高效注意力

清华大学计算机系 陈键飞

《大模型计算》课程团队

计算量/访存量

❖ 考虑8B模型：seqlen=8192，head_dim=128

```
S = q * k.t() / sqrt(d)
```

计算量8192*8192*128*2/2 = 8.59Gflops (causal)

访存量8192*128*2*2bytes + 8192*8192*4bytes = 0.254GB

```
P = softmax(S)
```

计算量可忽略不计

访存量8192*8192*2*4bytes = 0.5GB （一读一写）

```
o = P * v
```

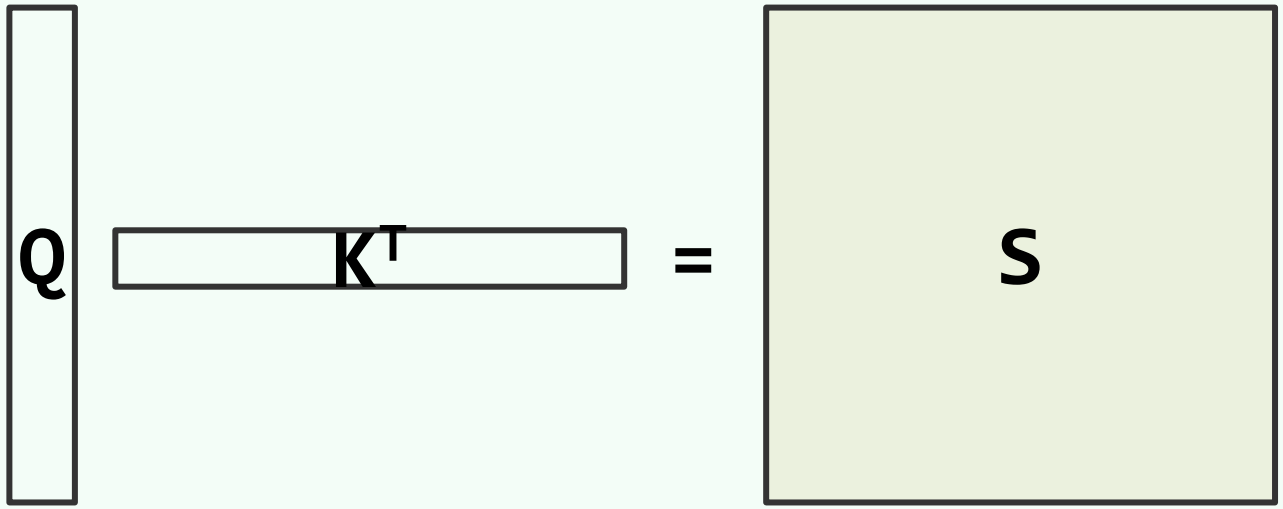
计算访存量与计算S相等

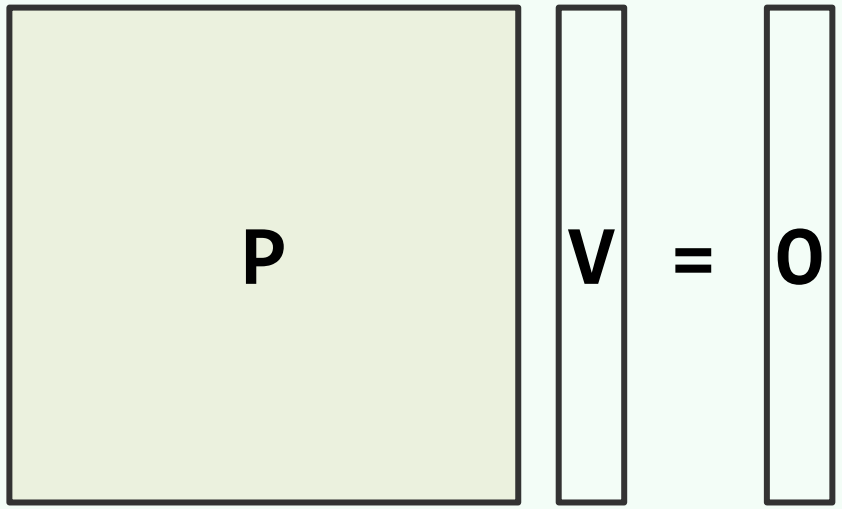
计算量=17.2G，访存量=1.008GB，**计算访存比=17 flops/Byte**

