高斯消元法的MPI实现

计算机科学与技术
1511186
梁宸

February 25, 2020

1 简介

本次作业实现了基于MPI的高斯消元算法的C++程序, 结合前几次作业的AVX和OpenMP, 并参考了作业说明中的文档实现了基于二维矩阵划分的流水线优化. 运行环境仍为Linux Mint 18.4 64bit, Intel Core i7.

2 算法设计

2.1 数据分配

在之前实现的多线程版本中, 每个工作线程所占有的子矩阵是随轮数变化的. 而在多进程的环境下, 可以认为进程间数据通信开销较大, 甚至在集群环境下会跨越物理机器通信. 因此每个进程所占用的子矩阵应该在初始分配后不再变化. 本次实现使用master-worker模式, 假设在开始时输入矩阵全部在master进程. 初始时master会尽可能将矩阵分为等份, 再将子矩阵的位置和内容发送给对应的worker进程.

2.2 工作流程

由于每个进程的子矩阵位置是固定的, 随着算法轮数增大, 当前活跃的矩阵部分越来越向右下角收缩. 每个进程在每一轮开始时, 先检查自己的子矩阵是否完全不活跃, 如果是, 则该进程可以将计算结果发送回master进程, 并结束工作. 否则继续下一步.

接下来进行基于流水线优化的工作流程, 其基本原理作业说明中已给出介绍. 每个进程每轮都需要传播部分行和列. 进程检查自己是否在当前活跃矩阵的最左处, 如果是, 那么它要负责将它自己子矩阵最左面的一列传播给它右侧的下一个进程; 否则它需要接收它左侧的进程发送来的列, 并转发给右侧; 相似地, 接下来每个线程检查自己是否为当前活跃矩阵的最上方, 如果是, 则它负责先将子矩阵最上方的行进行除法工作[1], 之后将该行发送给它下方的第一个进程; 如果该

[1]即作业说明文档中的division阶段.

进程不在最上方, 那么它需要等待接收上方进程发送来的行, 并将它转发给下方进程.

随后, 进程会用上述特定的行和列进行消去工作[2]. 这部分和上次作业的OpenMP版本相似, 即将每个进程的子矩阵按行划分给多个线程, 每个线程再用AVX指令将行消去, 消去时会用到上一步中得来的特定行和列中的元素. 消去工作结束后, 进程可以开始下一轮.

2.3 结果收集

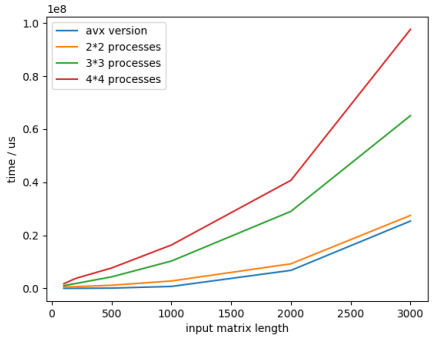
master进程在初始分配并发送原始矩阵后, 就开始接收工作进程的结果. 每个结果中包含子矩阵的位置和内容, master进程收到一个结果后, 就开始写原始矩阵中的对应位置. 事实上如果一个进程分配到的子矩阵在下三角, 那么它的内容一定是全0, 因此没有必要发送回master进程.

2.4 进程同步

本实现中采用MPI的阻塞recv, 因此进程间的同步可以在数据的send和recv同时实现. worker和master之间通过输入的子矩阵和计算结果的send/recv来同步. worker之间通过流水线中行和列的send/recv来同步. 这样每个进程都会在试图接收必要的数据时阻塞, 因此不会有更额外的同步开销.

3 实验结果与分析

设置不同的工作进程数和输入矩阵大小, 测量结果如下:



事实上在单机情况下, 测量针对集群的MPI程序性能意义不大. 反而会因进程和其子线程之间争夺系统资源, 进程通信, 以及进程创建销毁等而产生更大的额外开销. 结果中可以看到随着进程数增多, 在单机情况下的性能变差, 这样的结果其实是显而易见的.[3]

[2]即作业说明文档中的elimination阶段.

[3]其实感觉都按照作业说明上给出的流水线算法和OpenMP写的话, 大家最后的性能应该没什么区别.