Comprehensive introduction of DeepSeek-AI’s technical report：MLA&Load Balance

指导：朱嘉安师兄 主讲：任鑫

DeepSeek-V3 （Inference）

- ❑MLA（Multi-query Latent Attention）
- ❑Load Balance

Background of MLA

❑Memory Cost in Inference

- ❖Model Parameters (Fixed cost)
  - ➢ Fixed Cost (decided by model)
    - Model with 72B parameters cost about 144GB memory(BF16)

Background of MLA

❑Memory Cost in Inference

- ❖Model Parameters（Fixed cost）
  - ➢ Fixed Cost (decided by model)
    - Model with 72B parameters cost about 144GB memory(BF16)
- ❖KV-cache
  - ➢ Depends on **sequence length**、**batch size**、**precision**、**model architecture**

Background of MLA

❑Example

❖Model: Qwen 72B(ignore GQA, 80 layers （l） , 64 heads per layer(n_h)， vector dimension of each head is 128 (d_h))

➢ **KV-cache per Token:**

$$num_{kv} = 2 \times l \times n_h = 2 \times (80 \times 64)_{qwen\_72B} = 10240$$

Background of MLA

❑Example

❖Model: Qwen 72B(ignore GQA, 80 layers （l） , 64 heads per layer(n_h)， vector dimension of each head is 128 (d_h))

- **KV-cache per Token:**

$$num_{kv} = 2 \times l \times n_h = 2 \times (80 \times 64)_{qwen\_72B} = 10240$$

- **Use BF16 （Model Parameters cost 144G memory）：**

$$\mathrm{mem}_{kv}\_per_{token} = 2 \times num_{kv} \times d_h = 2 \times (10240 \times 128)_{qwen\_72B} = 2.62(MB)$$

Background of MLA

❑Example

❖Model: Qwen 72B(ignore GQA, 80 layers （l） , 64 heads per layer(n_h)， vector dimension of each head is 128 (d_h))

➢ **KV-cache per Token:**

$$num_{kv} = 2 \times l \times n_h = 2 \times (80 \times 64)_{qwen\_72B} = 10240$$

➢ **Use BF16 （Model Parameters cost 144G memory）：**

$$\mathrm{mem}_{kv}\_per_{token} = 2 \times num_{kv} \times d_h = 2 \times (10240 \times 128)_{qwen\_72B} = 2.62(MB)$$

➢ **Batch=1, Sequence Length = 2048:**

$$mem_{kv} = \mathrm{mem}_{kv}\_per_{token} \times B \times S = (2.62(MB) \times 1 \times 2048)_{\mathrm{query2B}} = 5.366GB$$

Background of MLA

❑Example

❖Model: Qwen 72B(ignore GQA, 80 layers （l）, 64 heads per layer(n_h)， vector dimension of each head is 128 (d_h))

➢ **KV-cache per Token:**

$$num_{kv} = 2 \times l \times n_h = 2 \times (80 \times 64)_{qwen\_72B} = 10240$$

➢ **Use BF16 （Model Parameters cost 144G memory）：**

$$\mathrm{mem}_{kv}\_per_{token} = 2 \times num_{kv} \times d_h = 2 \times (10240 \times 128)_{qwen\_72B} = 2.62(MB)$$

➢ **Batch=1, Sequence Length = 2048:**

$$mem_{kv} = \mathrm{mem}_{kv}\_per_{token} \times B \times S = (2.62(MB) \times 1 \times 2048)_{\mathrm{query2B}} = 5.366GB$$

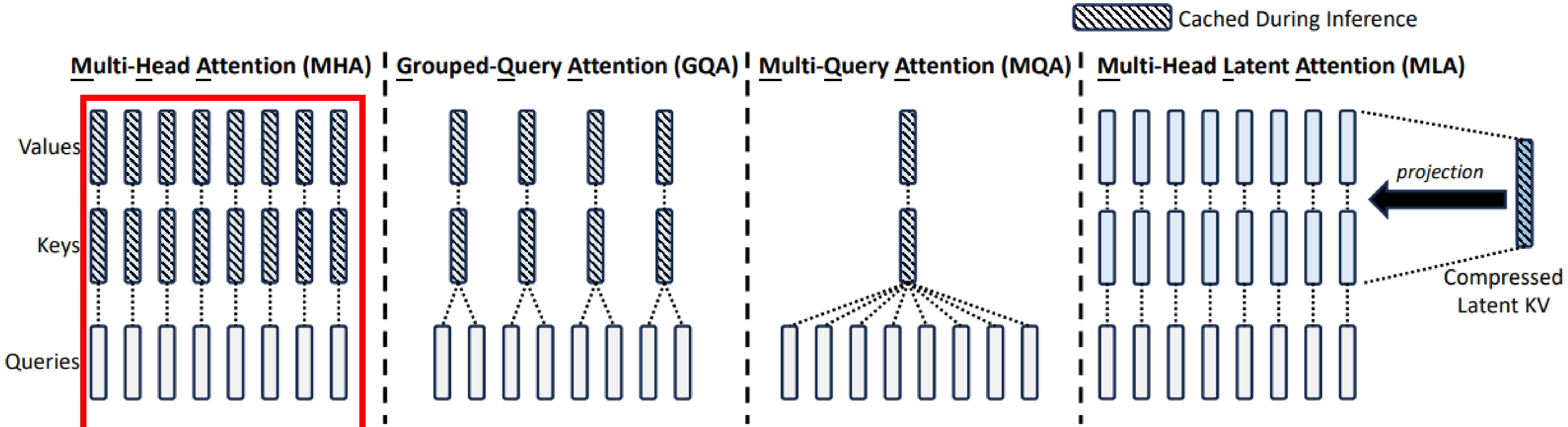
➢ **Batch=32, Sequence Length = 4096:**

$$mem_{kv} = \mathrm{mem}_{kv}\_per_{token} \times B \times S = (2.62(MB) \times 32 \times 4096)_{query2B} = 343.4GB$$

MLA

❑MHA vs GQA vs MQA vs MLA

n_h:heads number of each head d_h:vector dimension of each head l :layers n_g:number of groups

