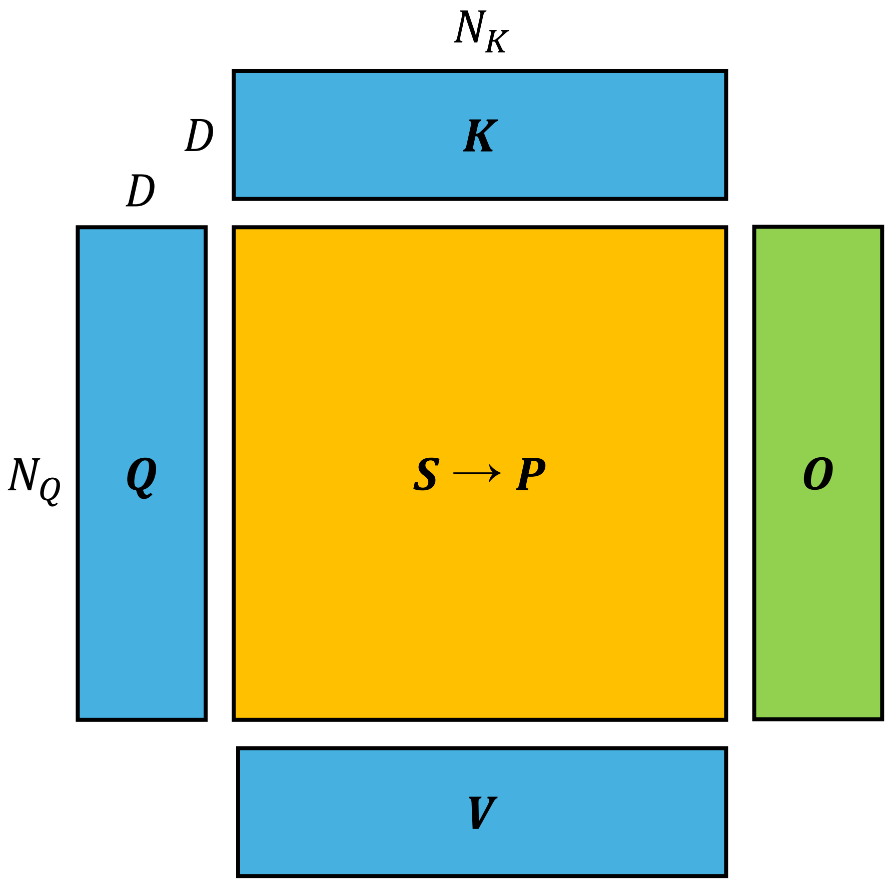
MSRA SH Triton Study Group
5. Flash-Attention (Algorithm)

