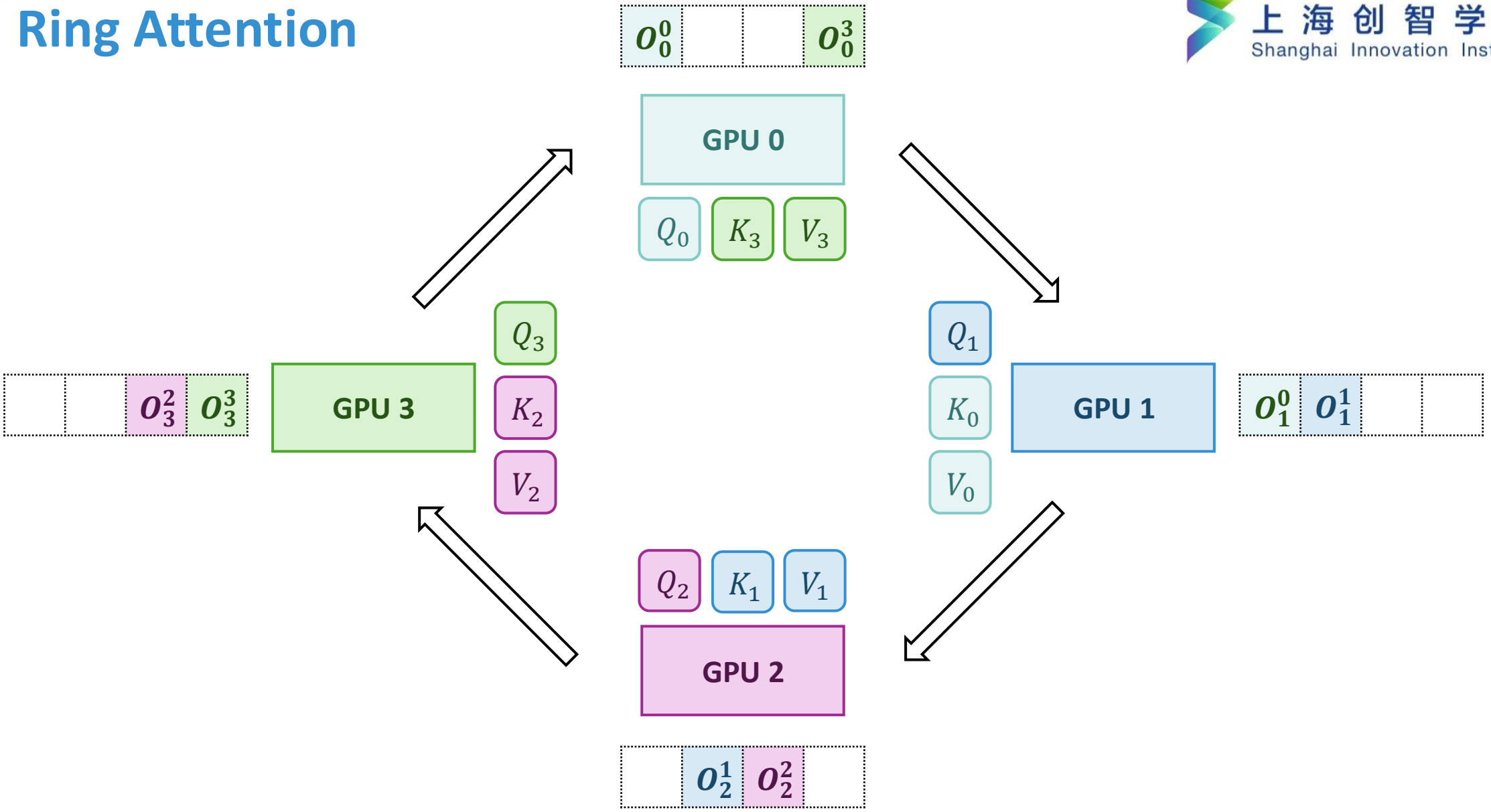
$$s(I) = \log(\sum_{i \in I} \exp(s_i)) \, , \mathrm{v}(I) = \sum_{i \in I} \mathrm{softmax}(s)_i v_i = \frac{\sum_{i \in I} \exp(s_i) v_i}{\exp(s(I))}$$

- For any partition $\{I_j\}$ of I such that $I = \mathrm{U}_{j=1}^{n} I_j$, the following relation holds
- $\mathrm{s}(\mathrm{U}_{j=1}^{n} I_j) = \log\left(\sum_j \exp\left(s(I_j)\right)\right), \mathrm{v}(\mathrm{U}_{j=1}^{n} I_j) = \sum_j \mathrm{softmax}([s(I_1), s(I_2) \ldots])_j \, v(I_j)$
- Attention computation is **communicative** and **associative** and can be done in a divide-and-conquer fashion. An important property for a lot of system optimizations
- Discussion: what can we do with this property?