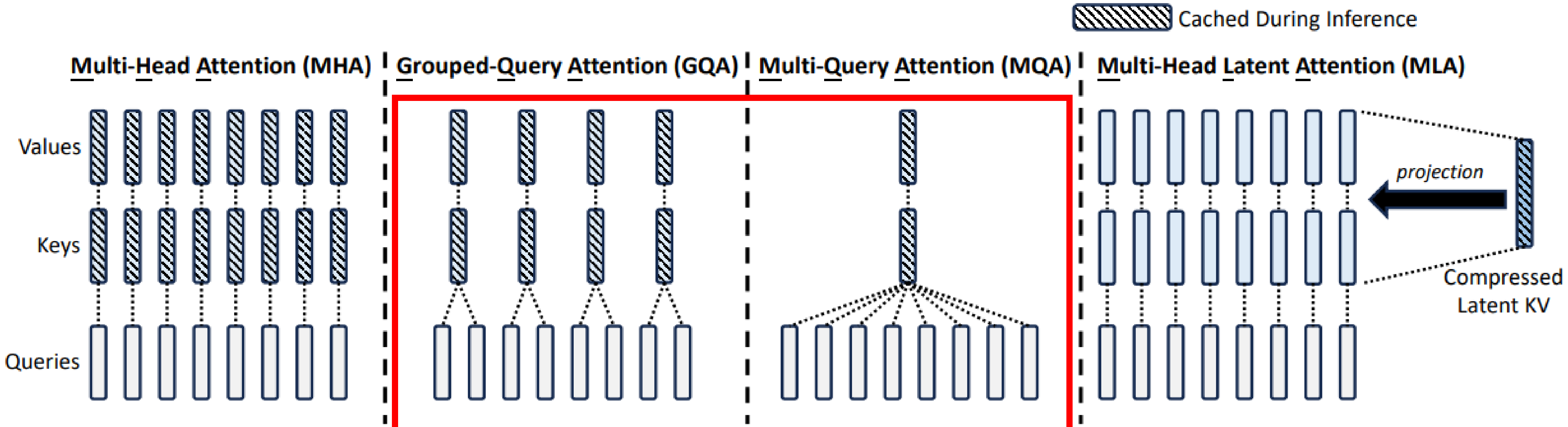


	MHA	GQA	MQA	MLA
KV-cache per Token	$2n_h d_h l$	$2n_g d_h l$	$2d_h l$	$\frac{9}{2}d_h l$
Performance	**Strong**	**Moderate**	**Weak**	**Stronger**

MLA

❑MHA vs GQA vs MQA vs MLA

n_h:heads number of each head d_h:vector dimension of each head l :layers n_g:number of groups

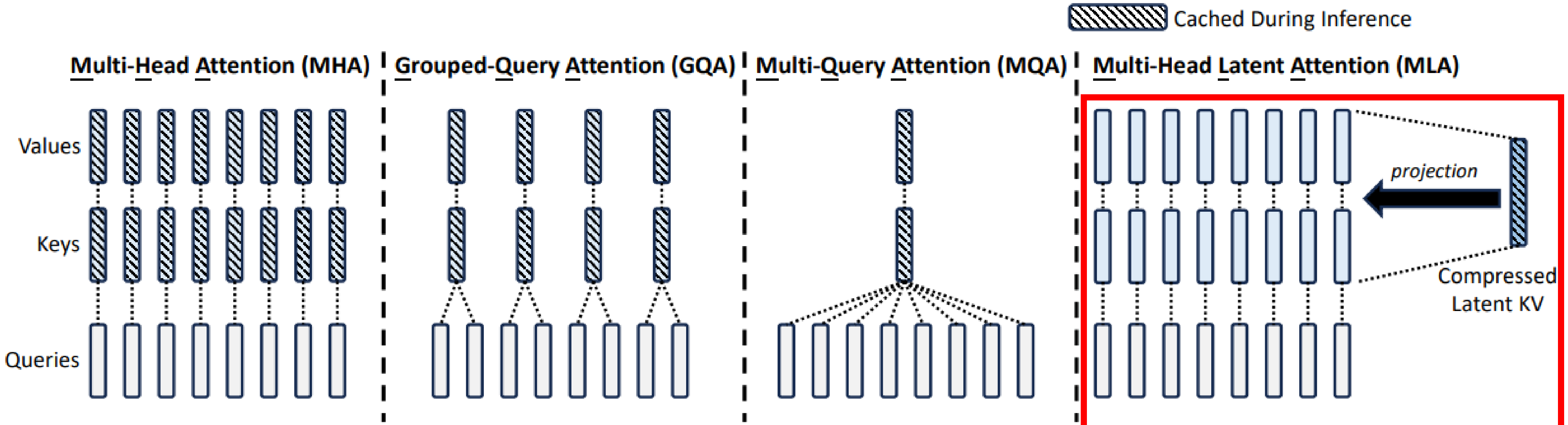


	MHA	GQA	MQA	MLA
KV-cache per Token	$2n_h d_h l$	$2n_g d_h l$	$2d_h l$	$\frac{9}{2}d_h l$
Performance	**Strong**	**Moderate**	**Weak**	**Stronger**

MLA

❑MHA vs GQA vs MQA vs MLA

n_h:heads number of each head d_h:vector dimension of each head l :layers n_g:number of groups

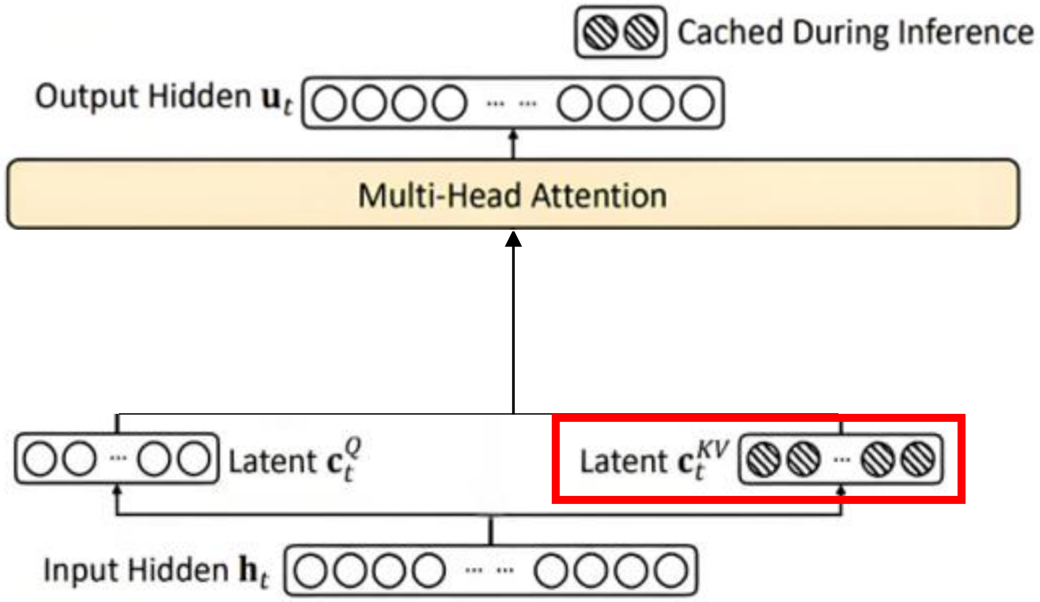


KV-cache per Token	$2n_h d_h l$	$2n_g d_h l$	$2d_h l$	$\frac{9}{2} d_h l$
Performance	**Strong**	**Moderate**	**Weak**	**Stronger**