2025/08/15

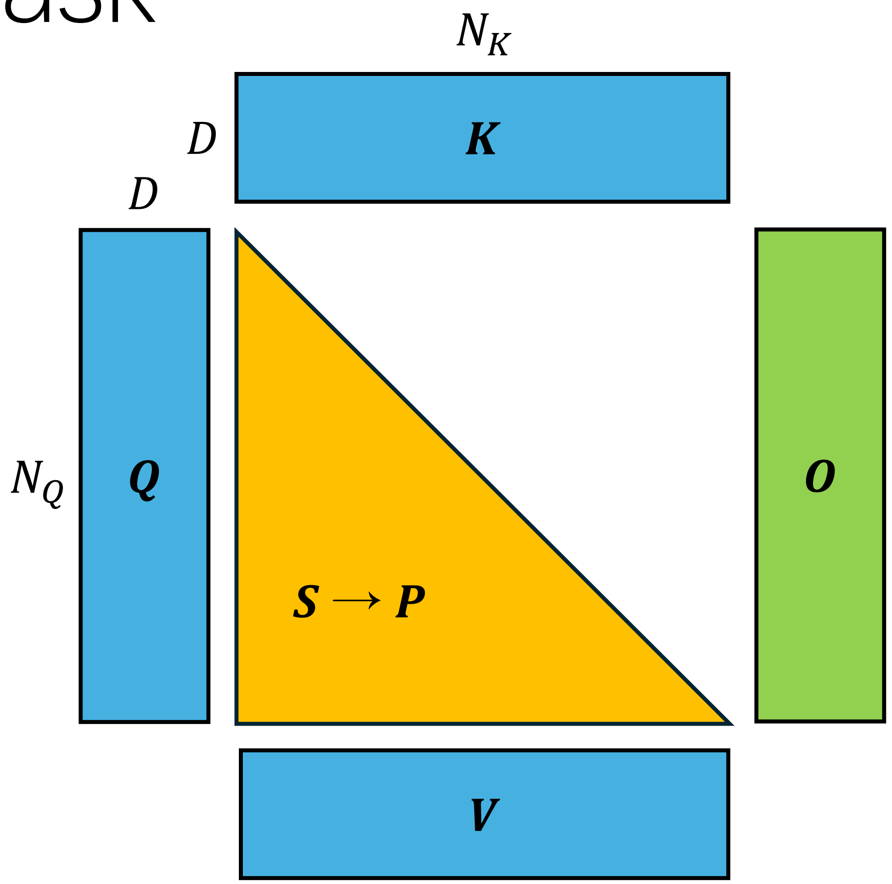
Attention

- $\boldsymbol{Q} \in \mathbb{R}^{N_Q \times D}$; $\boldsymbol{K}, \boldsymbol{V} \in \mathbb{R}^{N_K \times D}$
- $\boldsymbol{S} = \frac{\boldsymbol{Q}\boldsymbol{K}^{\mathrm{T}}}{\sqrt{D}} \in \mathbb{R}^{N_Q \times N_K}$
- $\boldsymbol{P}_{ij} = \mathrm{softmax}\left(\boldsymbol{S}_{ij}\right) = \frac{\exp\left(\boldsymbol{S}_{ij} - \max_j(\boldsymbol{S}_{ij})\right)}{\sum_j \exp\left(\boldsymbol{S}_{ij} - \max_j(\boldsymbol{S}_{ij})\right)}$
- $\boldsymbol{O} = \boldsymbol{P}\boldsymbol{V} \in \mathbb{R}^{N_Q \times D}$