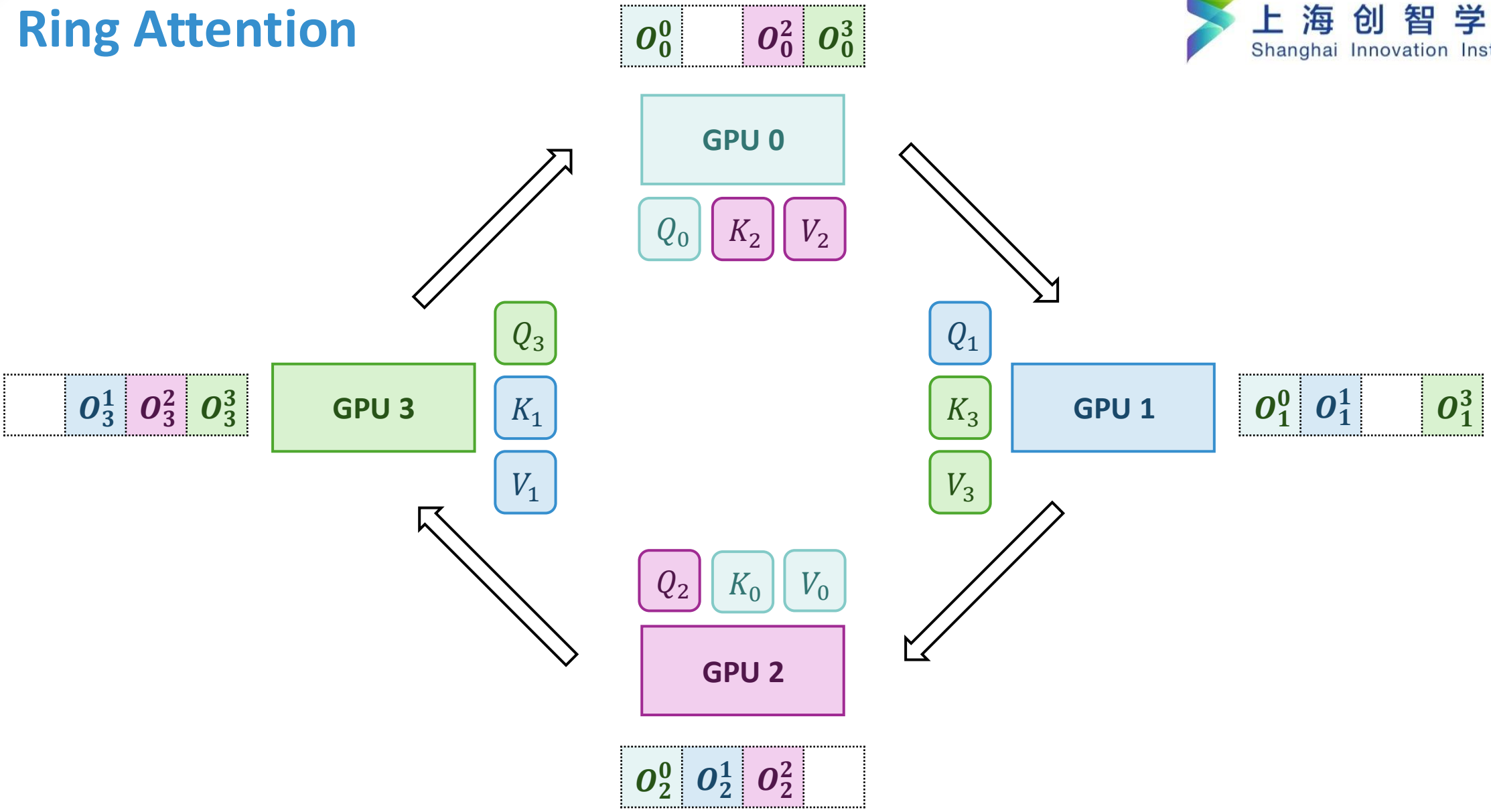
Ring Attention



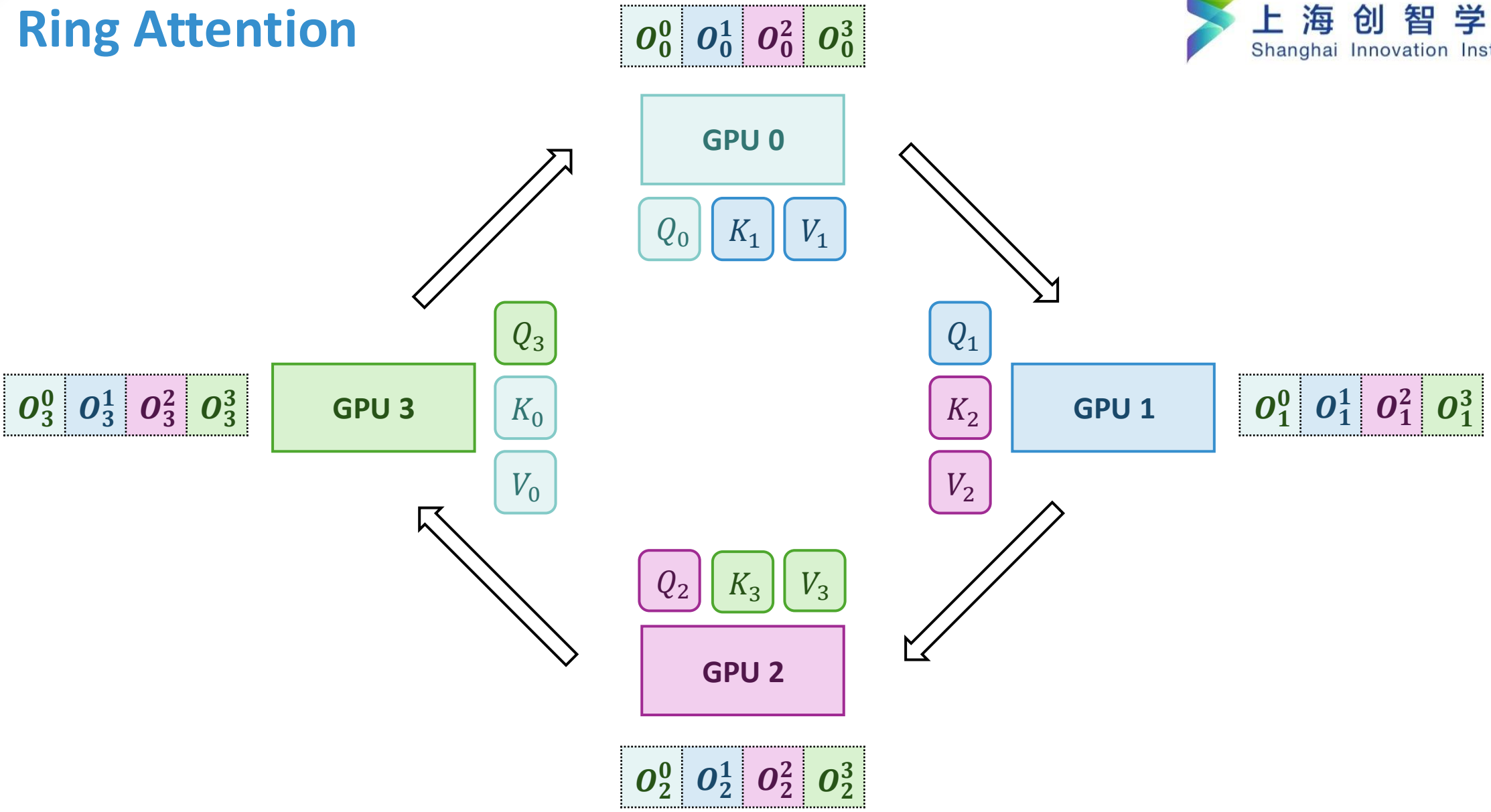
Ring Attention



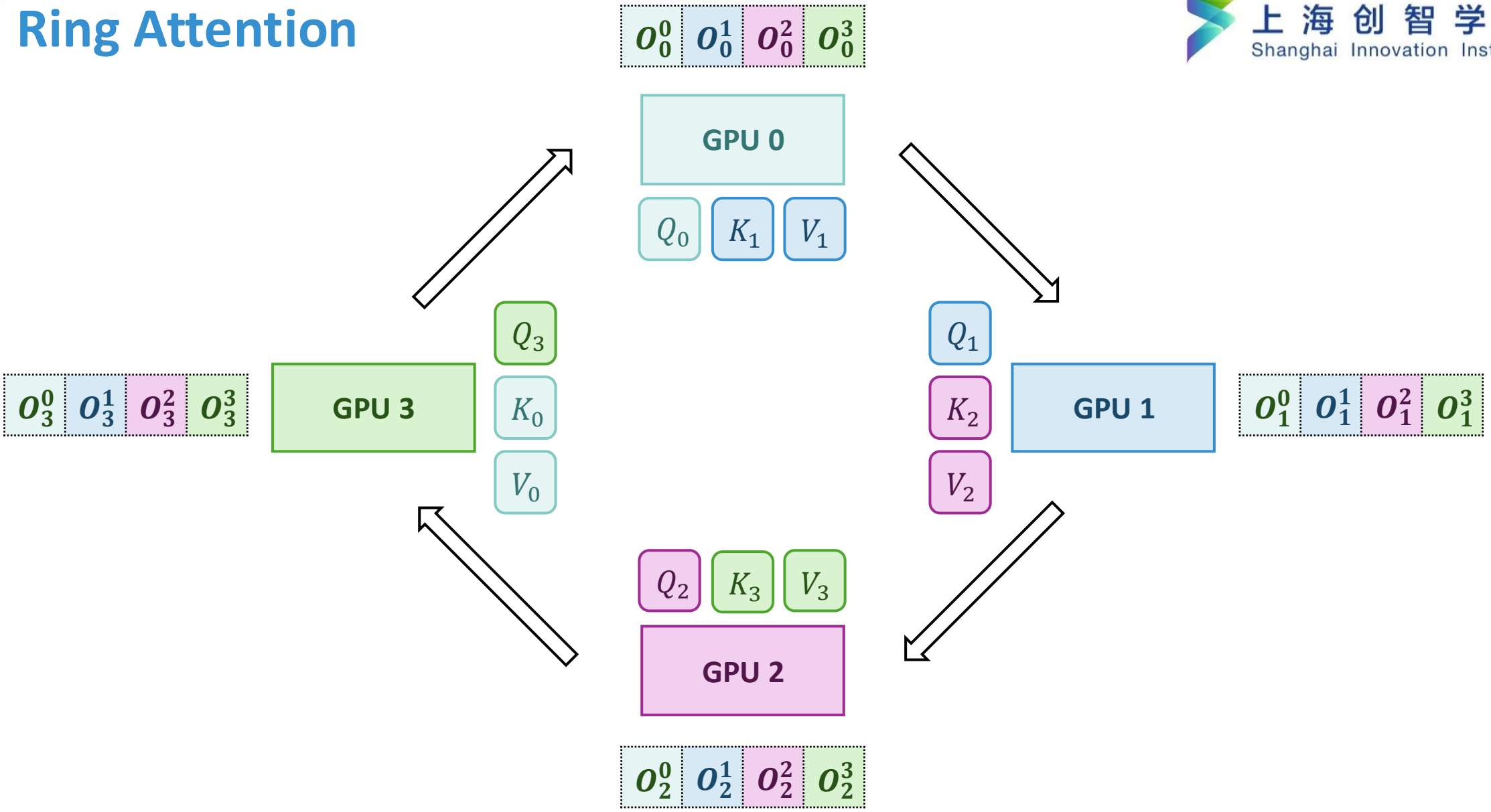
Ring Attention



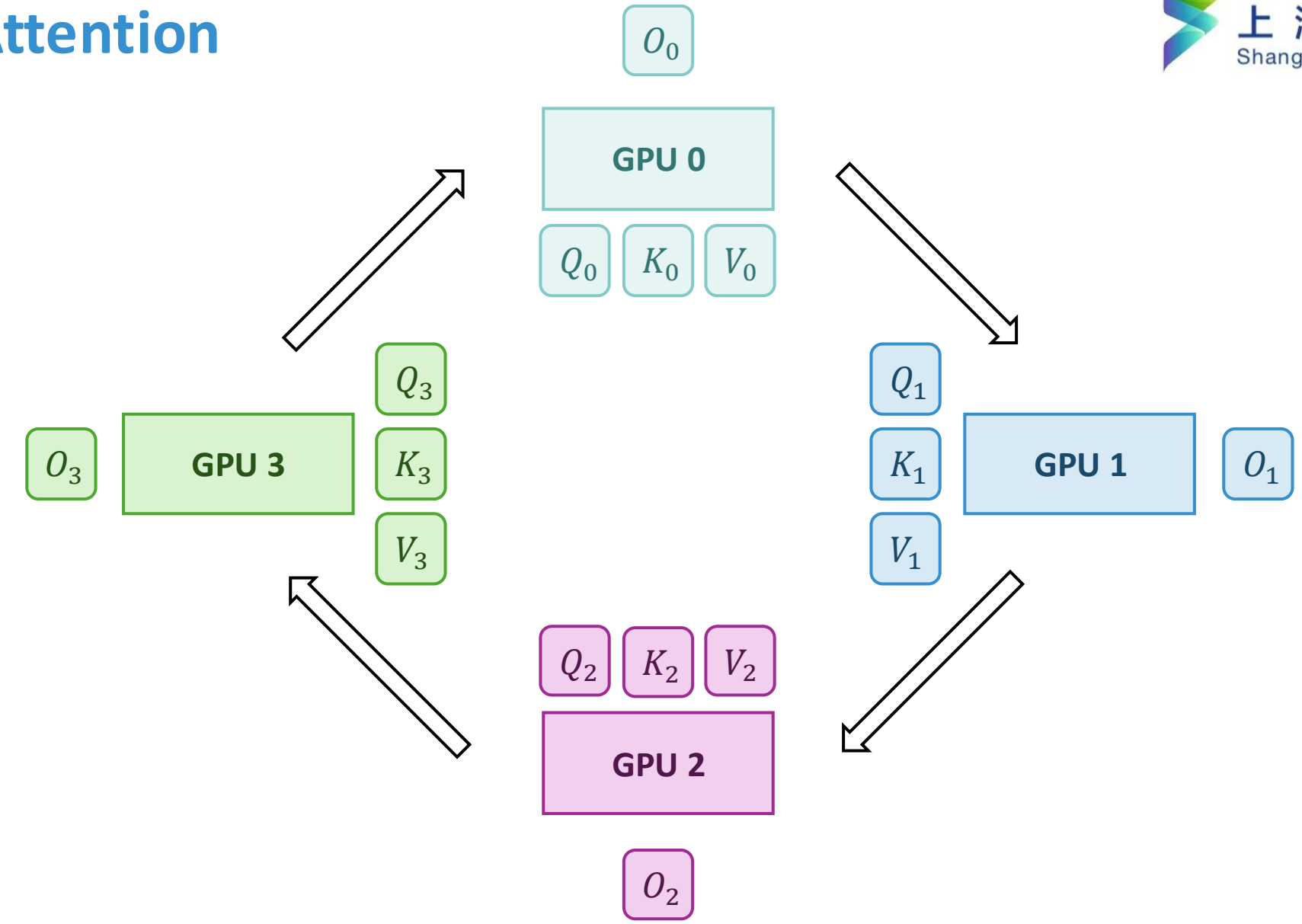