KV Cache

❑KV Cache

❖Traditional KV cache requires caching the complete KV pairs

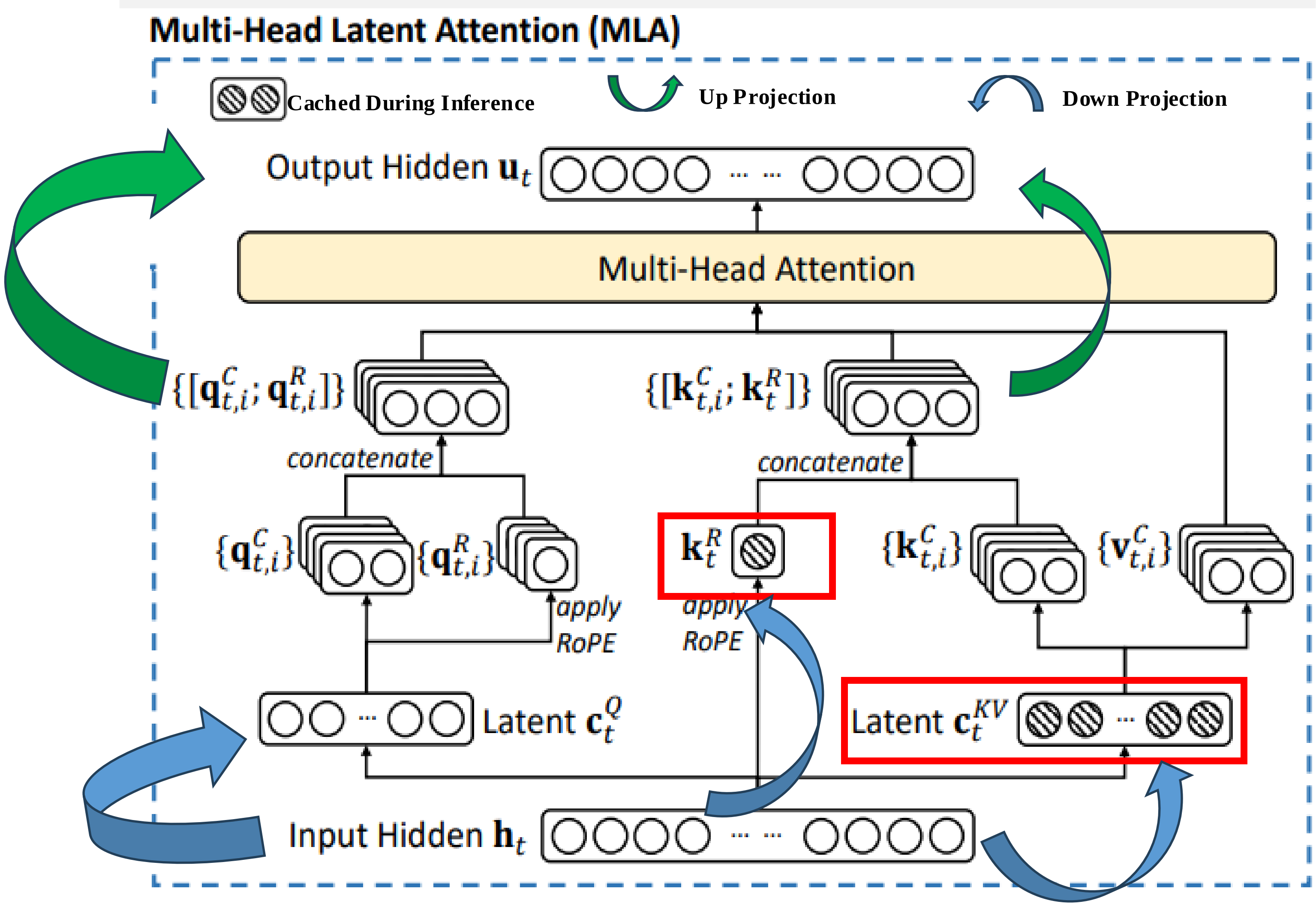


MLA

❑How to decrease KV Cache ?

❖Down Projection

- RoPE(Shared): k_t^R
- Key: c_t^{KV}



MLA

❑How to decrease KV Cache ?