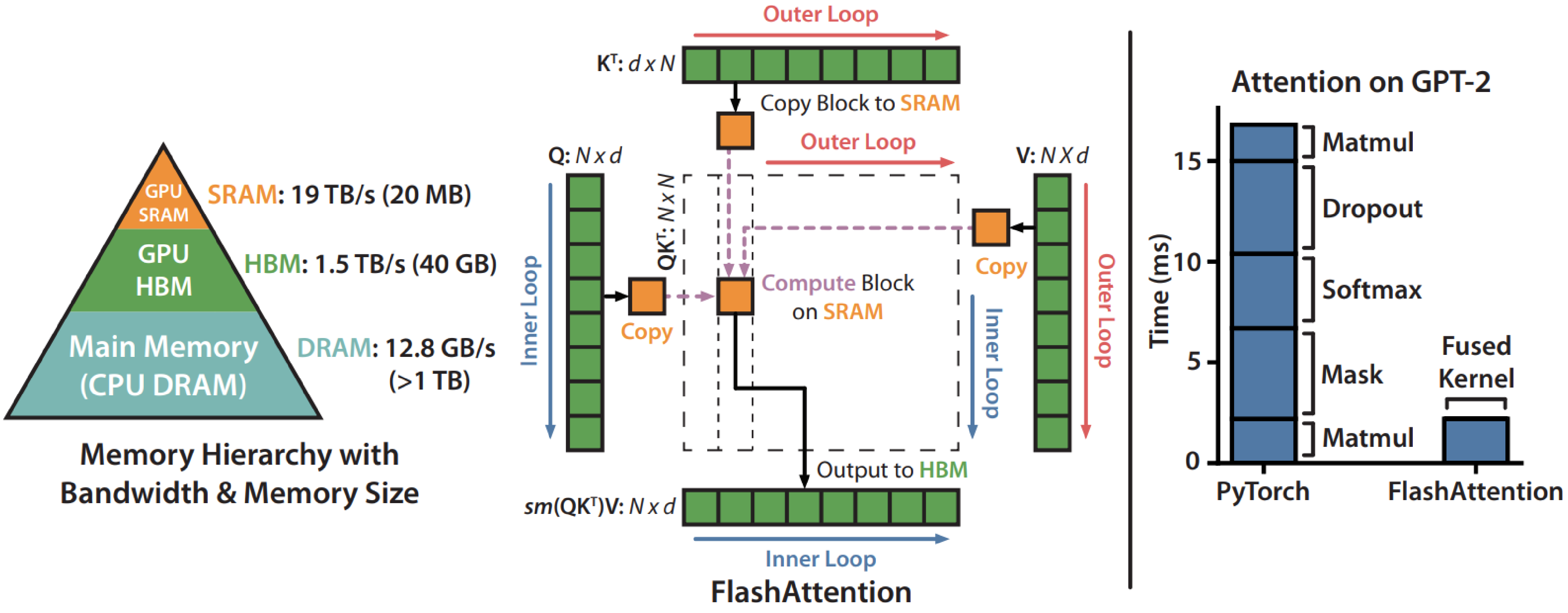


Self-Attention with Causal Mask

- Definition:
  - $N_Q = N_K$
  - $\boldsymbol{P}_{ij} \leftarrow \begin{cases} \boldsymbol{P}_{ij}, i \leq j \\ -\infty, i > j \end{cases}$
- Challenges:
  - The big $N_Q \times N_K$ tensor
  - $\mathrm{softmax}()$ is inefficient
  - Redundant calculation on masked $\boldsymbol{P}_{ij}$



Flash Attention



Fused Attention: GeMM-Lever Efficiency

- Assume 100% L2 cache hit ratio

	Total HBM Access	Total Calculation	Arithmetic Intensity
$\boldsymbol{S} = \boldsymbol{QK}^{\mathrm{T}}/\sqrt{D}$	$N_Q D + N_K D \ +N_Q N_K$	$N_Q N_K D$	Memory-bound
$\boldsymbol{P}_{ij} = \mathrm{softmax}(\boldsymbol{S}_{ij})$	$2N_Q N_K$	$CN_Q N_K$	Memory-bound
$\boldsymbol{O} = \boldsymbol{PV}$	$N_Q N_K + \ N_K D + N_Q D$	$N_Q N_K D$	Memory−bound
Fused attn()	$2N_Q D + 2N_K D$	$(2D + C)N_Q N_K$	Balanced

- $-N_Q N_K$ GPU HBM cost, $-4N_Q N_K$ GPU HBM access

Fused Attention: GeMM-Lever Efficiency

- Assume no L2 cache; $T_Q = T_K = D$; "best intensity" $E \coloneqq \frac{T_Q T_K}{T_Q + T_K}$

	Total HBM Access	Total Calculation	Arithmetic Intensity	Latency
$\boldsymbol{S} = \frac{\boldsymbol{Q}\boldsymbol{K}^{\mathrm{T}}}{\sqrt{D}}$	$\frac{N_K}{T_K} N_Q D + \frac{N_Q}{T_Q} N_K D \; + N_Q N_K$	$N_Q N_K D$	$\frac{T_Q T_K}{T_Q + T_K + \frac{T_Q T_K}{D}} < E$	$1 := \mathrm{mem}(3 N_Q N_K)$
$\boldsymbol{P}_{ij} = \mathrm{softmax}(\boldsymbol{S}_{ij})$	$2 N_Q N_K$	$C N_Q N_K$	$\frac{C}{2} \ll E$	$\frac{2}{3}$
$\boldsymbol{O} = \boldsymbol{P}\boldsymbol{V}$	$N_Q N_K + \frac{N_Q}{T_Q} N_K D + N_Q D$	$N_Q N_K D$	$\frac{T_Q T_K}{\frac{T_Q T_K}{D} + T_K + \frac{T_K}{N_K} T_Q} \approx E$	$\approx \frac{2}{3}$
Fused attn()	$2 \frac{N_K}{T_K} N_Q D + 2 \frac{N_Q}{T_Q} N_K D$	$(2D + C) N_Q N_K$	$\frac{2D + C}{2D} \frac{T_Q T_K}{T_Q + T_K} \approx E$	$\frac{4}{3}$

