Red-blue pebbling revisited: near optimal parallel matrix-matrix multiplication

(并行算法设计与分析 课程研究报告)

第八组：吴坎 刘德钦 魏琢艺

中山大学 计算机学院

二〇二二年四月

目录

稠密矩阵乘（general matrix-matrix multiplication, GEMM）

$C = A \times B,\ A \in \mathbb{C}^{m \times k},\ B \in \mathbb{C}^{k \times n}, C \in \mathbb{C}^{m \times n}$

GEMM 是并行计算/高性能计算领域关注的经典问题

线性代数/机器学习中最重要的算子之一。

问题的规模与并行的规模正在不断变大

- 以 BERT、GPT 为代表的模型越来越往“大”的方向发展。
- E 级超算离现实越来越近。
- 新型计算设备不断推出，GPU、NPU、TPU、FPGA…

原有的 PGEMM (Parallel GEMM) 算法存在局限性

- 并行度低，难以充分利用多节点、异构处理器的多核算力。
- 计算效率低，问题规模较大时计算访存比上升。
- 通信量大，大规模多进程并行中成为瓶颈。
- 矩阵存储方式受限，难以便捷集成进现有应用。
- 分块逻辑复杂，不能整除时处理过程繁琐。
- 并行策略固定，需要手动调优。

矩阵乘法的两种理解方式

- $m \times n$ 个独立的 k 维向量内积。
- k 个 GER (rank-1 update) 的结果累加。

经典算法：Cannon[1]、PUMMA[2]、SUMMA[3]

- 主要对 C 进行 2D 划分。
- 每个进程计算一个 C 的分块。

[1] Lynn Elliot Cannon. 1969. A Cellular Computer to Implement the Kalman Filter Algorithm. Ph.D. Dissertation.

[2] Jaeyoung Choi, David W Walker, and Jack J Dongarra. 1994. PUMMA: Parallel universal matrix multiplication algorithms on distributed memory concurrent computers. Concurrency: Practice and Experience 6, 7 (1994), 543–570.

[3] Robert A Van De Geijn and Jerrell Watts. 1997. SUMMA: Scalable universal matrix multiplication algorithm. Concurrency: Practice and Experience 9, 4 (1997), 255–274.

增大并行度

m、n 较小，k 很大的时候 2D 算法难以充分利用多核算力。

增加计算访存比

矩阵乘法计算访存比为 $2mnk/(mk+kn+mn)$，只对 m 和 n 并行会使其趋近于 1。

减少通信量

减少 A、B 矩阵的通信量，但需要额外交换 C 的数据。

2.5D 算法改进：引入参数 c，调节内存用量

$c=1$，退化为 Cannon 算法。

CARMA[4] 算法：简单、优雅的并行策略

- 将当前问题 m、n、k 中最大的一维划分为两半。
- 内存足够，所有进程对半分到两个子问题。
- 内存不够，所有进程依次解决两个子问题。
- 渐进通信复杂度达到了 PGEMM 的通信复杂度理论下界。
- 无需用户指定，根据问题自动切分维度。
- 切分奇数维度逻辑复杂。
- 实际应用中难以找到高效存储矩阵的方式。

[4] J. Demmel et al. 2013. Communication-Optimal Parallel Recursive Rectangular Matrix Multiplication. In IPDPS.

SC19 Best Student Paper Award[5]

- COSMA 同样是统一 1D、2D、3D 的算法
- COSMA 实测性能较 CARMA “高出一截”
- 作者称，在绝大多数情况下
  - COSMA 可以“贴着”并行矩阵乘法的通信复杂度理论下界
  - 而不是“摸着”下界（差一个常倍数）

COSMA 的论文表述与代码实现有差异

我们将分别介绍

[5] Grzegorz Kwasniewski, Marko Kabić, Maciej Besta, Joost VandeVondele, Raffaele Solcà, and Torsten Hoefler. 2019. Red-blue pebbling revisited: near optimal parallel matrix-matrix multiplication.