Ring Attention



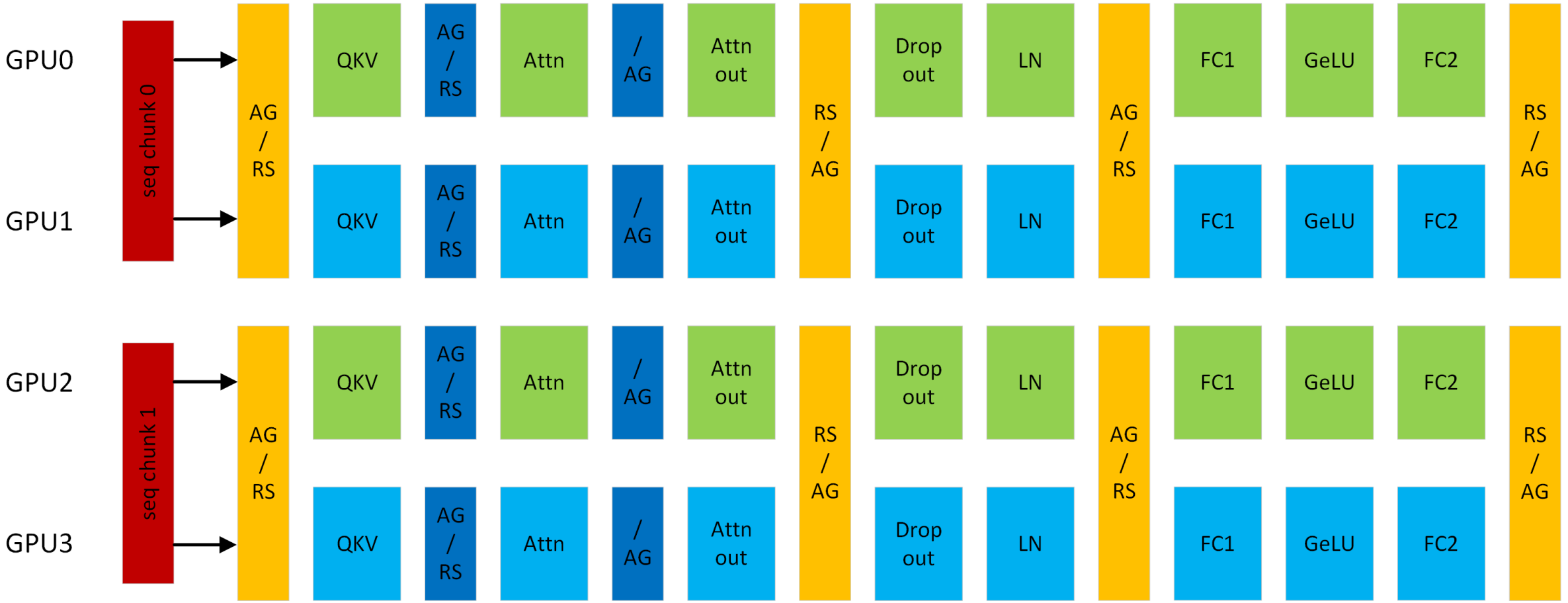
Ring Attention



Ring Attention



Transformer Layer with TP+CP



Activation Memory of Context Parallelism

上海创智学院 Shanghai Innovation Institute

- Attention Block Activations
  - Input to LayerNorm (before Attention): $B \cdot \frac{S}{c \cdot t} \cdot H \cdot P$
  - Input to FlashAttention (Q, K, V): $B \cdot \frac{S}{c} \cdot \frac{H+2H_{KV}}{t} \cdot P$
  - Output of Attention: $B \cdot \frac{S}{c} \cdot \frac{H}{t} \cdot P$
- MLP Block Activations
  - Input to LayerNorm (before MLP): $B \cdot \frac{S}{c \cdot t} \cdot H \cdot P$
  - Input to MLP Linear Layers (Up/Gate Projections): $B \cdot \frac{S}{c} \cdot \frac{H}{t} \cdot P$
  - Input to Down Projection: $B \cdot \frac{S}{c} \cdot \frac{2H_{inter}}{t} \cdot P$
- Total Memory per Layer: $\frac{1}{c \cdot t} \cdot B \cdot S \cdot (4H + 2H_{KV} + 2H_{inter}) \cdot P$