Online Softmax

- $$\begin{cases} \boldsymbol{P}_{ij}^{0} = \dfrac{\exp\left(\boldsymbol{S}_{ij}^{0} - \max\limits_{0\le l<T}\left(\boldsymbol{S}_{il}^{0}\right)\right)}{\sum_{l=0}^{T}\exp\left(\boldsymbol{S}_{ij}^{0} - \max\limits_{0\le l<T}\left(\boldsymbol{S}_{il}^{0}\right)\right)} = \dfrac{\exp\left(\boldsymbol{S}_{ij}^{0} - M_i^0\right)}{L_i^0} \\ \boldsymbol{P}_{ij}^{1} = \dfrac{\exp\left(\boldsymbol{S}_{ij}^{1} - \max\limits_{T\le l<2T}\left(\boldsymbol{S}_{il}^{1}\right)\right)}{\sum_{l=T}^{2T}\exp\left(\boldsymbol{S}_{ij}^{1} - \max\limits_{T\le l<2T}\left(\boldsymbol{S}_{il}^{1}\right)\right)} = \dfrac{\exp\left(\boldsymbol{S}_{ij}^{1} - M_i^1\right)}{L_i^1} \end{cases}$$
- $M_i^{[0,1]} = \max\left(M_i^0, M_i^1\right)$
- $L_i^{[0,1]} = \exp\left(M_i^0 - M_i^{[0,1]}\right)L_i^0 + \exp\left(M_i^1 - M_i^{[0,1]}\right)L_i^1$
- $$\boldsymbol{P}_{ij}^{[0,1]} = \dfrac{\exp\left(\boldsymbol{S}_{ij}^{[0,1]} - \max\limits_{0\le l<2T}\left(\boldsymbol{S}_{il}^{[0,1]}\right)\right)}{\sum_{l=0}^{2T}\exp\left(\boldsymbol{S}_{ij}^{[0,1]} - \max\limits_{0\le l<2T}\left(\boldsymbol{S}_{il}^{[0,1]}\right)\right)} = \dfrac{\exp\left(\boldsymbol{S}_{ij}^{[0,1]} - M_i^{[0,1]}\right)}{L_i^{[0,1]}}$$

- For $1 \le j \le \frac{N}{T}$
  - Load $\boldsymbol{S}_{ij}$
  - $M_i^{\text{local}} \leftarrow \max\limits_{-1}\left(\boldsymbol{S}_{ij}\right)$
  - $L_i^{\text{local}} \leftarrow \text{sum}_{-1}\left(\exp\left(\boldsymbol{S}_{ij} - M_i^{\text{local}}\right)\right)$
  - $\boldsymbol{P}_{ij}^{\text{local}} \leftarrow \dfrac{\exp\left(\boldsymbol{S}_{ij} - M_i^{\text{local}}\right)}{L_i^{\text{local}}}$
  - Update (M_i, L_i)
- For $1 \le j \le \frac{N}{T}$
  - $$\boldsymbol{P}_{ij} \leftarrow \dfrac{\exp\left(\boldsymbol{S}_{ij} - M_i\right)}{L_i} = \exp\left(M_i^{\text{local}} - M_i\right)\dfrac{L_i^{\text{local}}}{L_i}\boldsymbol{P}_{ij}^{\text{local}}$$

