NT 模式下微内核的合并与重叠

1.概述

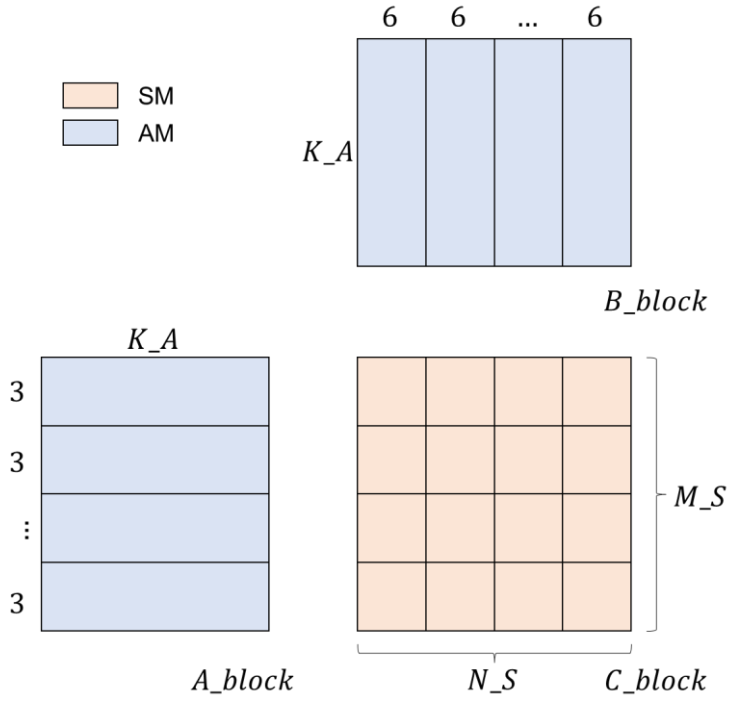
这个版本的微内核三个维度的长度皆可变，但要求 m 是 3 的整数倍，n 是 6 的整数倍，k 是 32 的整数倍。相比之前形状固定的简单微内核(m=6,n=8)，增加了两层外循环，并实现了计算与规约操作的重叠。

2.动机

NT 模式下，规约操作的引入导致计算指令在微内核中占比不足，程序性能因此低下。为了隐藏规约的开销，需要合并众多小微内核，并对它们的计算和规约进行重叠布置。考虑到 m=3,n=6 是能够充分利用 FMAC 部件的最小形状，所以选择 m=3,n=6 作为合并前微内核的形状。

3.方法

在 NT 模式下，最后一级缓冲区的大小一般是微内核形状的若干倍，所以在外 kernel 循环的最内层，需要连续调用若干次微内核来完成计算。以 m=3,n=6 的微内核为例，对最后一级缓冲区的划分如下图所示，其中 C 分块中的单元个数即为微内核要连续调用的次数。

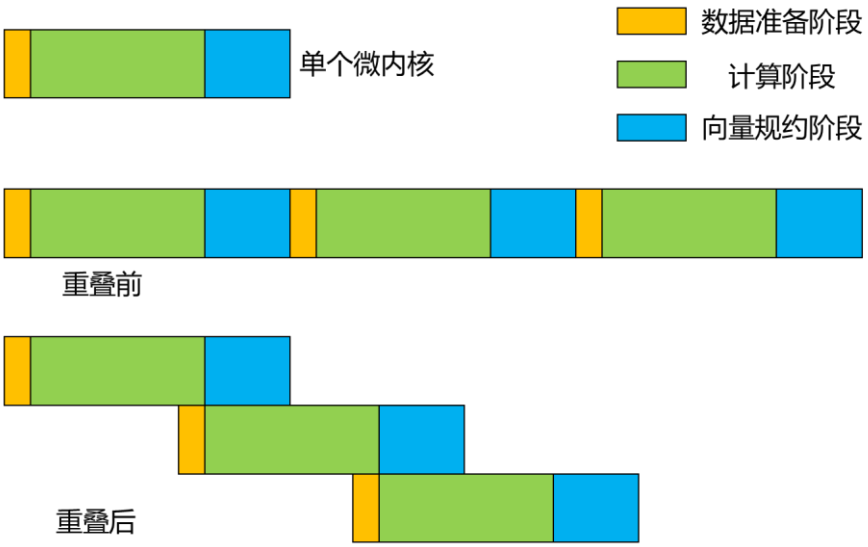


在逐一调用的情况下，微内核两两之间的计算与规约不会有任何重叠。为此，这个版本将调用操作(以微内核形状为步长，遍历最后一级缓冲区的两层循环)也移到了微内核之内，各个独立的微内核被合并。于是，外 kernel 的最内一层就只需要调用一次微内核，变化如下图。

```
for m = 0:3:M_S
    for n = 0:6:N_S
        micro_kernel_NT_asm_r3c6_V1(src_A+m*(K_A>>4), src_B+n*(K_A>>4), dst_C+m*N_S+n, K_A);
    end n
end m
```

```
micro_kernel_NT_asm_r3c6_V2(src_A, src_B, dst_C, K_A, M_S, N_S);
```

