稀疏矩阵求解 实验报告

2017310723 计研173 古裔正

一、实现

使用OpenMP实现，该项目在Mac Pro的Xcode下搭建。

在并行化的过程中，最需要优化的部分是n\*n矩阵和n维向量相乘的部分，这是每次迭代中唯一的O(n2)的操作。在使用openmp进行优化的过程中，需要注意线程调度的方式，由于矩阵比较稀疏，每一行的运行速度可能会不同，所以我采取了动态调度的方式，以20行为一个chunk，下面这段代码是稠密矩阵做法的openmp加速片段：

#pragma omp parallel for schedule(dynamic, 20)

for (int i = 0; i < N; i++) {

for (int j = 0; j < N; j++) { ……

特别的，这一题中的矩阵是稀疏矩阵，我尝试了稠密矩阵的存储方式和稀疏矩阵的存储方式，并分别验证了这两种方法在1、2、4、8核情况下的加速比。下面是稀疏矩阵做法的openmp加速片段：

#pragma omp parallel for schedule(dynamic, 20)

for (int r = 0; r < N; r++) {

int begin = 0, end = rowIndex[r];

if (r > 0) {

begin = rowIndex[r - 1];

}

for (int i = begin; i < end; i++) { ……

二、测试

在测试中，我随机生成了一些500\*500的矩阵A，矩阵A是对称正定的，每行期望有10个非零元素，因此比较稀疏。然后，随机生成了答案500维向量x，再计算的b=Ax，并将A和b作为我们程序的输入，检测程序的输出和x是否相同。

对于稠密矩阵做法和稀疏矩阵做法，对于每个线程数（1、2、4、8），我都生成了20组数组，验证他们的速度、迭代次数、是否正确等等。

在正确性方面，160组数据均在1000次迭代左右出解，并且答案正确。



上图是在八种情况下程序运行时间（平均值和标准差），在稠密矩阵做法中，单线程平均1.97s，双线程1.06s（加速比185.85%），四线程0.574s（加速比343.21%），八线程0.632s（加速比311.71%）。在稀疏矩阵做法中，单线程平均0.0899s，双线程0.0659s（加速比136.42%），四线程0.0532s（加速比168.98%），八线程0.0756s（加速比118.92%）。这台机子是4核8线程的i7处理器。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 稠密存储方式 | | 稀疏存储方式 | |
|  | 时间(s) | 加速比 | 时间(s) | 加速比 |
| 1 | 1.97 |  | 0.0899 |  |
| 2 | 1.06 | 185.85% | 0.0659 | 136.42% |
| 4 | 0.574 | 343.21% | 0.0532 | 168.98% |
| 8 | 0.632 | 311.71% | 0.0756 | 118.92% |

可见在稠密矩阵存储方式中，并行的加速比较大；稀疏矩阵存储方法中并行的加速比要少一些。而我的这台机子在4线程的情况下性能最优，8线程反而效果变差。