B: batch size

S: Sequence Length

H: Hidden Size

H_{KV}: GQA hidden size for KV

H_{inter}: MLP Intermediate Size

t: TP / SP world size

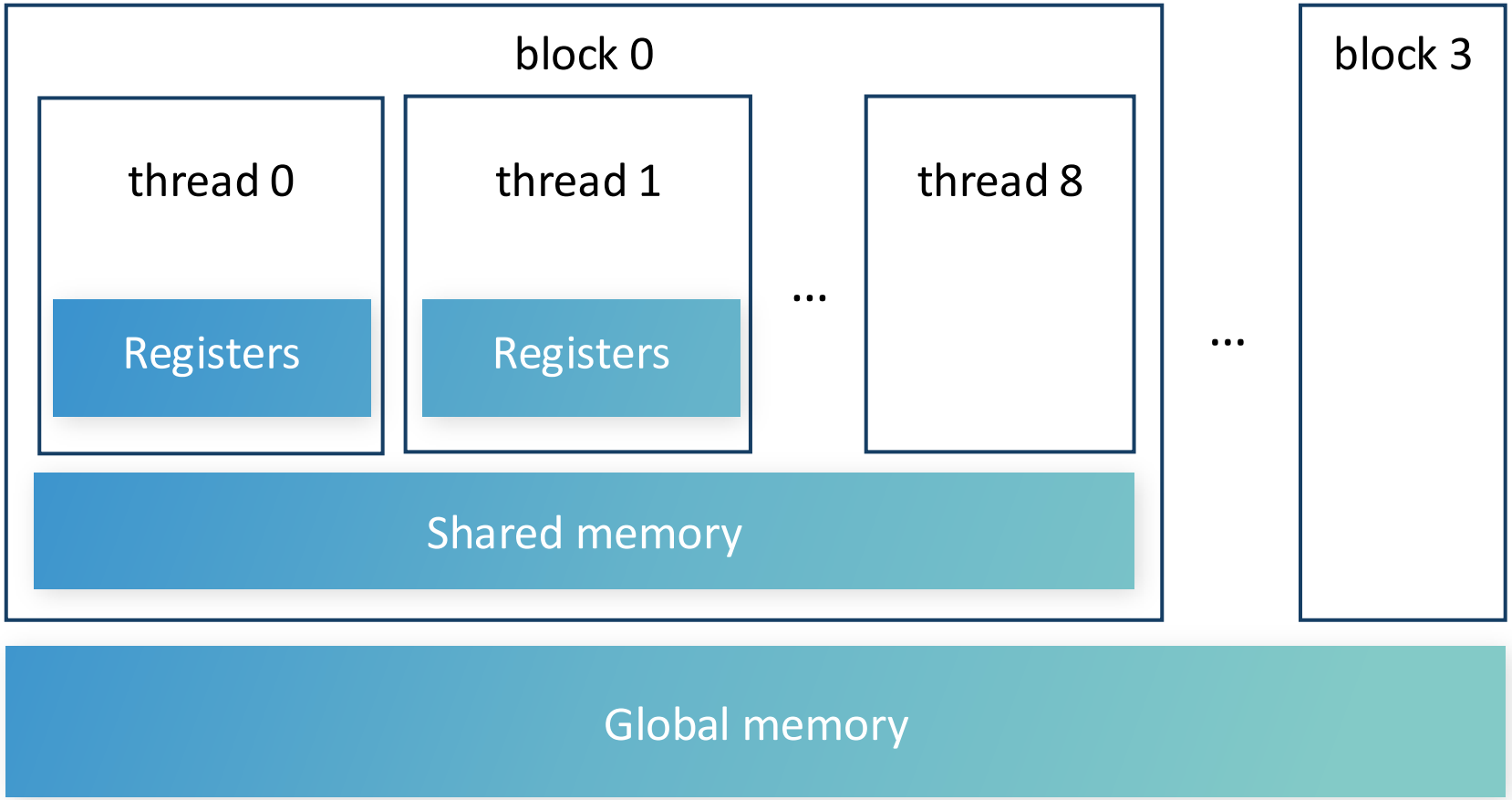
c: CP world size

P: Precision (2 for bf16)

03

Activation Checkpoint

Recap: GPU memory hierarchy



Shared memory: 64 KB per core

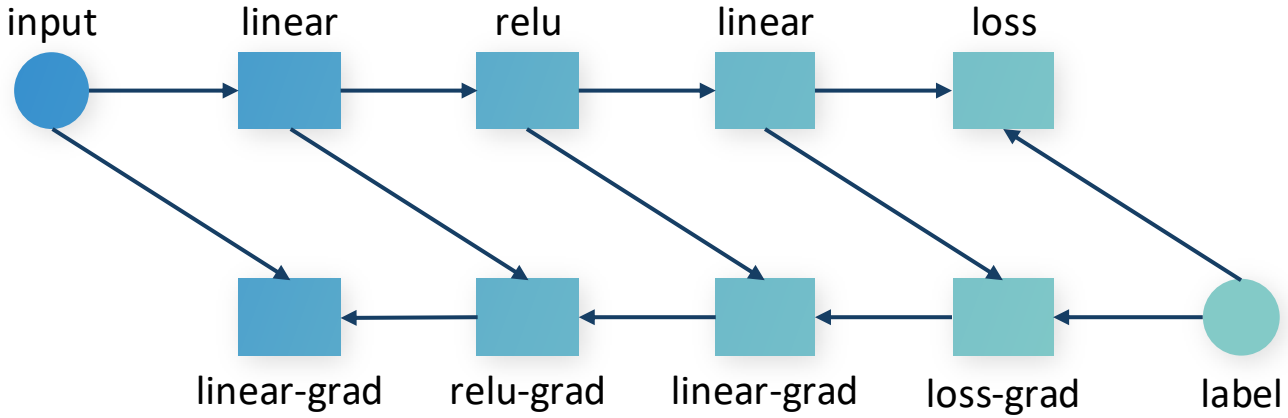
GPU memory(Global memory):

RTX3080 10GB
RTX3090 24GB
A100 40/80 GB

Sources of memory consumption

上海创智学院 Shanghai Innovation Institute

A simplified view of a typical computational graph for training, weights are omitted and implied in the grad steps.