合并使得微内核的不同阶段能够实现前后重叠。NT 模式下单个微内核由三个阶段组成，分别是数据准备、计算和向量规约。重叠后的执行顺序如下图所示，在本轮计算的前半段，对上一轮的计算结果进行规约，在本轮计算的后半段，对下一轮的数据进行准备。



4. 开销

合并+重叠以后的微内核可以分为如下三个阶段：

阶段	操作	Cycles
数据准备	计算常量、地址、偏移量， 初始化基址寄存器、偏移寄存器、结果寄存器， 加载第一轮计算要用到的数据	11
主循环	计算本轮的数据， 同时对上一轮结果进行规约和写回， 对下一轮计算进行数据准备	(14+(k/32)*12) *(m/3)*(n/6)
向量规约+结果写回	对最后一轮的计算结果进行规约和写回	81

其中规约操作的开销又可细分为：

向量规约 与 表量写回 子阶段	Cycles
等待计算流水排空	3
第一次写 AM	4
第一次模 4 读 AM	9
第一次向量加	15
第二次写 AM	3
第二次模 4 读 AM	9
第二次向量加	12
取标量，累加原数据，写回	26
总计	81

由此可知，微内核执行的周期数是：

$$11+\left(14+\frac{3k}{8}\right)\times\frac{m}{3}\times\frac{n}{6}+81$$
$$=92+\left(14+\frac{3k}{8}\right)\times\frac{m}{3}\times\frac{n}{6}$$

微内核相对于峰值性能的效率是：

$$\frac{1}{1+\frac{8}{3k}(14+\frac{92}{\frac{m}{3}\times\frac{n}{6}})}$$

5.性能测试

5.1 微内核周期数测试

通过 get_clk 函数对微内核单次执行的周期数进行测试，结果如下：

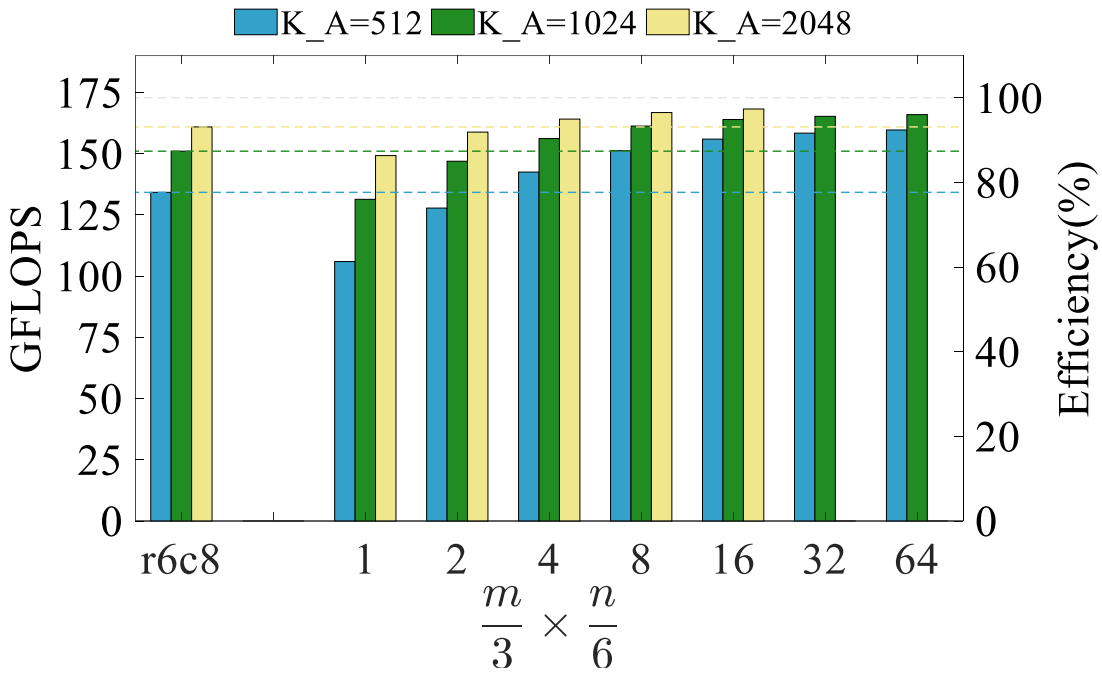
形状参数	理论周期数	实测周期数
m/3=1, n/6=1, k=512	298	326
m/3=4, n/6=4, k=512	3388	3416
m/3=2, n/6=2, k=1024	1684	1712

已知 get_clk 耗时 21 拍，微内核调用跳转延时为 7 拍，可以验证：

理论周期数 = 实测周期数 - get_clk - 跳转延时

5.2 微内核 GFLOPS 测试

再对微内核进行 GFLOPS 测试，K_A 分别取 512,1024,2048，合并的微内核个数$(\frac{m}{3}\times\frac{n}{6})$从 1 逐渐到 64，性能如下图所示。作为对比，其中的 r6c8 是上一个版本单个微内核(形状固定 m=6,n=8)的性能。



5.3 kernel 性能测试

将微内核放入 kernel，测试算子的整体性能。K_A 分别取 512,1024,2048，其他分块参数做到充分利用片上缓存。在输入矩阵足够大的情况下，平均单核性能随计算核数的变化如下图所视。作为对比，虚线表示使用上一版没有经过合并+重叠的微内核时，算子的性能。