OpenMP 版本矩阵矩阵乘

黄浩睿 2018202143

问题描述

编写一个程序，使用 OpenMP 优化单精度浮点下矩阵乘法（SGEMM）的性能。

给定测试环境为 i9-10900X @ 3.70GHz x 10（20 线程），L1d/L2/L3 Cache 分别为 32K/1M/19M。

算法

基本的算法为分块矩阵乘法。将矩阵按行和列分为大小相等的边长为 S 的块，每行/每列共 B/S 个块。将块视为矩阵元素进行矩阵乘法，结果矩阵也按照同样的大小进行分块。**这使得每个块的结果可被分别计算，线程之间不会有写冲突**。

针对系统的架构进行优化，将 B 矩阵转置，且对内存进行重新布局（在计算函数中完成，输入输出仍为原始行列布局），使得每个块的内存地址连续（这有助于充分利用 Cache）。经过计算，大小为 S = 256 的块，所占空间为 256K，足够把 A、B 和结果 C 三个矩阵的一个块都放入 L2 Cache 中。且块中有多行可以放入 L1d Cache 中。所以这种分块方式对 Cache 的利用率较高。**这部分过程同样可以并行操作，同样对每个块进行并行**。（这里假设了矩阵大小是块大小 256 的倍数，实际情况中如果不是则可以向上取整。）

循环展开与向量化，由 GCC 编译器进行自动优化（实测中手动编写的 AVX512 代码效率更差，可能是因为使用方式不正确）。

评估

在服务器上对该算法实现进行测试，指定不同的矩阵大小，经过 10 次测试取平均值后发现，当矩阵大小增加时，该并行算法的性能会上升，所测试的最大矩阵大小为 8192，测试结果为 **527 GFLOPS**，详细数据如下:

注：朴素算法为下发文件中的参考算法，使用同样的编译参数。

矩阵大小 / 算法 / GFLOPS	朴素算法	优化算法（单核）	优化算法（多核）
1024	0.635845	45.109409	270.331218
2048	0.567655	48.501849	380.267311
4096	0.390006	52.486537	493.445938
8192	-	54.977634	527.603663
16384	-	46.177147	522.223751
32768	-	46.415970	515.280754

结论

在与朴素算法的对比中，优化算法在单核时有 140 倍的加速比，在多核时可达 1300 倍的加速比，性能提升十分显著。

将优化算法的单核与多核性能相比较，在矩阵大小足够大时，其并行化之后的加速比约为 10 倍，由于 Intel 的 Turbo Boost 与 Hyper Thread 技术带来的影响，该加速比可能不准确，但基本可以认为其等于 CPU 核心数。所以该算法的并行程度是较高的。

在矩阵较小时，优化算法的多核性能相对较差，推测是由于对内存重新布局的过程（平方复杂度）占用了相对更多的时间，而这部分算法对 Cache 友好程度较差，所以影响了整体效率。在对没有进行内存重新布局的算法实现进行测试时，较小的矩阵性能是更优的。而矩阵大小增加时，矩阵乘法本身占用的时间比例大幅增加，所以内存重新布局带来的性能提升更加明显。由于实际使用并行矩阵乘法时矩阵大小一般较大，所以该优化是合理的。

根据资料，该 CPU 进行 SGEMM 时的最高 GFLOPS/s 为 1500。本优化算法的性能仅仅达到了理论值的 1/3，推测其原因是对于 Cache 的优化不够，有很大的资源浪费到了访存上。