How to Fuse: Forward

- $\begin{cases} \boldsymbol{S}^0_{ij} = D^{-1/2} \sum_{k=0}^{D} \boldsymbol{Q}_{ik} \boldsymbol{K}^0_{jk} \\ \boldsymbol{S}^1_{ij} = D^{-1/2} \sum_{k=0}^{D} \boldsymbol{Q}_{ik} \boldsymbol{K}^1_{jk} \end{cases}$
- $\begin{cases} \boldsymbol{P}^0_{ij} = \frac{\exp\left(\boldsymbol{S}^0_{ij} - \max_{0 \le l < T}(\boldsymbol{S}^0_{il})\right)}{\sum_{l=0}^{T} \exp\left(\boldsymbol{S}^0_{ij} - \max_{0 \le l < T}(\boldsymbol{S}^0_{il})\right)} = \frac{\exp\left(\boldsymbol{S}^0_{ij} - M^0_i\right)}{L^0_i} \\ \boldsymbol{P}^1_{ij} = \frac{\exp\left(\boldsymbol{S}^1_{ij} - \max_{T \le l < 2T}(\boldsymbol{S}^1_{il})\right)}{\sum_{l=T}^{2T} \exp\left(\boldsymbol{S}^1_{ij} - \max_{T \le l < 2T}(\boldsymbol{S}^1_{il})\right)} = \frac{\exp\left(\boldsymbol{S}^1_{ij} - M^1_i\right)}{L^1_i} \end{cases}$
- $M_i^{[0,1]} = \max\left(M_i^0, M_i^1\right)$
- $L_i^{[0,1]} = \exp\left(M_i^0 - M_i^{[0,1]}\right) L_i^0 + \exp\left(M_i^1 - M_i^{[0,1]}\right) L_i^1$
- $\boldsymbol{P}^{[0,1]}_{ij} = \frac{\exp\left(\boldsymbol{S}^{[0,1]}_{ij} - \max_{0 \le l < 2T}\left(\boldsymbol{S}^{[0,1]}_{il}\right)\right)}{\sum_{l=0}^{2T} \exp\left(\boldsymbol{S}^{[0,1]}_{ij} - \max_{0 \le l < 2T}\left(\boldsymbol{S}^{[0,1]}_{il}\right)\right)} = \frac{\exp\left(\boldsymbol{S}^{[0,1]}_{ij} - M_i^{[0,1]}\right)}{L_i^{[0,1]}}$

- $\begin{cases} \boldsymbol{O}^0_{ik} = \sum_{j=0}^{T} \boldsymbol{P}^0_{ij} \boldsymbol{V}^0_{jk} \\ \boldsymbol{O}^1_{ik} = \sum_{j=T}^{2T} \boldsymbol{P}^1_{ij} \boldsymbol{V}^1_{jk} \end{cases}$
- $\boldsymbol{O}^{[0,1]}_{ik} = \sum_{j=0}^{2T} \boldsymbol{P}^{[0,1]}_{ij} \boldsymbol{V}^{[0,1]}_{jk} = \sum_{j=0}^{T} \boldsymbol{P}^{[0,1]}_{ij} \boldsymbol{V}^0_{jk} + \sum_{j=T}^{2T} \boldsymbol{P}^{[0,1]}_{ij} \boldsymbol{V}^1_{jk}$
- $= \sum_{j=0}^{T} \frac{\exp\left(\boldsymbol{S}^0_{ij} - M_i^{[0,1]}\right)}{L_i^{[0,1]}} \boldsymbol{V}^0_{jk} + \sum_{j=T}^{2T} \frac{\exp\left(\boldsymbol{S}^1_{ij} - M_i^{[0,1]}\right)}{L_i^{[0,1]}} \boldsymbol{V}^1_{jk}$
- $= \frac{\exp\left(M_i^{[0,1]}\right)}{\exp\left(M_i^0\right)} \frac{L_i^0}{L_i^{[0,1]}} \sum_{n=0}^{T} \frac{\exp\left(\boldsymbol{S}^0_{ij} - M^0_i\right)}{L^0_i} \boldsymbol{V}^0_{jk} +$
  $\frac{\exp\left(M_i^{[0,1]}\right)}{\exp\left(M_i^1\right)} \frac{L_i^1}{L_i^{[0,1]}} \sum_{n=T}^{2T} \frac{\exp\left(\boldsymbol{S}^1_{ij} - M^1_i\right)}{L^1_i} \boldsymbol{V}^1_{jk}$
- $= \frac{\exp\left(M_i^{[0,1]}\right)}{\exp\left(M_i^0\right)} \frac{L_i^0}{L_i^{[0,1]}} \boldsymbol{O}^0_{ik} + \frac{\exp\left(M_i^{[0,1]}\right)}{\exp\left(M_i^1\right)} \frac{L_i^1}{L_i^{[0,1]}} \boldsymbol{O}^1_{ik}$
- $\coloneqq \alpha^0 \beta^0 \boldsymbol{O}^0_{ik} + \alpha^1 \beta^1 \boldsymbol{O}^1_{ik}$