❖Down Projection

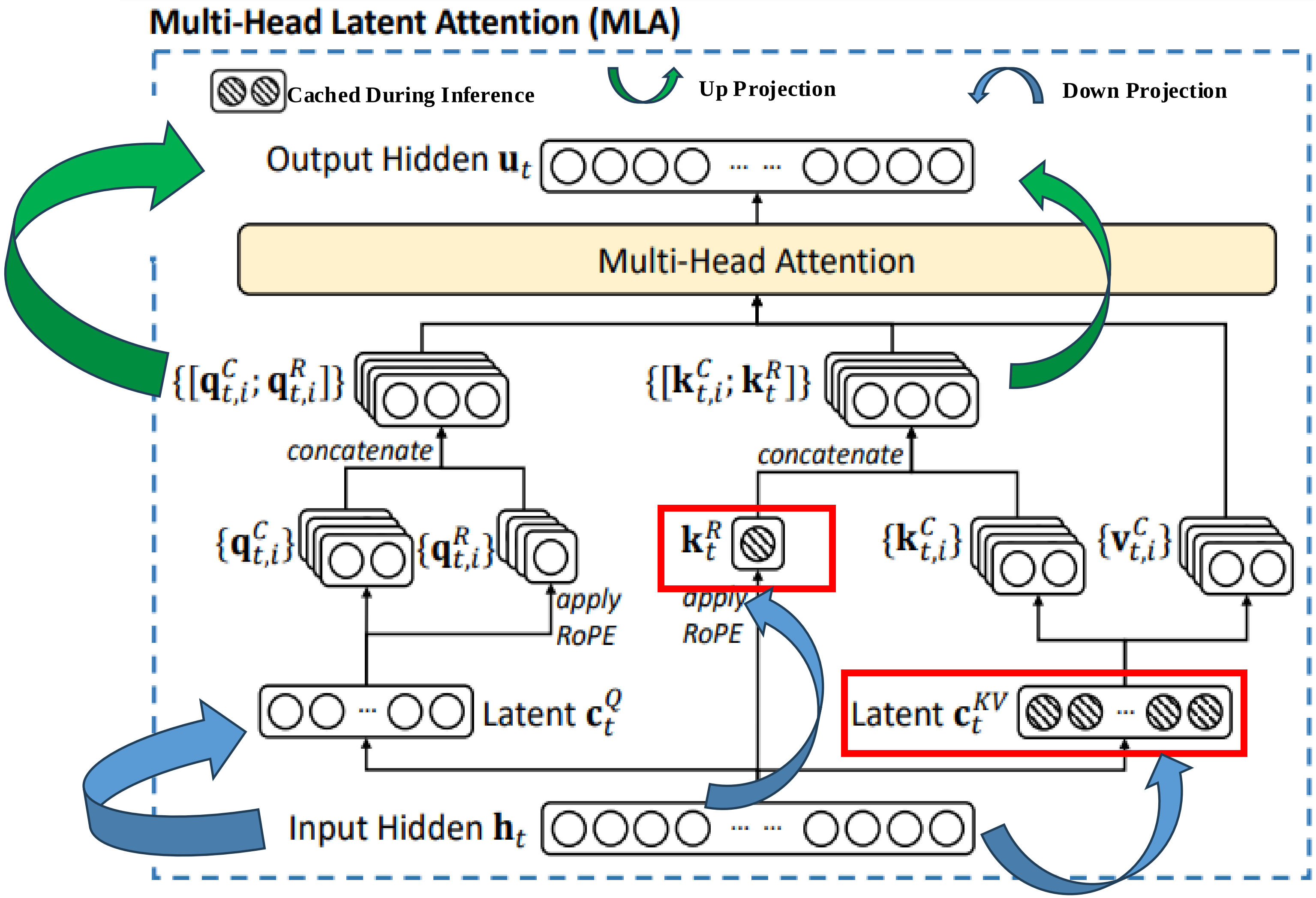
➢ RoPE(Shared): k_t^R

➢ Key: c_t^{KV}

❑Memory cost of per layer:

❖From $2n_h d_h$ to $k_t^R + c_t^{KV}$

$(\frac{1}{2}d_h + 4d_h)$



MLA

❑How to decrease KV Cache ?

❖Down Projection

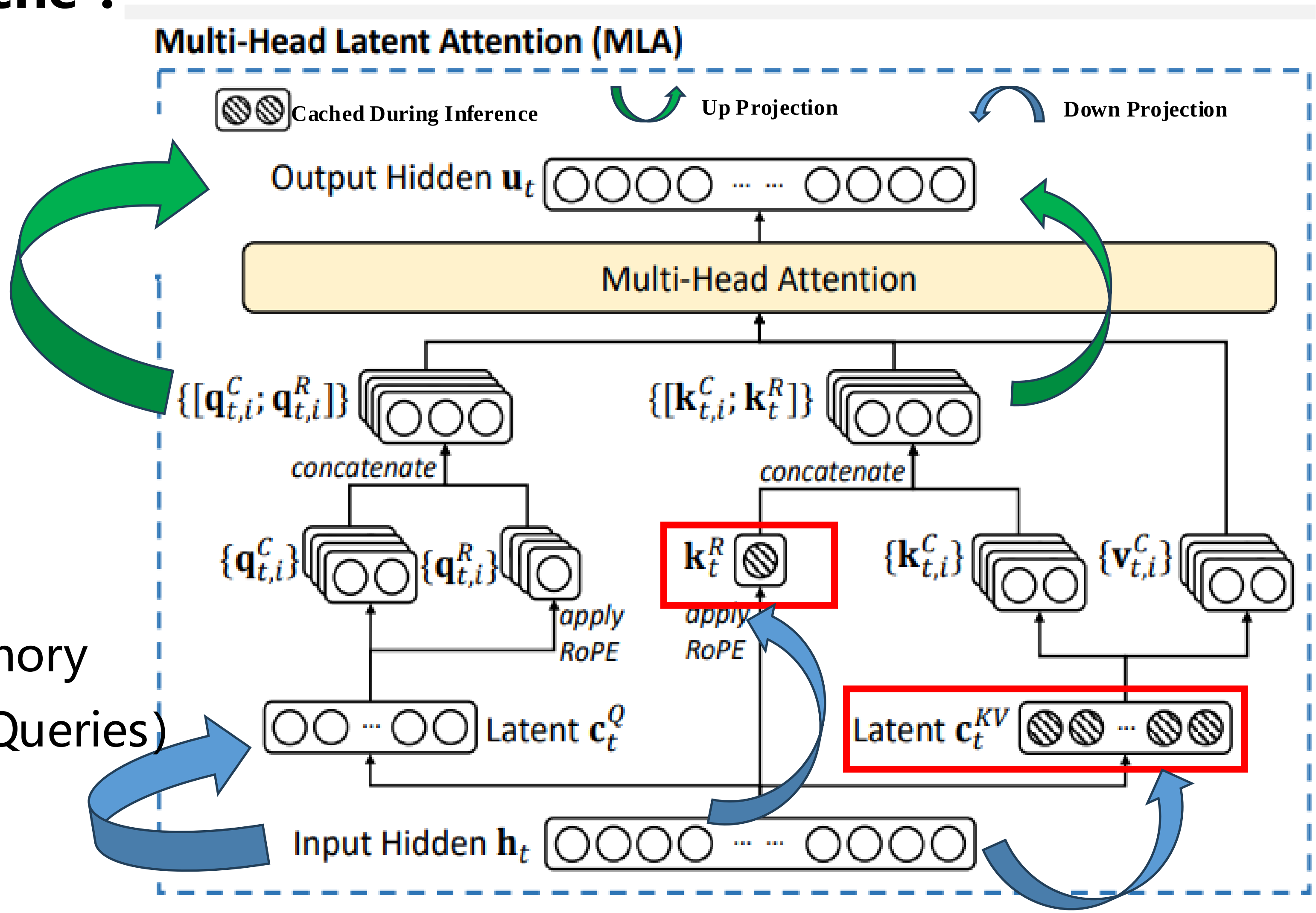
- ➢ RoPE(Shared): k_t^R
- ➢ Key: c_t^{KV}