在RTX 4090上，如果打满1008GB/s带宽（FP16），则输出算力=17.2Tflops

理论峰值165Tflops

问题





算子的输入输出（QKVO）都很小

但中间结果（SP）很大

思路

- QKV一起往GPU里喂
- SP不出芯片，直接吐出O

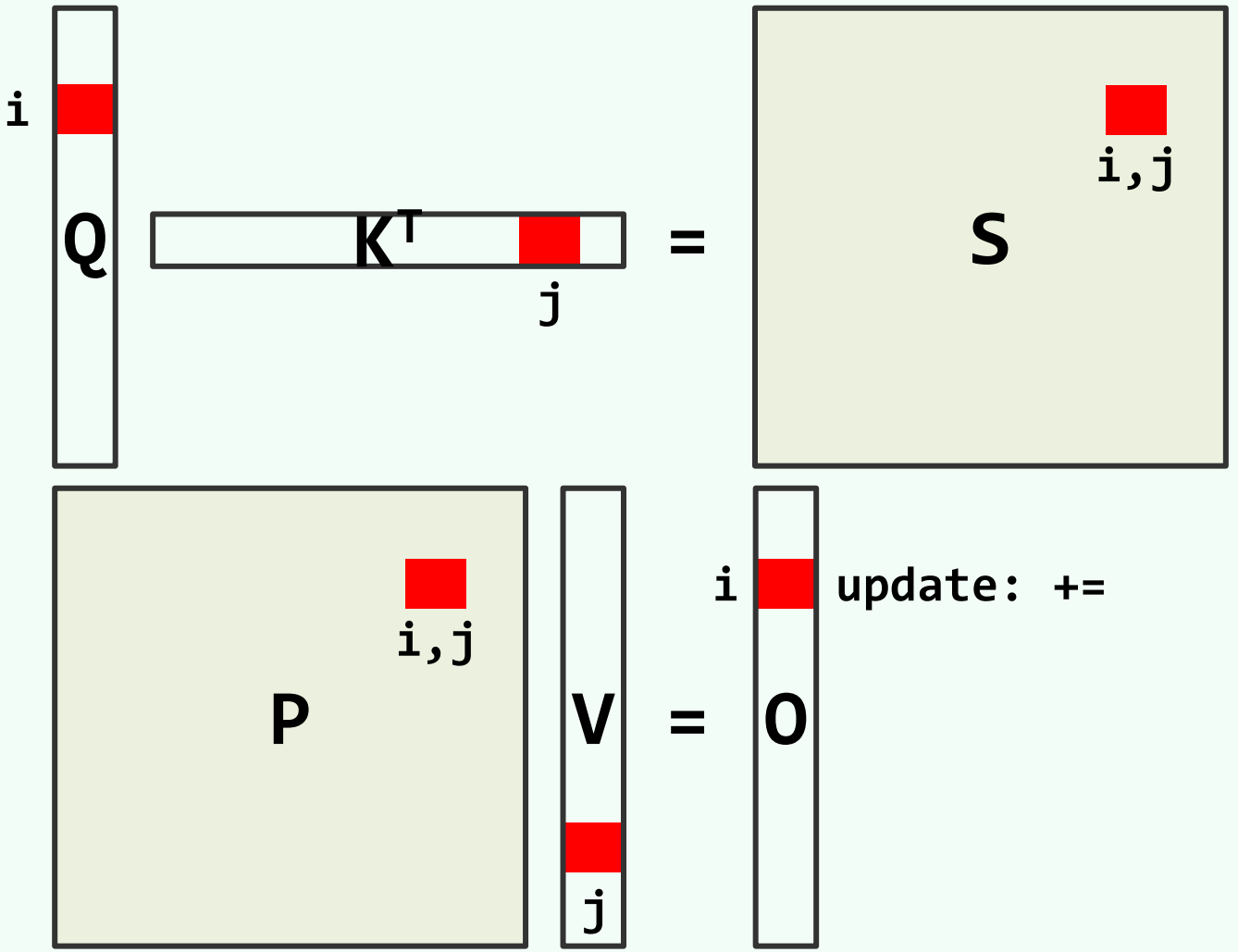
$$O_i \mathrel{+}= P_{ij} V_j$$

$$P_{ij} = ?\left(S_{ij}\right)$$

$$S_{ij} = Q_i K_j^T$$

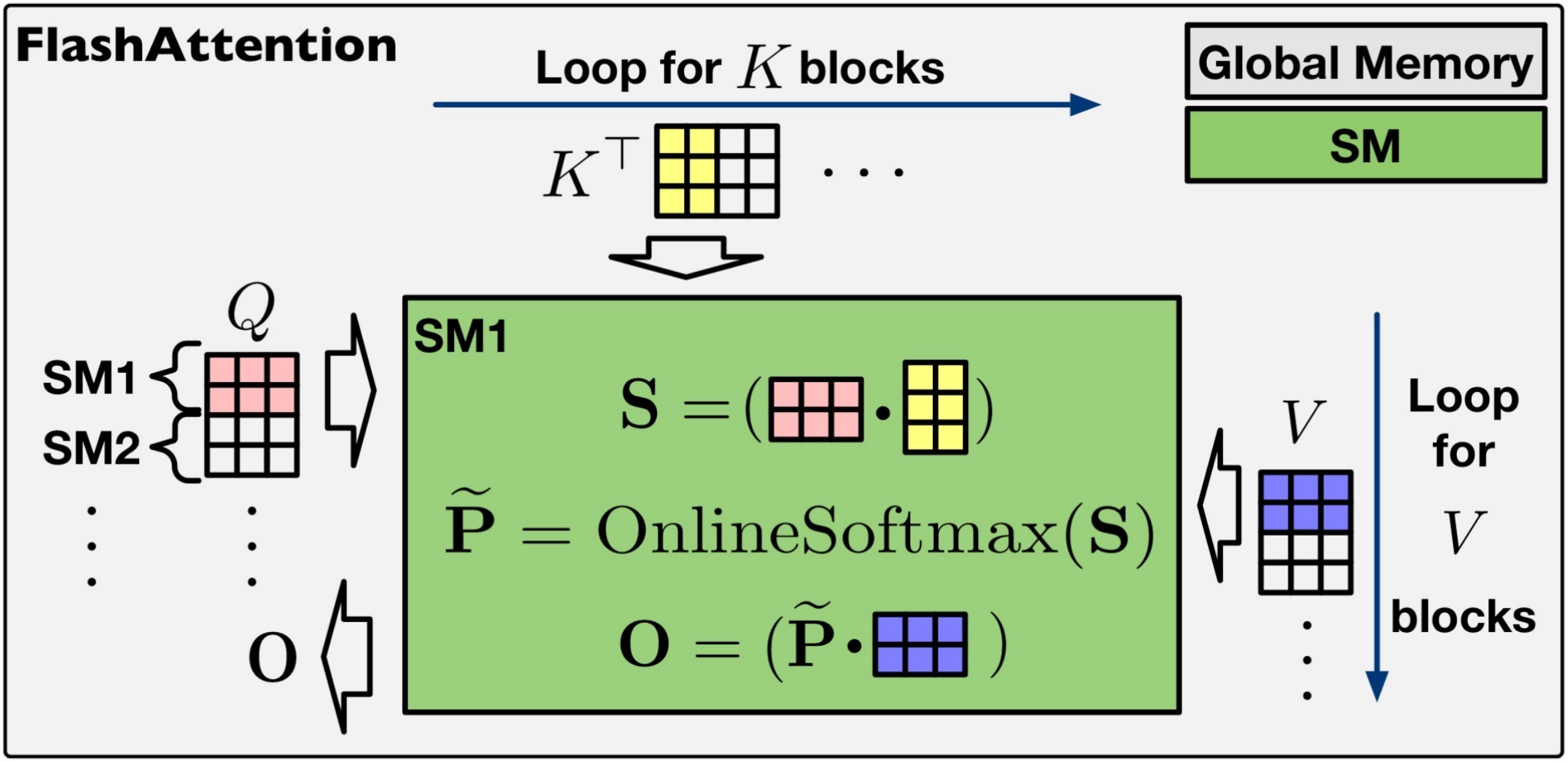
唯一的问题：softmax

P_{ij}**不仅取决于**S_{ij}**，还取决于其他tile**



Online softmax

- 要算$O = \sum_j P_j V_j = \sum_j \left(\frac{\exp(S_j)}{\sum_k \exp(S_k)} \right) V_j$
- 问题：$\sum_k \exp(S_k)$事先不知道
- 解法：分成两个求和$O = \frac{\left(\sum_j \exp(S_j) V_j \right)}{\sum_k \exp(S_k)}$
- 循环元素j：维护两个求和$O \mathrel{+}= \exp(S_j) V_j$和$\ell \mathrel{+}= \exp(S_j)$
- 最后输出O/ℓ
- 期间O和ℓ可以同时缩放避免softmax带来的数值问题
- 额外维护$m = \max\{m, S_j\}$
- 将两个求和同时缩放到m：$O = O/\exp(m^{new} - m^{old})$，$\ell = \ell/\exp(m^{new} - m^{old})$


