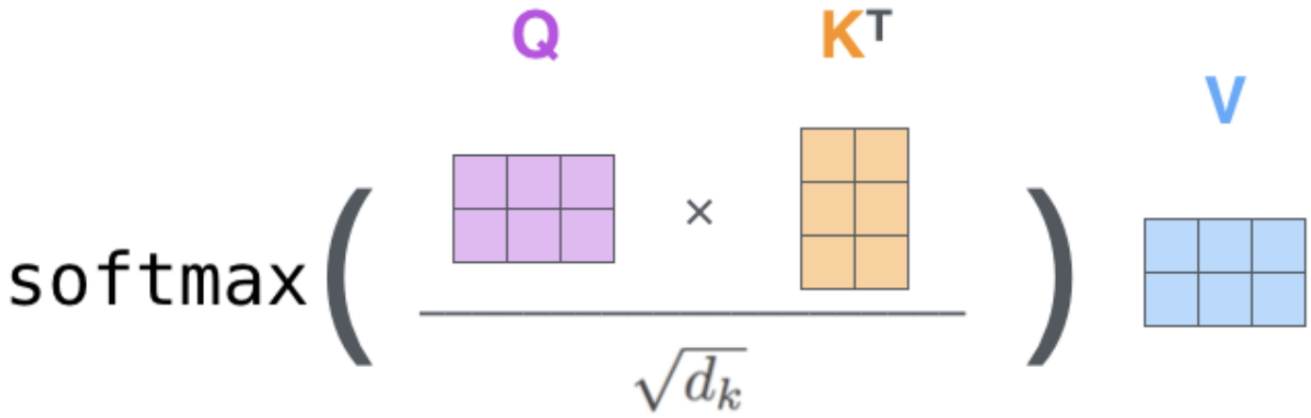
FlashAttention

Attention??



???Softmax?????

```
def softmax(x):
    x_max = x.max()
    x_exp = torch.exp(x - x_max)
    x_exp_sum = x_exp.sum()
    return x_exp / x_exp_sum
```

1. 记录每个 sub block的 softmax结果 + x_max(标量) + x_exp_sum(标量)
2. 更新全局的 max(标量) 和 exp_sum(标量)
3. 通过一次遍历 elementwise计算，就可以修正局部 softmax成全局 softmax
    1. sum和 max的 分块计算 避免了重复的数据读取进行统计
    2. exp指数的 加减法 操作可以通过 exp指数 乘除法 逆操作
    3. sum的结果，可以通过乘除法修正分块的错误偏置