❑Memory cost of per layer:

❖From $2n_h d_h$ to $k_t^R + c_t^{KV}$ ($\frac{1}{2} d_h + 4d_h$)

❑More benefits:

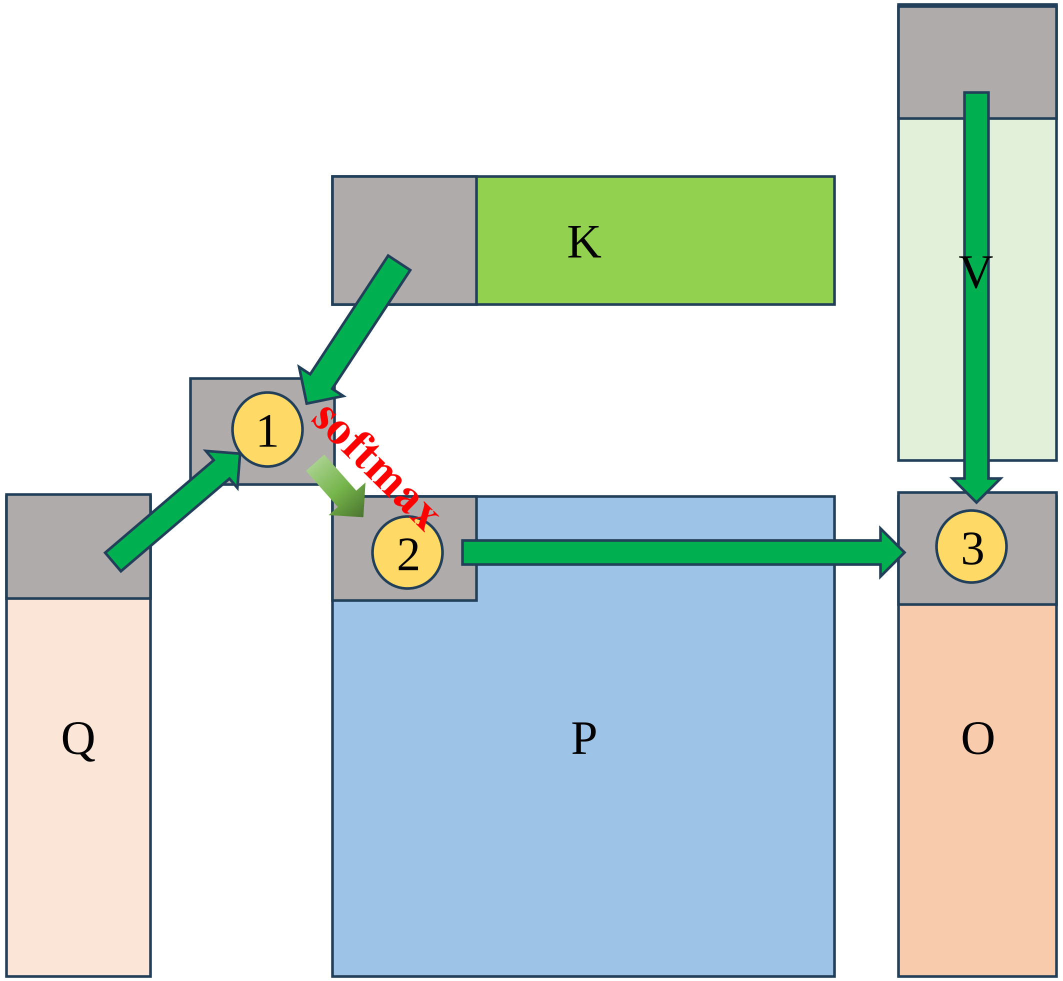
❖Reduce the activation memory during training(Attention Queries)



FLASH MLA

❑An efficient MLA decoding kernel for Hopper GPUs: Flash MLA

- ❖Background: Fully utilize SRAM
  - ➢ SRAM: Fast but small
  - ➢ HBM: Slow but big enough



FLASH MLA

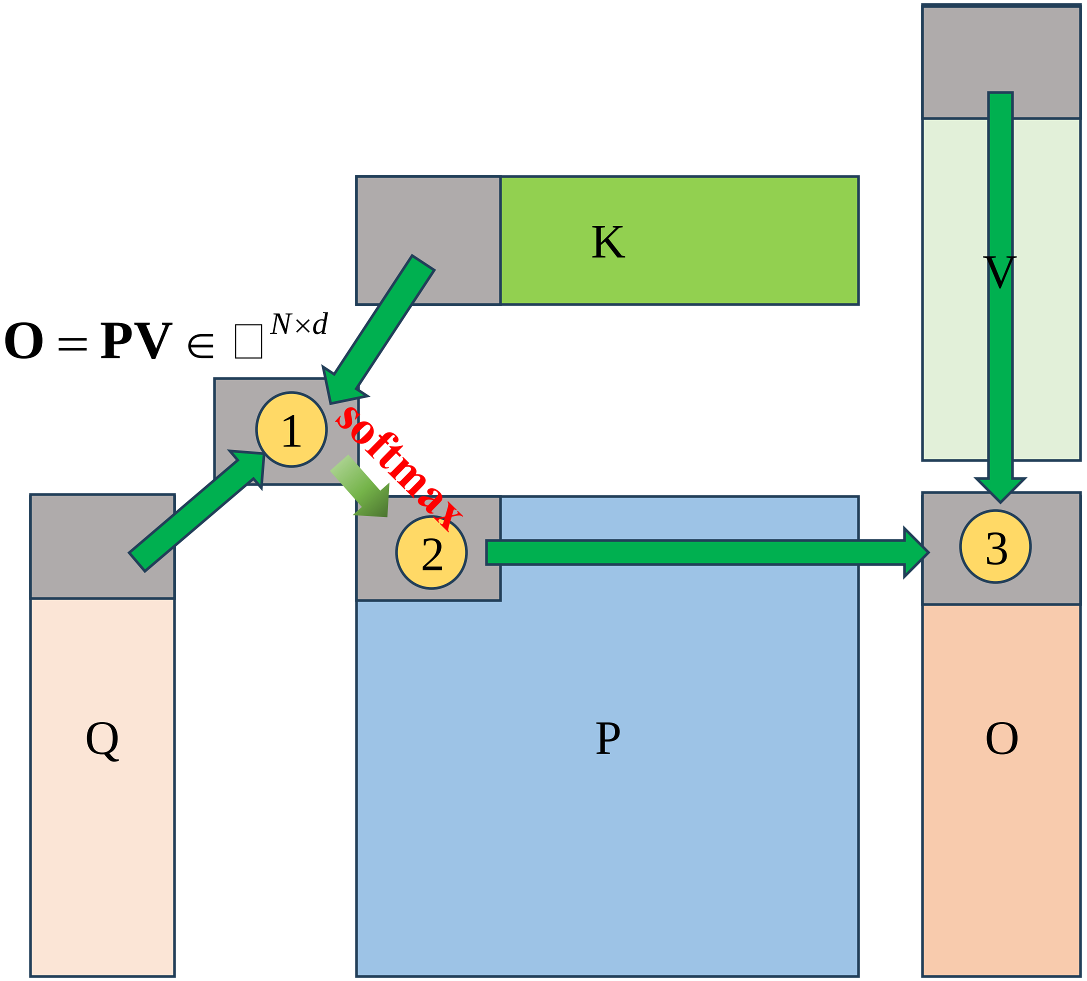
❑An efficient MLA decoding kernel for Hopper GPUs: Flash MLA