FlashAttention
Loop for K blocks
Global Memory
SM
$K^\top$
Q
SM1
SM2
SM1
$\mathbf{S} = (\quad \cdot \quad)$
$\widetilde{\mathbf{P}} = \mathrm{OnlineSoftmax}(\mathbf{S})$
$\mathbf{O} = (\widetilde{\mathbf{P}} \cdot \quad)$
V
Loop for V blocks
O

Algorithm 1 FLASHATTENTION-2 forward pass

Require: Matrices $\mathbf{Q}, \mathbf{K}, \mathbf{V} \in \mathbb{R}^{N \times d}$ in HBM, block sizes B_c, B_r.

1: Divide $\mathbf{Q}$ into $T_r = \left\lceil \frac{N}{B_r} \right\rceil$ blocks $\mathbf{Q}_1, \dots, \mathbf{Q}_{T_r}$ of size $B_r \times d$ each, and divide $\mathbf{K}, \mathbf{V}$ in to $T_c = \left\lceil \frac{N}{B_c} \right\rceil$ blocks $\mathbf{K}_1, \dots, \mathbf{K}_{T_c}$ and $\mathbf{V}_1, \dots, \mathbf{V}_{T_c}$, of size $B_c \times d$ each.

2: Divide the output $\mathbf{O} \in \mathbb{R}^{N \times d}$ into T_r blocks $\mathbf{O}_i, \dots, \mathbf{O}_{T_r}$ of size $B_r \times d$ each, and divide the logsumexp L into T_r blocks $L_i, \dots, L_{T_r}$ of size B_r each.

3: **for** $1 \le i \le T_r$ **do**

4: Load $\mathbf{Q}_i$ from HBM to on-chip SRAM.

5: On chip, initialize $\mathbf{O}_i^{(0)} = (0)_{B_r \times d} \in \mathbb{R}^{B_r \times d}, \ell_i^{(0)} = (0)_{B_r} \in \mathbb{R}^{B_r}, m_i^{(0)} = (-\infty)_{B_r} \in \mathbb{R}^{B_r}$.

6: **for** $1 \le j \le T_c$ **do**

7: Load $\mathbf{K}_j, \mathbf{V}_j$ from HBM to on-chip SRAM.

8: On chip, compute $\mathbf{S}_i^{(j)} = \mathbf{Q}_i \mathbf{K}_j^T \in \mathbb{R}^{B_r \times B_c}$.

9: On chip, compute $m_i^{(j)} = \max(m_i^{(j-1)}, \text{rowmax}(\mathbf{S}_i^{(j)})) \in \mathbb{R}^{B_r}$, $\tilde{\mathbf{P}}_i^{(j)} = \exp(\mathbf{S}_i^{(j)} - m_i^{(j)}) \in \mathbb{R}^{B_r \times B_c}$ (pointwise), $\ell_i^{(j)} = e^{m_i^{j-1} - m_i^{(j)}} \ell_i^{(j-1)} + \text{rowsum}(\tilde{\mathbf{P}}_i^{(j)}) \in \mathbb{R}^{B_r}$.

10: On chip, compute $\mathbf{O}_i^{(j)} = \text{diag}(e^{m_i^{(j-1)} - m_i^{(j)}})^{-1} \mathbf{O}_i^{(j-1)} + \tilde{\mathbf{P}}_i^{(j)} \mathbf{V}_j$.

11: **end for**

12: On chip, compute $\mathbf{O}_i = \text{diag}(\ell_i^{(T_c)})^{-1} \mathbf{O}_i^{(T_c)}$.

13: On chip, compute $L_i = m_i^{(T_c)} + \log(\ell_i^{(T_c)})$.

14: Write $\mathbf{O}_i$ to HBM as the i-th block of $\mathbf{O}$.

15: Write L_i to HBM as the i-th block of L.

16: **end for**

17: Return the output $\mathbf{O}$ and the logsumexp L.