其中，步骤 1可以在计算 qk时候顺便计算，步骤 3可以在计算 v时候顺便计算，所以 softmax结合 qkv 计算不浪费存储器的读写

原始 softmax需要遍历 3遍数据， 1. 统计 max， 2.统计 sum， 3，除法

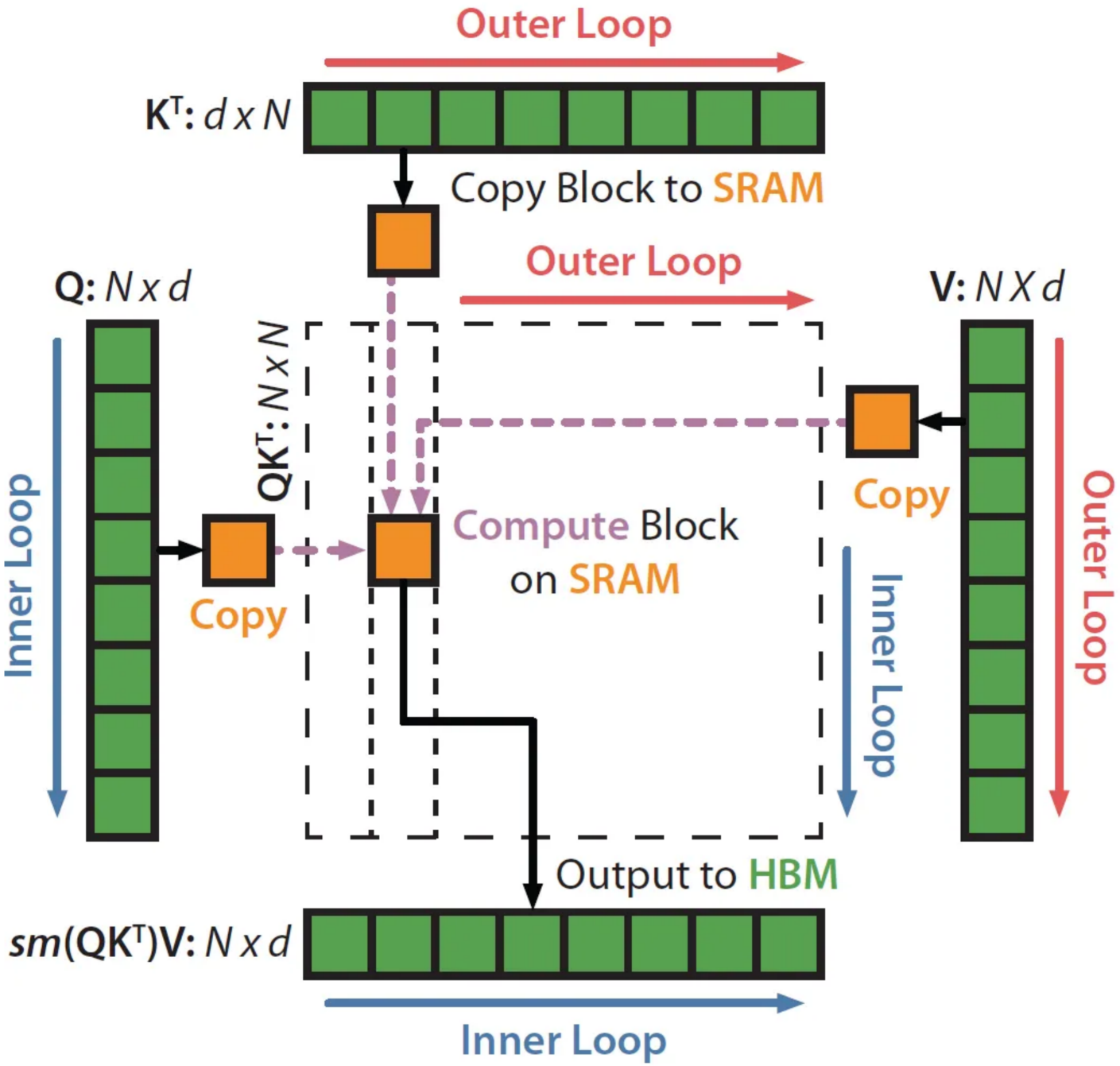
Flash Attention????

Algorithm 2 FLASHATTENTION Forward Pass

Require: Matrices $\mathbf{Q}, \mathbf{K}, \mathbf{V} \in \mathbb{R}^{N \times d}$ in HBM, on-chip SRAM of size M, softmax scaling constant $\tau \in \mathbb{R}$, masking function MASK, dropout probability p_{drop}.