❖Background: Fully utilize SRAM

- ➢ SRAM: Fast but small
- ➢ HBM: Slow but big enough

❖Traditional attention computation :

- ➢ $\mathbf{S} = \mathbf{QK}^{\top} \in \mathbb{R}^{N\times N}$, $\mathbf{P} = \mathrm{softmax}(\mathbf{S}) \in \mathbb{R}^{N\times N}$, $\mathbf{O} = \mathbf{PV} \in \mathbb{R}^{N\times d}$



DeepSeek-V3

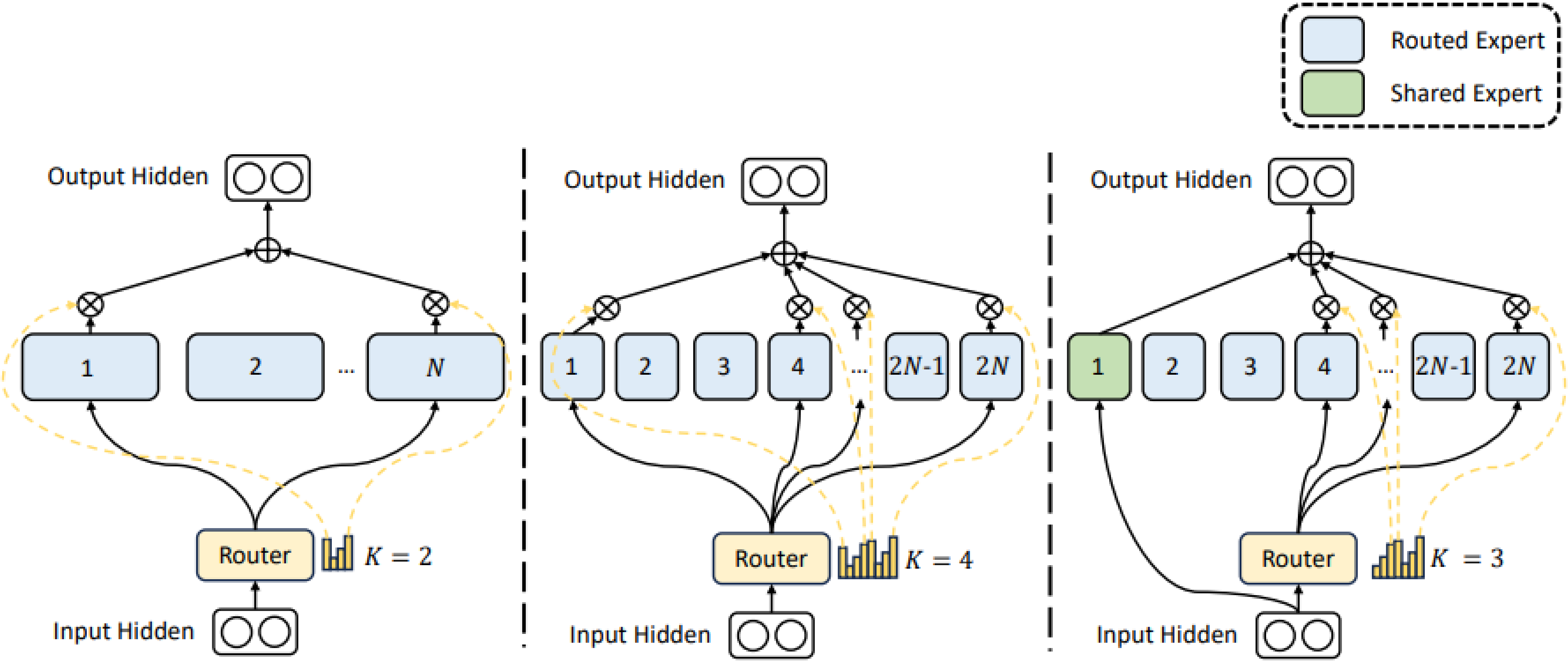
- ❑MLA（Multi-query Latent Attention)
- ❑**Load Balance**

Background of Load Balance

- ❑DeepSeekMoE architecture
- ❑Problems of Unbalanced Expert load

Background of Load Balance

❑DeepSeekMoE architecture



(a) Conventional Top-2 Routing → **(b) + Fine-grained Expert Segmentation** → **(c) + Shared Expert Isolation (DeepSeekMoE)**

Background of Load Balance

- ❑DeepSeekMoE architecture
- ❑**Problems of Unbalanced Expert load**

Background of Load Balance

❑Problems of Unbalanced Expert load

- ❖Expert Parallelism(EP)
  - ➢ A distributed strategy that assigns experts to multiple devices

Background of Load Balance

❑Problems of Unbalanced Expert load

- ❖Expert Parallelism(EP)
  - ➢ A distributed strategy that assigns experts to multiple devices
- ❖Imbalanced Communication with EP Enabled
  - ➢ Frequently selected experts to handle more data transfer, creating communication bottlenecks

Background of Load Balance

❑Problems of Unbalanced Expert load

- ❖Expert Parallelism(EP)
  - ➢ A distributed strategy that assigns experts to multiple devices
- ❖Imbalanced Communication with EP Enabled
  - ➢ Frequently selected experts to handle more data transfer, creating communication bottlenecks
- ❖Imbalanced Computation with EP Enabled:
  - ➢ GPUs with heavily loaded experts take longer to compute, reducing overall training efficiency and GPU utilization