FlashAttention: 性能分析

- 考虑8B模型: seqlen=8192, head_dim=128
- 假设blocksize是128(Q) * 128(D) * 64(K)
- 计算问题分为64*128个tile
- 每个tile需要读取KV: 2*64*128*2 = 0.03125MB
- 但是每个tile要算两个矩阵乘: 2*128*128*64*2 = 4Mflops
- 计算访存比 = 128 flops / Byte

- causal attention只需要算一半的tile，共计128MB的IO（节省为1/8）
- 如果打满1008GB显存带宽，实际能跑到139Tflops
- 实测能跑到170+Tflops （命中L2? + GPU turbo）

结构参数

模型版本	参数量	层数	Hidden Dim	Heads	FFN Dim	Max Seq Len	Vocab Size
LLaMA-1	7B	32	4096	32	11008	2048	32000
LLaMA-1	13B	40	5120	40	13824	2048	32000
LLaMA-1	33B	60	6656	52	17920	2048	32000
LLaMA-1	65B	80	8192	64	22016	2048	32000
LLaMA-2	7B	32	4096	32	11008	4096	32000
LLaMA-2	13B	40	5120	40	13824	4096	32000
LLaMA-2	70B	80	8192	64	28672	4096	32000
LLaMA-3	1B	24	2048	16	5504	2048	128256
LLaMA-3	8B	32	4096	32	14336	8192	128256
LLaMA-3	70B	80	8192	64	28672	8192	128256
LLaMA-3	405B	128	12288	96	49152	128K	128256

