Flash Attention

Flash Attention 是一种用于加速 Transformer 模型中 Self-Attention 的方法。它利用底层硬件的内存层次知识，例如GPU的内存层次结构，来提高计算速度和减少内存访问开销。 Flash Attention 的核心原理是通过将输入分块并在每个块上执行注意力操作，从而减少对高带宽内存(HBM)的读写操作。具体而言，Flash Attention 使用平铺和重计算等经典技术，将输入块从 HBM 加载到 SRAM (快速缓存)，在 SRAM 上执行注意力操作，并将结果更新回 HBM。Flash Attention 减少了内存读写量，从而实现了2-4倍的时钟时间加速。

最新的 Flash Attention-2 版本进一步优化了 Flash Attention 算法，使用了更好的并行化和工作分区方法，使得计算速度提高了2倍。Flash Attention-2 还支持更高的头维数和多查询注意力等新特性，进一步提升了性能和灵活性。

传统 Attention

对于输入序列 $Q, K, V \in R^{N \times d}$ ，其中 N 是序列长度， d 是词嵌入维度，传统的 Self-Attention 计算可以表示为：

$$S = QK^T \in R^{N \times N}$$

$$P = \mathrm{softmax}(S) \in R^{N \times N}$$

$$O = PV \in R^{N \times d}$$

其时间和内存(HBM)复杂度为 $O(N^2)$ 。

Flash Attention

Flash Attention 的设计减少了 HBM 的访问，将 Q, K, V 切分为小块后放入 SRAM 中，提高了访问速度。它的核心技术包括：平铺(tiling)、重计算(recomputation)。

平铺

首先回顾传统的 Attention 计算中的 softmax 操作：

$$\mathrm{softmax}(x_i) = \frac{e^{x_i}}{\sum_{k=1}^{N} e^{x_k}}$$

Flash Attention 为了减少计算中的不稳定性，会将 x_i 减去最大值，再进行 softmax，即 safe softmax：

$$m(x) = \max(x_k)$$

$$f(x_i) = e^{x_i - m(x)}$$

$$l(x) = \sum_k^N f(x_k)$$

$$\text{softmax}(x_i) = \frac{f(x_i)}{l(x)}$$

对于分块数据，如 $x = [x^{(1)}, x^{(2)}]$ ，safe softmax 可以分解计算为：

$$m(x) = m([x^{(1)}, x^{(2)}]) = \max(m(x^{(1)}), m(x^{(2)})$$

$$f(x^{(1)}) = [f(x_i)] = [e^{x_i - m(x^{(1)})}], f(x^{(2)}) = [f(x_j)] = [e^{x_j - m(x^{(2)})}]$$

$$l(x^{(1)} = \sum f(x^{(1)}) = \sum_i f(x_i), l(x^{(2)} = \sum f(x^{(2)}) = \sum_j f(x_j)$$

$$f(x) = [e^{m(x^{(1)}) - m(x)} f(x^{(1)}), e^{m(x^{(2)}) - m(x)} f(x^{(2)})]$$

$$l(x) = e^{m(x^{(1)}) - m(x)} l(x^{(1)}) + e^{m(x^{(2)}) - m(x)} l(x^{(2)})$$

$$\text{softmax}(x) = \frac{f(x)}{l(x)}$$

重计算

重计算将数据引入SRAM中进行分块计算，减少了HBM的访问，提升了计算速度。其主要步骤有：

- 初始化：根据SRAM大小计算分别计算 Q 和 K, V 的数据块大小，如 Q 划分为 T_r 块，如 K, V 划分为 T_c 块；初始化 $O = (0)_{N \times d}, l = (0)_N, m = (-\infty)_N$ ，并将其划分为 T_r 块； Q 和 K 向量的点积缩放常数 $\tau = \sqrt{d}$ ；掩码函数 MASK ；dropout 比率 p。
- 循环计算：
  - for $1 <= j <= T_c$ (外循环)：
    - 将 K_j, V_j 从 HBM 加载到 SRAM 中；
    - for $1 <= i <= T_r$ (内循环)：
      - 将 Q_i, O_i, l_i, m_i 从 HBM 加载到 SRAM 中；
      - 计算 $S_{i,j} = \tau Q_i K_j^T$ ；
      - 计算 $S_{i,j}^{masked} = \text{MASK}(S_{i,j})$

    - 计算 $\tilde{m}_{ij} = \text{rowmax}(S_{i,j}^{masked}), \tilde{P}_{ij} = \exp(S_{i,j}^{masked} - \tilde{m}_{ij}), \tilde{l}_{ij} = \text{rowsum}(\tilde{P}_{ij})$；
    - 计算 $m_i^{new} = \max(m_i, \tilde{m}_{ij}), l_i^{new} = e^{m_i - m_i^{new}} l_i + e^{\tilde{m}_{ij} - m_i^{new}} \tilde{l}_{ij}$；
    - 计算 $\tilde{P}_{ij}^{dropped} = \text{dropout}(\tilde{P}_{ij}, p)$；
    - 将 $O_i \leftarrow \text{diag}(l_i^{new})^{-1}(\text{diag}(l_i)e^{m_i - m_i^{new}} O_i + e^{\tilde{m}_{ij} - m_i^{new}} \tilde{P}_{ij}^{dropped} V_j)$ 写到HBM；
    - 将 $m_i \leftarrow m_i^{new}$ 写到HBM；
    - 将 $l_i \leftarrow l_i^{new}$ 写到HBM。
- 返回 O, l, m。

Flash Attention-2

Flash Attention-2 是 Flash Attention 的改进版本，它引入了更多的优化技术，如更好的并行化和工作分区方法。其主要改进包括：

- 优化循环读写流程：Flash Attention-2 将 Q 作为外循环，将 K, V 作为内循环。这样在计算一个 O_i 的周期内，就不需要对 O_i 进行多次读写。
- 减少非乘法计算量：在计算 每个 O_i 时，每次循环先不考虑 softmax 的分母，而是在最后一步的时候进行处理，这样可以减少非乘法计算量。
- 忽略需要完全掩码的块的计算：如果一个块的所有元素都被掩码了，那么就不需要对这个块进行计算。