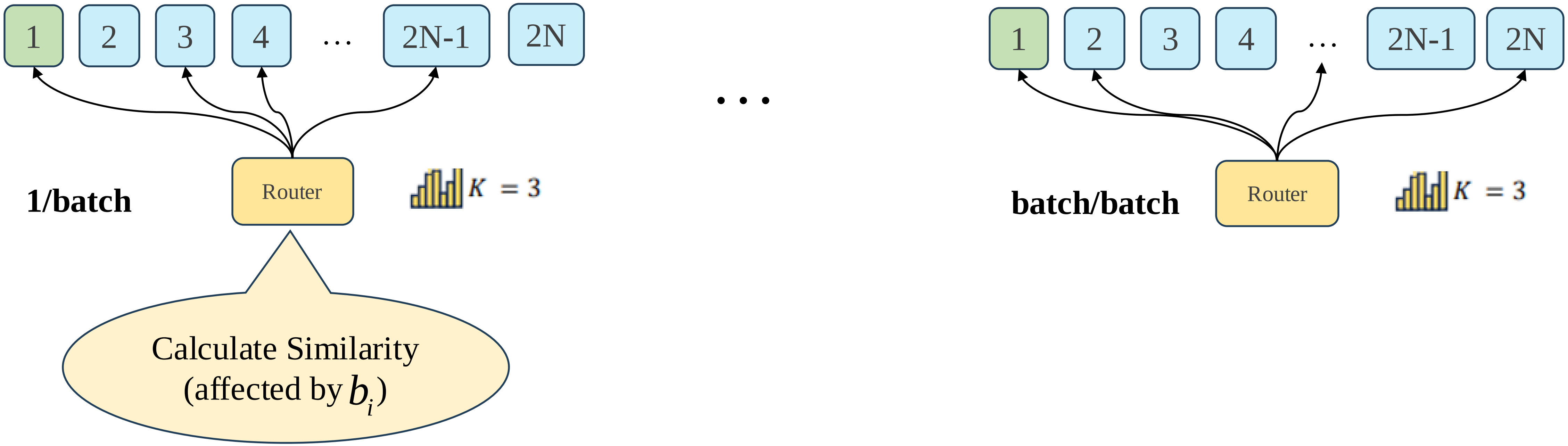
Load Balance in DeepSeek V3

- ❑ **Auxiliary-Loss-Free Load Balancing**
- ❑ Complementary Sequence-Wise Auxiliary Loss
- ❑ Others

Load Balance in DeepSeek V3

❑Auxiliary-Loss-Free Load Balancing

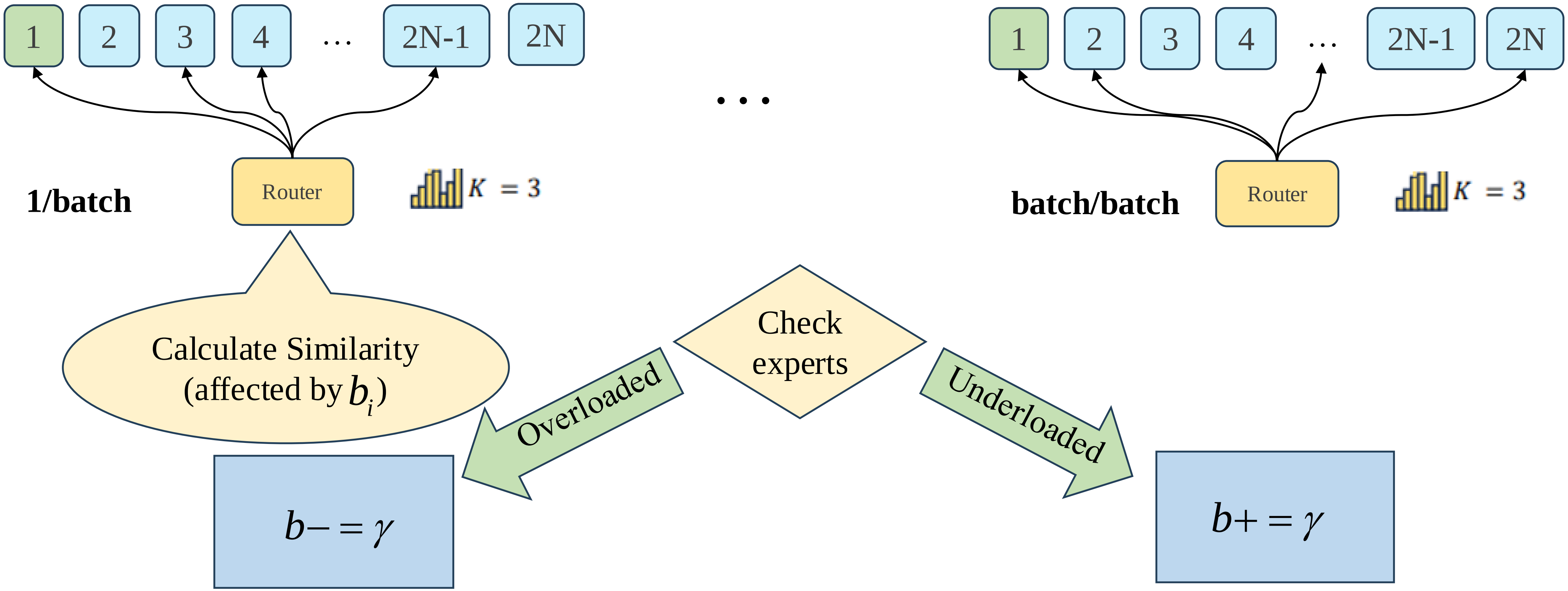
- ❖Reduce the weight of experts that appear frequently
- ❖Increase the weight of experts that appear infrequently



Load Balance in DeepSeek V3

❑Auxiliary-Loss-Free Load Balancing

- ❖Reduce the weight of experts that appear frequently
- ❖Increase the weight of experts that appear infrequently