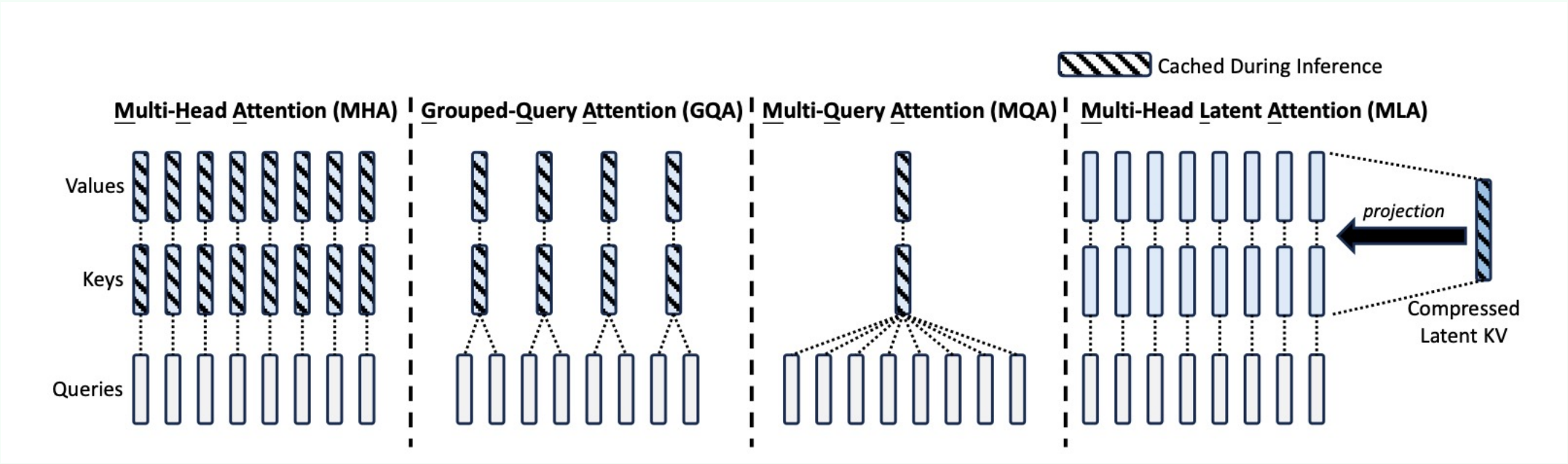
MHA, MQA, GQA

问题：Kvcache太大。DeepSeekv3使用L=61, numheads=128, Kdim=192, Vdim=128

使用MHA需要约2.4MB / token

128K上下文越需要298GB KVCache

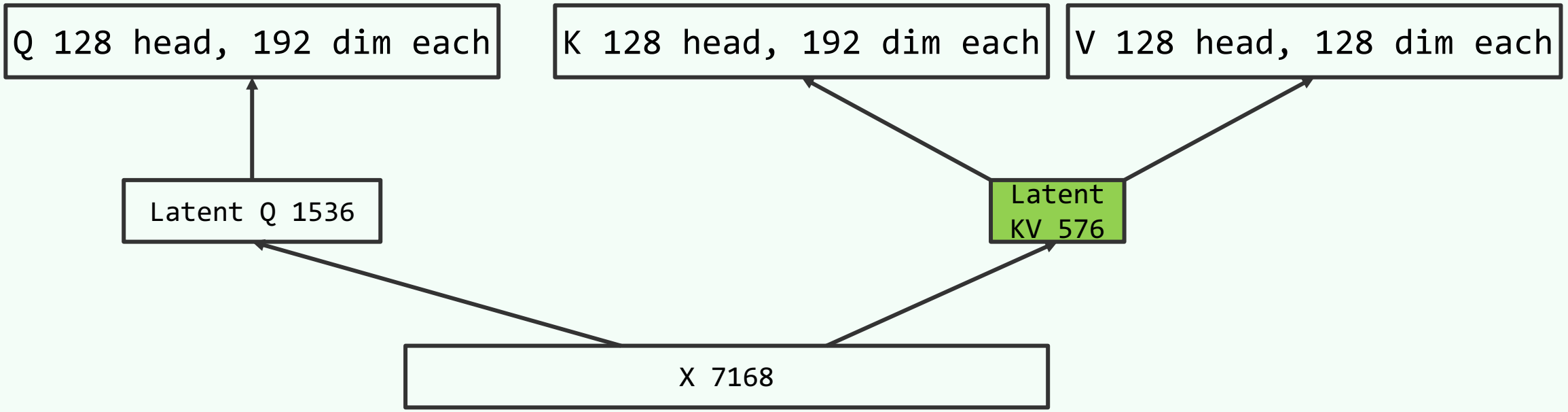
思路：减少Kvhead的数量



Multi-latent attention

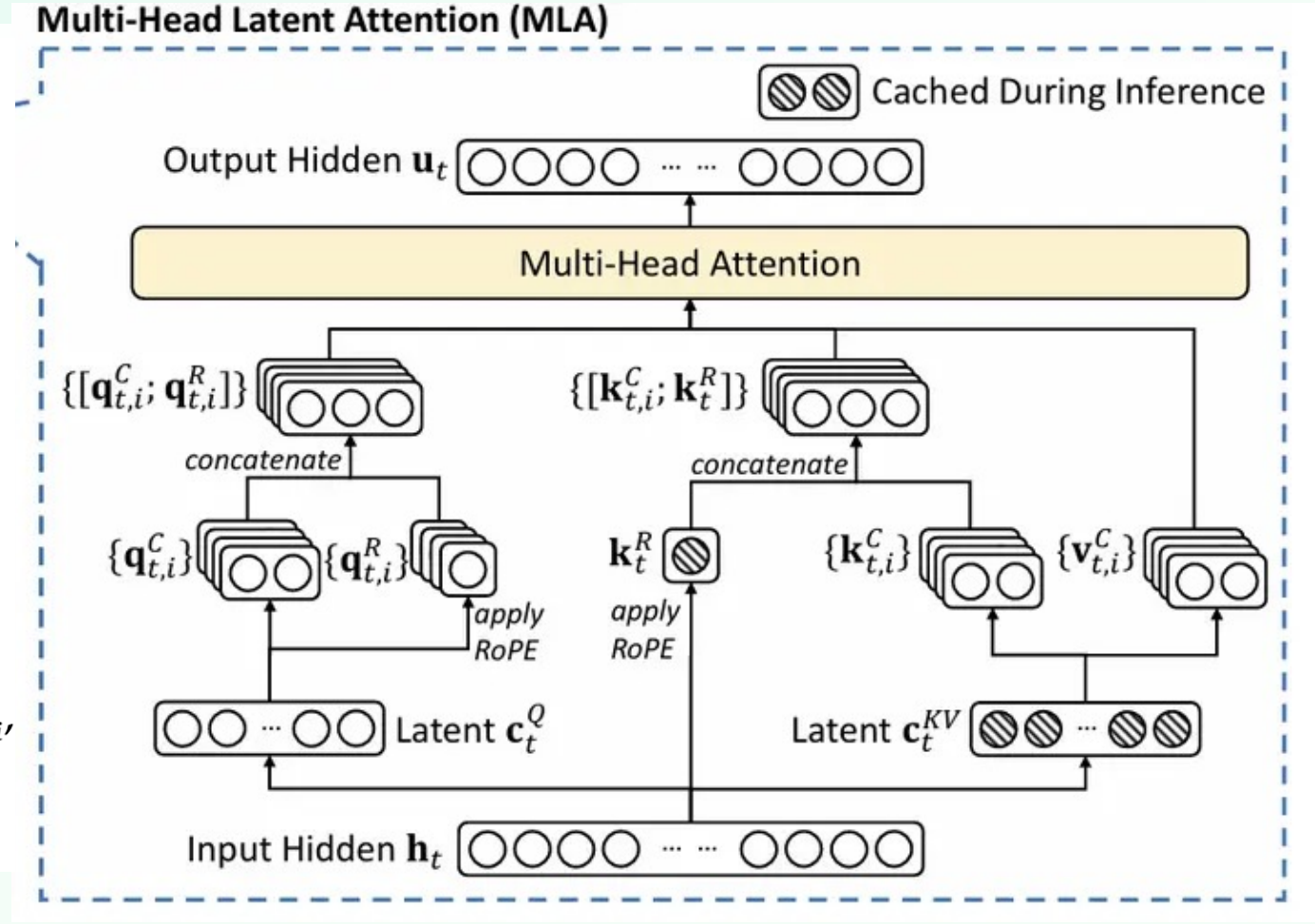
- 思路：将attention projection变成MLP，增加“压缩-解压”，引入瓶颈层
- Kvcache：576*61 = 34KB / token，71倍压缩率，128K上下文仅需4.2GB！
- 参数量：7168*(1536+576) (compress) + 1536*128*192 (Q up)