How to Fuse: Forward

- Tile $\boldsymbol{Q}, \boldsymbol{O}$ into $\frac{N_Q}{T_Q}$ blocks $\boldsymbol{Q}_i, \boldsymbol{O}_i$; Tile $\boldsymbol{K}, \boldsymbol{V}$ into $\frac{N_K}{T_K}$ blocks $\boldsymbol{K}_j, \boldsymbol{V}_j$
- Parallel for $1 \leq i \leq \frac{N_Q}{T_Q}$
  - in Registers: $\boldsymbol{Q}_i \leftarrow (0)^{T_Q \times D}_{\mathrm{float32}}, M_i \leftarrow (-\infty)^{T_Q}_{\mathrm{float32}}, L_i \leftarrow (0)^{T_Q}_{\mathrm{float32}}$
  - For $1 \leq j \leq \frac{N_K}{T_K}$
    - $\boldsymbol{S}_{ij} \leftarrow \frac{\boldsymbol{Q}_i \boldsymbol{K}_j^{\mathrm{T}}}{\sqrt{D}}, M_i^{\mathrm{local}} \leftarrow \max\limits_{-1}\left(\boldsymbol{S}_{ij}\right), L_i^{\mathrm{local}} \leftarrow \mathrm{sum}_{-1}\left(\exp\left(\boldsymbol{S}_{ij} - M_i^{\mathrm{local}}\right)\right)$
    - Update (M_i, L_i) and calc (α, β) given $\left(M_i, M_i^{\mathrm{local}}, L_i, L_i^{\mathrm{local}}\right)$
    - $\boldsymbol{P}_{ij} \leftarrow \frac{\exp\left(\boldsymbol{S}_{ij} - M_i\right)}{L_i}, \boldsymbol{O}_i \leftarrow \alpha\beta \boldsymbol{O}_i + \boldsymbol{P}_{ij} \boldsymbol{V}_j$
  - Save $\boldsymbol{O}_i$ to GPU HBM

How to Fuse: Backward

- $\frac{\partial \phi}{\partial \boldsymbol{V}} = \boldsymbol{P}^{\mathrm{T}} \cdot \frac{\partial \phi}{\partial \boldsymbol{O}}, \ \frac{\partial \phi}{\partial \boldsymbol{P}} = \frac{\partial \phi}{\partial \boldsymbol{O}} \cdot \boldsymbol{V}^{\mathrm{T}}$
- $\frac{\partial \phi}{\partial \boldsymbol{S}_i} = \frac{\partial \phi}{\partial \boldsymbol{P}_i}\left(\mathrm{diag}(\boldsymbol{P}_i) - \boldsymbol{P}_i^{\mathrm{T}} \boldsymbol{P}_i\right) = \frac{\partial \phi}{\partial \boldsymbol{P}_i} \circ \boldsymbol{P}_i - \left(\frac{\partial \phi}{\partial \boldsymbol{P}_i} \cdot \boldsymbol{P}_i^{\mathrm{T}}\right) \boldsymbol{P}_i$
- $\frac{\partial \phi}{\partial \boldsymbol{P}_i} \cdot \boldsymbol{P}_i^{\mathrm{T}} = \left(\frac{\partial \phi}{\partial \boldsymbol{O}_i} \cdot \boldsymbol{V}^{\mathrm{T}}\right) \cdot \boldsymbol{P}_i^{\mathrm{T}} = \frac{\partial \phi}{\partial \boldsymbol{O}_i} \cdot \boldsymbol{O}_i^{\mathrm{T}} = \sum_{k=0}^{D} \frac{\partial \phi}{\partial \boldsymbol{O}_{ik}} \boldsymbol{O}_{ik} := \Delta_i$
- $\frac{\partial \phi}{\partial \boldsymbol{S}} = \left(\frac{\partial \phi}{\partial \boldsymbol{P}} - \Delta\right) \circ \boldsymbol{P}$
- $\frac{\partial \phi}{\partial \boldsymbol{Q}} = \frac{\partial \phi}{\partial \boldsymbol{S}} \cdot \boldsymbol{K}, \frac{\partial \phi}{\partial \boldsymbol{K}} = \frac{\partial \phi}{\partial \boldsymbol{S}^{\mathrm{T}}} \cdot \boldsymbol{Q}$