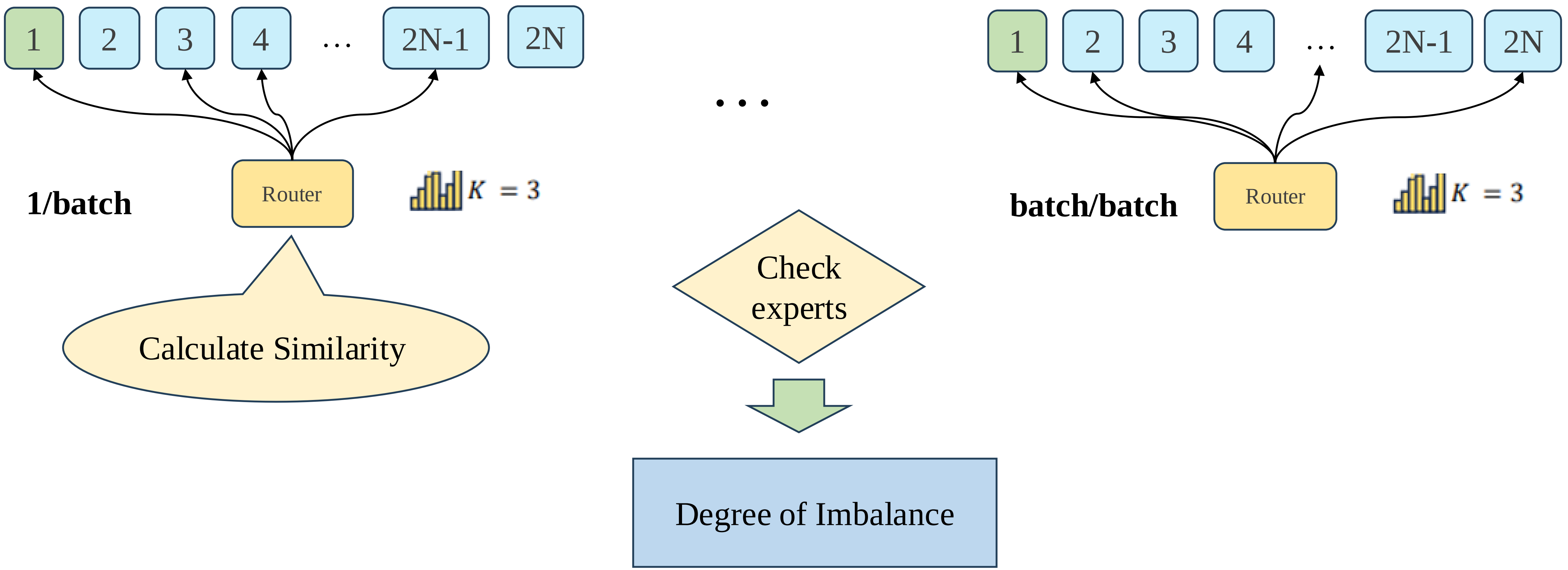
Load Balance in DeepSeek V3

- ❑Auxiliary-Loss-Free Load Balancing
- ❑**Complementary Sequence-Wise Auxiliary Loss**
- ❑Others

Load Balance in DeepSeek V3

❑Complementary Sequence-Wise Auxiliary Loss

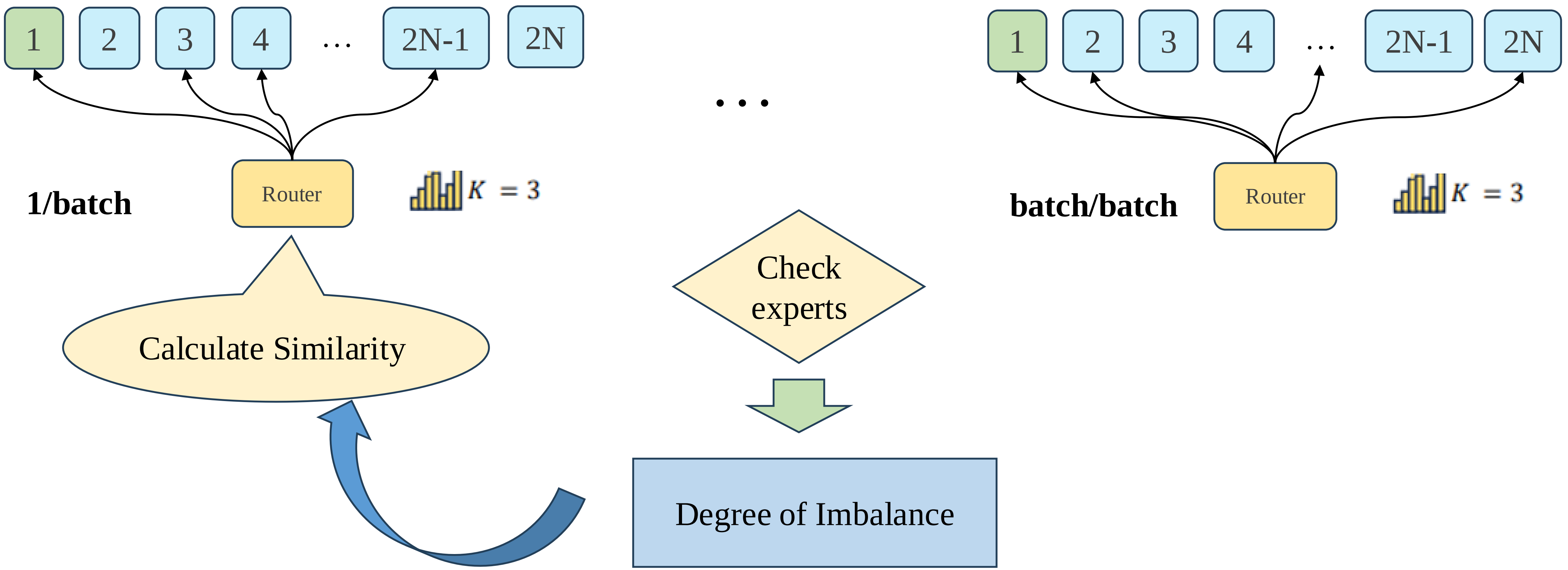
❖Calculate **the deviation of each expert** from the most balanced scenario to make the router distribute more evenly in the next batch



Load Balance in DeepSeek V3

❑Complementary Sequence-Wise Auxiliary Loss

❖Calculate **the deviation of each expert** from the most balanced scenario to make the router distribute more evenly in the next batch



Load Balance in DeepSeek V3

- ❑Auxiliary-Loss-Free Load Balancing
- ❑Complementary Sequence-Wise Auxiliary Loss
- ❑**Others**

Load Balance in DeepSeek V3

❑Others

❖Node-Limited Routing

- **Limit each token to be sent to at most M nodes**

❖No Token-Dropping

- **Due to the effective load balancing strategy, DeepSeek-V3 does not drop any tokens during either training or inference**