FlashAttention

A Preprint

GPU has 2 types of memory:

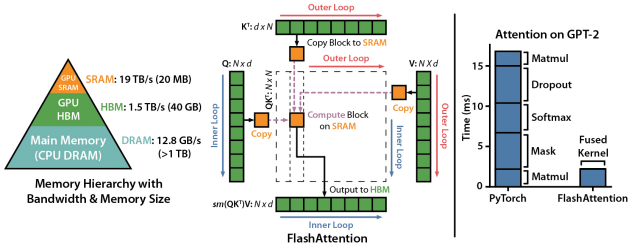
$$\text{Slow, but big HBM memory where the main data is stored} \tag{1}$$

$$\text{Fast, but small SRAM memory which is located directly on computational kernels} \tag{2}$$

The main idea is to avoid writing attention on HBM and both taking it from HBM. Authors restructure the attention computation to split the input into blocks and make several passes over input blocks, thus incrementally performing the softmax reduction (also known as tiling). Authors also store softmax normalization factor from the forward pass on-chip to use it in backward pass.

Compute-bound: the time taken by the operation is determined by how many arithmetic operations there are, while time accessing HBM is much smaller. Typical examples are matrix multiply with large inner dimension, and convolution with large number of channels.

2. Memory-bound: the time taken by the operation is determined by the number of memory accesses, while time spent in computation is much smaller. Examples include most other operations: elementwise (e.g., activation, dropout), and reduction (e.g., sum, softmax, batch norm, layer norm).



Tiling is done next way:

$$m(x) = \max x_i, \quad f(x) = [e^{x_1 - m(x)} \dots e^{x_i - m(x)}], \quad l(x) = \sum f(x), \quad softmax(x) = \frac{f(x)}{l(x)}$$

Algorithm 1 FlashAttention

Require: Matrices $Q, K, V \in \mathbb{R}^{N \times d}$ in HBM (High Bandwidth Memory), on-chip SRAM of size M.

Set block sizes $B_c = \left\lfloor \frac{M}{4d} \right\rfloor$, $B_r = \min\left(\left\lfloor \frac{M}{4d} \right\rfloor, d\right)$

Initialize $O = \mathbf{0}_{N \times d} \in \mathbb{R}^{N \times d}$, $\ell = \mathbf{0}_N \in \mathbb{R}^N$, $m = -\infty_N \in \mathbb{R}^N$ in HBM

Divide Q into $T_r = \left\lceil \frac{N}{B_r} \right\rceil$ blocks $Q_1, \ldots, Q_{T_r}$ of size $B_r \times d$

Divide K, V into $T_c = \left\lceil \frac{N}{B_c} \right\rceil$ blocks $K_1, \ldots, K_{T_c}$ and $V_1, \ldots, V_{T_c}$ of size $B_c \times d$

Divide O into T_r blocks O_i of size $B_r \times d$, ℓ into T_r blocks ℓ_i of size B_r, m into T_r blocks m_i of size B_r

for $j = 1$ to T_c **do**

  Load K_j, V_j from HBM to on-chip SRAM

  for $i = 1$ to T_r **do**

    Load Q_i, O_i, ℓ_i, m_i from HBM to SRAM

    On chip, compute $S_{ij} = Q_i K_j^\top \in \mathbb{R}^{B_r \times B_c}$

    Compute:

$$\tilde{m}_{ij} = \text{rowmax}(S_{ij}) \in \mathbb{R}^{B_r}$$
$$\tilde{P}_{ij} = \exp(S_{ij} - \tilde{m}_{ij}) \in \mathbb{R}^{B_r \times B_c}$$
$$\tilde{\ell}_{ij} = \text{rowsum}(\tilde{P}_{ij}) \in \mathbb{R}^{B_r}$$

    Compute:

$$m_i^{\text{new}} = \max(m_i, \tilde{m}_{ij}) \in \mathbb{R}^{B_r}$$
$$\ell_i^{\text{new}} = e^{m_i - m_i^{\text{new}}} \cdot \ell_i + e^{\tilde{m}_{ij} - m_i^{\text{new}}} \cdot \tilde{\ell}_{ij} \in \mathbb{R}^{B_r}$$

    Update output:

$$O_i \leftarrow \text{diag}(\ell_i^{\text{new}})^{-1} \left(\text{diag}(\ell_i) e^{m_i - m_i^{\text{new}}} O_i + e^{\tilde{m}_{ij} - m_i^{\text{new}}} \tilde{P}_{ij} V_j \right)$$

    Write O_i, $\ell_i \leftarrow \ell_i^{\text{new}}$, $m_i \leftarrow m_i^{\text{new}}$ back to HBM

  end for

end for

return O