  + 7168*64(KVrope)+576*128*256 (KVup) + 128*128*7168 (Oproj) = 11.56B



MLA

$$
\begin{aligned}
\mathbf{c}_t^Q &= W^{DQ}\mathbf{h}_t, \\
[\mathbf{q}_{t,1}^C; \mathbf{q}_{t,2}^C; \ldots; \mathbf{q}_{t,n_h}^C] = \mathbf{q}_t^C &= W^{UQ}\mathbf{c}_t^Q, \\
[\mathbf{q}_{t,1}^R; \mathbf{q}_{t,2}^R; \ldots; \mathbf{q}_{t,n_h}^R] = \mathbf{q}_t^R &= \mathrm{RoPE}(W^{QR}\mathbf{c}_t^Q), \\
\mathbf{q}_{t,i} &= [\mathbf{q}_{t,i}^C; \mathbf{q}_{t,i}^R], \\
\boxed{\mathbf{c}_t^{KV}} &= W^{DKV}\mathbf{h}_t, \\
[\mathbf{k}_{t,1}^C; \mathbf{k}_{t,2}^C; \ldots; \mathbf{k}_{t,n_h}^C] = \mathbf{k}_t^C &= W^{UK}\mathbf{c}_t^{KV}, \\
\boxed{\mathbf{k}_t^R} &= \mathrm{RoPE}(W^{KR}\mathbf{h}_t), \\
\mathbf{k}_{t,i} &= [\mathbf{k}_{t,i}^C; \mathbf{k}_t^R], \\
[\mathbf{v}_{t,1}^C; \mathbf{v}_{t,2}^C; \ldots; \mathbf{v}_{t,n_h}^C] = \mathbf{v}_t^C &= W^{UV}\mathbf{c}_t^{KV}, \\
\mathbf{o}_{t,i} &= \sum_{j=1}^{t} \mathrm{Softmax}_j\left(\frac{\mathbf{q}_{t,i}^T \mathbf{k}_{j,i}}{\sqrt{d_h + d_h^R}}\right)\mathbf{v}_{j,i}^C, \\
\mathbf{u}_t &= W^O[\mathbf{o}_{t,1}; \mathbf{o}_{t,2}; \ldots; \mathbf{o}_{t,n_h}],
\end{aligned}
$$



MLA解码：计算QK

- 训练模式：直接暴力解压QKV并计算FlashAttention
- 推理模式：不显式解压，直接通过矩阵乘法完成解码计算

dim*512

1*1536 1536*dim 512*seqlen

$$q_{t,i}K_i^T = \left(c_t^Q\right)\ \left(W_i^{UQ}\right)^T W_i^{UK}\left(c^{KV}\right)^T$$

1*dim dim*seqlen

$$= l_{t,i}^Q\left(c^{KV}\right)^T$$

1*512 512*seqlen

所有的head共享，只要读一次

t: query token id

j: kv token id

i: head id

凑成矩阵

#heads * seqlen #heads * 512

$$S_t = \ell_t^Q\left(c^{KV}\right)^T$$

512 * seqlen

- 因为latent KV是所有head共享的
- 所以一个矩阵乘法可以同时把所有head的S（矩阵）一起算出来
- 为什么MHA不能一起算?