核心思想

- 每个进程计算 $a \times a \times b$ 的子问题；通过求解一个最优化问题，找到给定可用内存大小 S 限制下的串行最优 (a, b) 解。
- 根据 red-blue pebbling game 理论，对于计算依赖有向无环图进行划分，得到通信代价最小的并行计算策略。

Algorithm 1 COSMA

Input: matrices $A \in \mathbb{R}^{m \times k}$, $B \in \mathbb{R}^{k \times n}$,
number of processors: p, memory size: S, computation-I/O tradeoff ratio ρ
Output: matrix $C = AB \in \mathbb{R}^{m \times n}$

$a \leftarrow FindSeqSchedule(S, m, n, k, p)$ ▷ sequential I/O optimality (§ 5)
$b \leftarrow ParallelizeSched(a, m, n, k, p)$ ▷ parallel I/O optimality (§ 6)
$(\mathcal{G}, a_{opt}, b_{opt}) \leftarrow FitRanks(m, n, k, a, b, p, \delta)$
for all $p_i \in \{1 \dots p\}$ **do in parallel**
  $(A_l, B_l, C_l) \leftarrow GetDataDecomp(A, B, \mathcal{G}, p_i)$
  $s \leftarrow \left\lfloor \frac{S - a_{opt}^2}{2a_{opt}} \right\rfloor$ ▷ latency-minimizing size of a step (6.3)
  $t \leftarrow \left\lceil \frac{b_{opt}}{s} \right\rceil$ ▷ number of steps
  for $j \in \{1 \dots t\}$ **do**
    $(A_l, B_l) \leftarrow DistrData(A_l, B_l, \mathcal{G}, j, p_i)$
    $C_l \leftarrow Multiply(A_l, B_l, j)$ ▷ compute locally
  end for
  $C \leftarrow Reduce(C_l, \mathcal{G})$ ▷ reduce the partial results
end for

Figure 1: COSMA 伪代码

The local domain $\mathcal{D}_j$ is a grid of size $[a \times a \times b]$, containing b outer products of vectors of length a. The optimization problem of finding $\mathcal{P}_{opt}$ using the computational intensity (Lemma 2) is formulated as follows:

$$\text{maximize } \rho = \frac{a^2 b}{ab + ab + a^2} \quad (3)$$
subject to:
$$a^2 \le S \text{ (the I/O constraint)}$$
$$a^2 b = \frac{mnk}{p} \text{ (the load balance constraint)}$$
$$pS \ge mn + mk + nk \text{ (matrices must fit into memory)}$$

The I/O constraint $a^2 \le S$ is binding (changes to equality) for $p \le \frac{mnk}{S^{3/2}}$. Therefore, the solution to this problem is:

$$a = \min\left\{\sqrt{S}, \left(\frac{mnk}{p}\right)^{1/3}\right\},\ b = \max\left\{\frac{mnk}{pS}, \left(\frac{mnk}{p}\right)^{1/3}\right\} \quad (4)$$

The I/O complexity of this schedule is:

$$Q \ge \frac{a^2 b}{\rho} = \min\left\{\frac{2mnk}{p\sqrt{S}} + S, 3\left(\frac{mnk}{p}\right)^{\frac{2}{3}}\right\} \quad (5)$$

Figure 2: 最优化 (a, a, b)

核心思想

- 穷举 dm, dn, dk 使 $m/dm \approx n/dn \approx k/dk$，并分解质因数。
- 根据内存用量限制，决定每一步在 m, n, k 其中哪个方向进行并行划分或者串行划分。

```
343     for (int i = 0; i < divisors.size(); ++i) {
344         int div = divisors[i];
345
346         if (step_type[i] == 'p') {
347             if (!split_A(i)) {
348                 memory_A.push_back(math_utils::divide_and_round_up(m * k * div, P));
349             } else if (!split_B(i)) {
350                 memory_B.push_back(math_utils::divide_and_round_up(k * n * div, P));
351             } else {
352                 memory_C.push_back(math_utils::divide_and_round_up(m * n * div, P));
353             }
354             P /= div;
355         }
356
357         m /= divisor_m(i);
358         n /= divisor_n(i);
359         k /= divisor_k(i);
360     }
```

Figure 3: https://github.com/eth-cscs/COSMA/blob/v2.5.1/src/cosma/strategy.cpp#L343

条条大路通罗马

- 代码中的 COSMA：CARMA 算法的精神续作，解决了其最大痛点：对半划分。
- 求解最优化问题、最优化 (a, b) 等，并没有在代码中出现。
- 论文和代码各走一条道，双双达到最优通信代价，相较之前的工作提升了至多 $8.2\times$。

Questions?

Thank you!