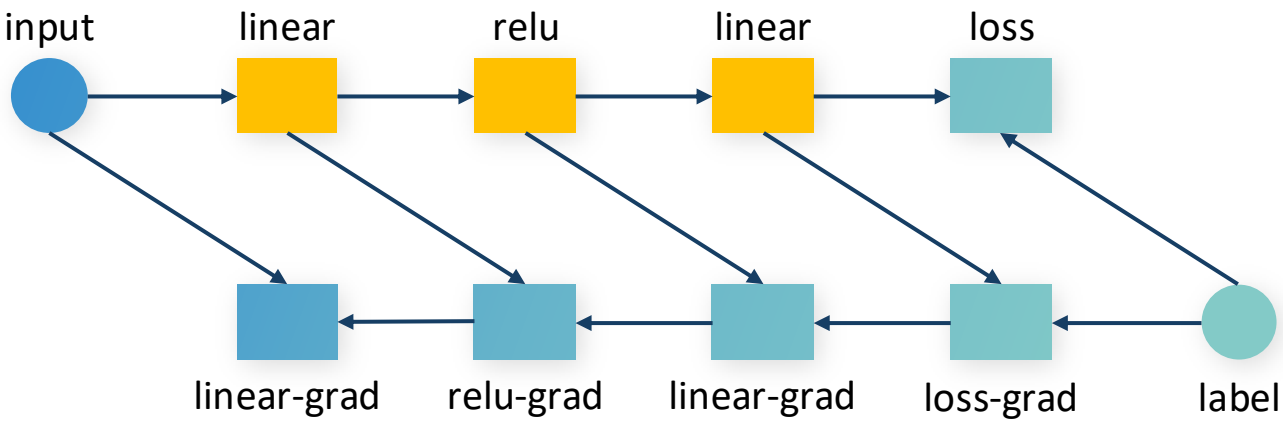


Sources of memory consumption

- Model weights
- Optimizer states
- Intermediate activation values

Optimizer states

Activation Memory Cost for Training

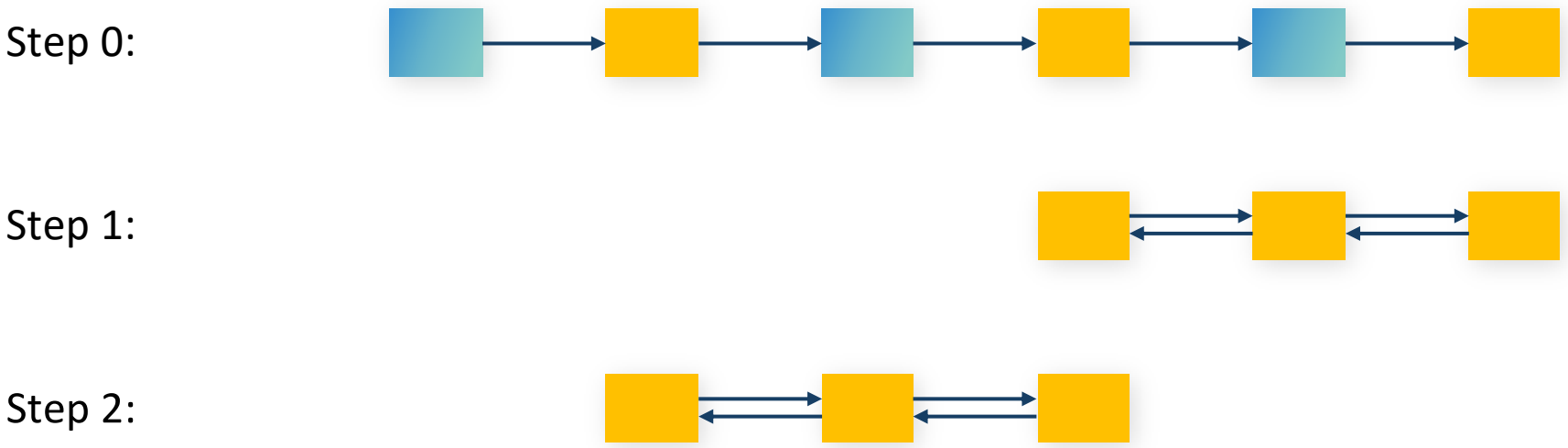


Because the need to keep intermediate value around (checkpoint) for the gradient steps.

Training a N-layer neural network would require $O(N)$ memory.

We will use the following simplified view to combine gradient and forward computation

Checkpointing Techniques in AD



- Only checkpoint colored nodes (step 0)
- Recompute the missing intermediate nodes in small segments (step 1, 2)

Sublinear Memory Cost

上海创智学院
Shanghai Innovation Institute

Forward computation

Gradient per segment
with re-computation

For a N layer neural network,
if we checkpoint every K layers

$$Memory\ cost = O\left(\frac{N}{K}\right) + O(K)$$

Checkpoint cost

Re-computation cost

Pick $K = \sqrt{N}$

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