1: Initialize the pseudo-random number generator state $\mathcal{R}$ and save to HBM.
2: Set block sizes $B_c = \lceil \frac{M}{4d} \rceil$, $B_r = \min\left(\lceil \frac{M}{4d} \rceil, d\right)$.
3: Initialize $\mathbf{O} = (0)_{N \times d} \in \mathbb{R}^{N \times d}, \ell = (0)_N \in \mathbb{R}^N, m = (-\infty)_N \in \mathbb{R}^N$ in HBM.
4: Divide $\mathbf{Q}$ into $T_r = \left\lceil \frac{N}{B_r} \right\rceil$ blocks $\mathbf{Q}_1, \ldots, \mathbf{Q}_{T_r}$ 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 $\mathbf{K}_1, \ldots, \mathbf{K}_{T_c}$ and $\mathbf{V}_1, \ldots, \mathbf{V}_{T_c}$, of size $B_c \times d$ each.
5: Divide $\mathbf{O}$ into T_r blocks $\mathbf{O}_i, \ldots, \mathbf{O}_{T_r}$ of size $B_r \times d$ each, divide ℓ into T_r blocks $\ell_i, \ldots, \ell_{T_r}$ of size B_r each, divide m into T_r blocks $m_1, \ldots, m_{T_r}$ of size B_r each.
6: **for** $1 \le j \le T_c$ **do**
7: Load $\mathbf{K}_j, \mathbf{V}_j$ from HBM to on-chip SRAM.
8: **for** $1 \le i \le T_r$ **do**
9: Load $\mathbf{Q}_i, \mathbf{O}_i, \ell_i, m_i$ from HBM to on-chip SRAM.
10: On chip, compute $\mathbf{S}_{ij} = \tau \mathbf{Q}_i \mathbf{K}_j^T \in \mathbb{R}^{B_r \times B_c}$.
11: On chip, compute $\mathbf{S}_{ij}^{\text{masked}} = \text{MASK}(\mathbf{S}_{ij})$.
12: On chip, compute $\tilde{m}_{ij} = \text{rowmax}(\mathbf{S}_{ij}^{\text{masked}}) \in \mathbb{R}^{B_r}$, $\tilde{\mathbf{P}}_{ij} = \exp(\mathbf{S}_{ij}^{\text{masked}} - \tilde{m}_{ij}) \in \mathbb{R}^{B_r \times B_c}$ (pointwise), $\tilde{\ell}_{ij} = \text{rowsum}(\tilde{\mathbf{P}}_{ij}) \in \mathbb{R}^{B_r}$.
13: On chip, compute $m_i^{\text{new}} = \max(m_i, \tilde{m}_{ij}) \in \mathbb{R}^{B_r}$, $\ell_i^{\text{new}} = e^{m_i - m_i^{\text{new}}} \ell_i + e^{\tilde{m}_{ij} - m_i^{\text{new}}} \tilde{\ell}_{ij} \in \mathbb{R}^{B_r}$.
14: On chip, compute $\tilde{\mathbf{P}}_{ij}^{\text{dropped}} = \text{dropout}(\tilde{\mathbf{P}}_{ij}, p_{\text{drop}})$.
15: Write $\mathbf{O}_i \leftarrow \text{diag}(\ell_i^{\text{new}})^{-1}(\text{diag}(\ell_i) e^{m_i - m_i^{\text{new}}} \mathbf{O}_i + e^{\tilde{m}_{ij} - m_i^{\text{new}}} \tilde{\mathbf{P}}_{ij}^{\text{dropped}} \mathbf{V}_j)$ to HBM.
16: Write $\ell_i \leftarrow \ell_i^{\text{new}}$, $m_i \leftarrow m_i^{\text{new}}$ to HBM.
17: **end for**
18: **end for**
19: Return $\mathbf{O}, \ell, m, \mathcal{R}$.

- 1-5: 主要在初始化和进行切分:
- 6-7: 遍历 K， V的每一块（ Outer Loop）
- 8: 遍历 Q的每一块 (Inner Loop)
- 9: 将分块后的 QKV的小块加载到 SRAM (Copy Block to SRAM)
- 10: 计算 Sij (Compute Block on SRAM)
- 11: 计算 Sij mask (Compute Block on SRAM)
- 12: 计算当前块的 m,l统计量 (Compute Block on SRAM)
- 13: 更新全局 m,l统计量 (Compute Block on SRAM)
- 14: dropout (Compute Block on SRAM)
- 15: 计算 Oi并写入 HBM (Output to HBM)
- 16: 把 li,mi写入 HBM (Output to HBM)



FlashAttention3

1. 使用 Hopper的异步 wgmma指令来重叠 cuda cores和 tensor cores的操作，充分利用 1D和 2D算力

Revision #8
Created 2025-03-08 10:18:23 UTC by Colin
Updated 2026-04-29 07:34:24 UTC by Colin