Flash Attention: Tips

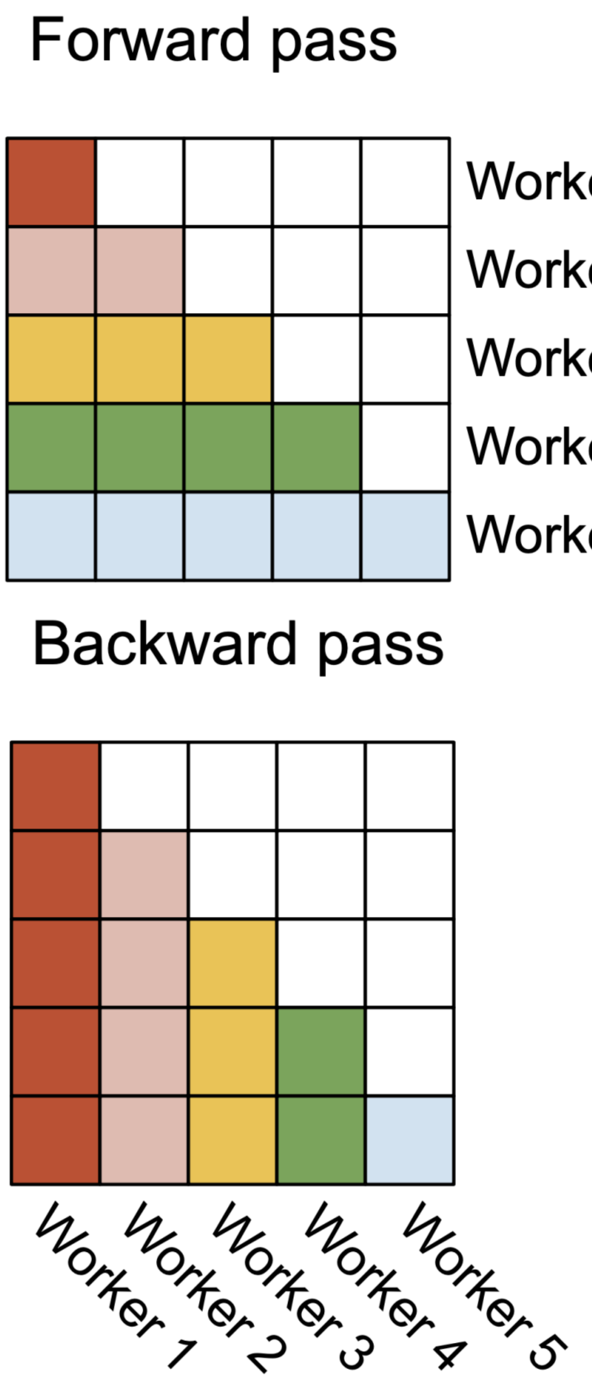
- Based on small head size (to store $\boldsymbol{Q}^i, \boldsymbol{K}^j, \boldsymbol{V}^j, \boldsymbol{O}^i$ in SRAM)
- The two for-loops are swapable
- Training: save M and L for backward
- Backward is much slower (recompute & limited by SRAM)
- Skip masked blocks

Flash Attention Implementations

- Official: Dao-AILab/flash-attention
- Easy to modify: openai/triton
- For C++ template lovers: nvidia/cutlass
- Compatibility: facebookresearch/xformers

Flash Attention 2

- Forward:
  - $\boldsymbol{O}_i \leftarrow \alpha \boldsymbol{O}_i + \exp(\boldsymbol{S}_{ij} - M_i)\boldsymbol{V}_j; \boldsymbol{O}_i \leftarrow \frac{\boldsymbol{O}_i}{L_i}$
  - Save $L' = M + \log(L)$ for recompute
- Backward:
  - $\boldsymbol{P} = \frac{\exp(\boldsymbol{S}-M)}{L} = \exp(\boldsymbol{S} - L')$
- Long Context Generation:
  - Partition on K/V caches
  - Another kernel to combine results
- Parallelism Workload Balance



Reading Materials

- Flash Attention: [2205.14135] FlashAttention: Fast and Memory-Efficient Exact Attention with IO-Awareness
- Flash Attention 2: [2307.08691] FlashAttention-2: Faster Attention with Better Parallelism and Work Partitioning
- Flash Decoding: Flash-Decoding for long-context inference – PyTorch
- Flash Attention 3: [2407.08608] FlashAttention-3: Fast and Accurate Attention with Asynchrony and Low-precision