Flash Attention v2 原理

FlashAttention-2 相比于 FlashAttention 主要改进了一下几个方面:

Scale Once

在 flash attention v1 中, 注意到每一次更新 o_i' 都需要除去一个 l_i', 这会极大的浪费 GPU 中非矩阵乘法运算单元, 于是出现了能不能尽可能减少做这个 scaling 的次数.

注意到在 flash attention v1 的更新公式中,

$$o_i' = o_{i-1}' \frac{(e^{m_{i-1}-m_i})l_{i-1}'}{l_i'} + \frac{e^{x_i-m_i}}{l_i'}V[i,:],$$

则

$$\begin{aligned}
o_1' &= \frac{e^{x_1-m_1}}{l_1'}V[1,:] \\
o_2' &= o_1' \frac{(e^{m_1-m_2})l_1'}{l_2'} + \frac{e^{x_2-m_2}}{l_2'}V[2,:].
\end{aligned}$$

考虑

$$o_1'' = e^{x_1-m_1}V[1,:] = o_1'l_1',$$

则

$$\begin{aligned}
o_2'' &= o_2'l_2' \\
&= o_1'l_1'e^{m_1-m_2} + e^{x_2-m_2}V[2,:] \\
&= o_1''e^{m_1-m_2} + e^{x_2-m_2}V[2,:],
\end{aligned}$$

以此类推, 得到新的迭代公式

$$o_i'' = o_{i-1}''e^{m_{i-1}-m_i} + e^{x_i-m_i}V[i,:],$$

且对任意的 o_i', o_i'', 都有

$$o_i' = \frac{o_i''}{l_i'}.$$

也就是说, 在更新 o 的时候, 不需要每次都做 scale, 只需在最后一次做 scale 即可.

Parallelism

flash attention v1 在 `batch_size` 和 `num_head` 维度上做并行化, 每一个 block 处理一个头, 但是, 当处理长序列时, 由于内存限制, 通常会减少 `batch_size` 和 `num_head` 数量, 导致并行化程度降低. 因此, flash attention v2 在 `seq_len` 维度上也做了并行化.

Mask

在 flash attention v1 中, 对在有掩码的位置(上三角)也进行了计算, 但其实这部分的计算可以省去.

Q, K, V 循环顺序变更

在 flash attention v1 中, 以 K, V 作为主循环, 导致需要多次重复从 HBM 中读取 Q, O, l, m 并写回, wraps 间需要相互通信来处理 Q, O 的值也不能在一次读取中得出, 造成极大的性能损失.

flash attention v2 将 Q, O, l, m 作为主循环, K, V 移到内循环处理, 这样使得 warps 之间不再需要相互通信去处理 Q. 同时, 频繁读取 K, V 的代价也小于频繁读取 Q, O, l, m 的代价, 在一次主循环内, O_i 时存储在 SRAM 上的, 代价远小于 HBM.