MLA: ([heads, 1, dim] * [1, seqlen, dim]).sum(2)

MHA: ([heads, 1, dim] * [heads, seqlen, dim]).sum(2)

MLA解码：计算PV

- $o_{t,i} = p_{t,i} V$ seqlen*dim

  1*dim 1*seqlen

- $= p_{t,i} c^{KV} \left(W_i^{UV}\right)^T$ 不分head

  seqlen*512 512*dim

结合律：先算V=c*W还是先算r=p*c??

令 $R_{t,i} = \left(p_t c^{KV}\right)_i$ seqlen*512

#heads*seqlen

- $o_{t,i} = R_{t,i} \left(W_i^{UV}\right)^T$ 512*128

  #heads*512

同样是矩阵乘法!

总结：MLA

$$l_t^Q = \left(c_t^Q\right) \left(W_i^{UQ}\right)^T W_i^{UK} \qquad R_{t,i} = \left(p_t c^{KV}\right)_i$$

$$S_t = \ell_t^Q \left(c^{KV}\right)^T \qquad o_{t,i} = R_{t,i}\left(W_i^{UV}\right)^T$$

- 计算l：7168*1536+1536*192+192*512
- 计算S：seqlen*heads*576
- 计算R：seqlen*heads*576

两个O(seqlen)的都是矩阵乘法

- 计算O：heads*dim*512
- 访存：读取$W_i^{UQ}, W_i^{UK}, W_i^{UV}$：2*(1536*128*128+512*128*128*2)=80MB
- 访存：两个seqlen*heads*512的GEMM，若tilesize是32,128,512则
- 读l,c、写R：128*512+seqlen*512+128*512

总结：MLA

- 访存：读取$W_i^{UQ}, W_i^{UK}, W_i^{UV}$：2*(1536*128*128+512*128*128*2)=80MB
- 访存：两个seqlen*heads*512的GEMM，若tilesize是32,128,512则
- 读l,c、写R：2*(128*512+seqlen*512+128*512)=8.25MB (@8192) 125.25MB(@120K)
- 计算量：1.25Gflops (@8192) 18.9Gflops (@128K)
- 计算访存比：13.5flops/Byte (@8192,bs=1) 87.8flops/byte (@128000,bs=1)
- （理论上）单卡4090读取一次Kvcache=61*125.25 / (1008*1024) = 7.5ms
- 相比暴力解码后算MHA：计算=4*seqlen*numheads*headdim=0.5Gflops(@8192)/8Gflops(@128K)
- 访存：读取整个KV=2*2*seqlen*numheads*headdim=512MB (@8192) / 8000MB(@120K)
- 计算访存比=1flop / byte

DeepSeek真厉害！

效果

反而比MHA更好

Attention Mechanism	KV Cache per Token (# Element)	Capability
Multi-Head Attention (MHA)	$2n_h d_h l$	Strong
Grouped-Query Attention (GQA)	$2n_g d_h l$	Moderate
Multi-Query Attention (MQA)	$2d_h l$	Weak
MLA (Ours)	$(d_c + d_h^R)l \approx \frac{9}{2} d_h l$	Stronger

为什么(128,128)的KV能压缩进512维latent?

- 压缩感知理论，有限等距性质（Restricted Isometry Property）

对于任意高维稀疏向量x，利用RIP矩阵Φ（如randn）投影到低维空间后保范数。

$$(1-\delta)\|x\|^2 \leq \|\Phi x\|^2 \leq (1+\delta)\|x\|^2$$

- 将128*128维空间中的128个128维超平面（每个head的QK）嵌入到一个512维的空间应该是简单的……
- 每个head的QK是一个16384维向量，但只有128维非零
- 区别：RIP是保距离，我们需要的是保内积

为什么(128,128)的KV能压缩进512维latent?

- 根据RIP (Restricted Isometry Property) 理论
- 为了保住 s-稀疏向量的内积，所需的测量维度 M (即 Latent Dim) 下界大约是:

$$M \approx C \cdot s \cdot \log\left(\frac{D}{s}\right)$$

- s（稀疏度）=128, D（总维度）=16384
- $M \approx 896C$

- 核心：M不需要是numheads*headdim，只需要是headdim * log(numheads)!
- 实际：投影矩阵非随机，而是学习的，可能比该结果更优

谢谢

清华大学计算机系 陈键飞

《大模型计算》课程团队