中国科学技术大学计算机学院

《计算机系体结构实验报告》

实验题目：数据级并行

学生姓名：柯志伟

学生学号：PB20061338

完成时间：2022年6月10日

【实验题目】

数据级并行

【实验目的】

以矩阵乘法作为切入点，在CPU与GPU平台上分别实现矩阵乘法的数据并行算法，并对比分析各种算法间与算法内部参数对程序效率的影响，具体如下：

- CPU实验

  1. 实现CPU基础矩阵乘法，AVX矩阵乘法、AVX分块矩阵乘法
  2. 改变输入规模，对比分析三种实现的程序的性能
  3. 分析分块大小对分块矩阵乘法性能的影响

- GPU实验

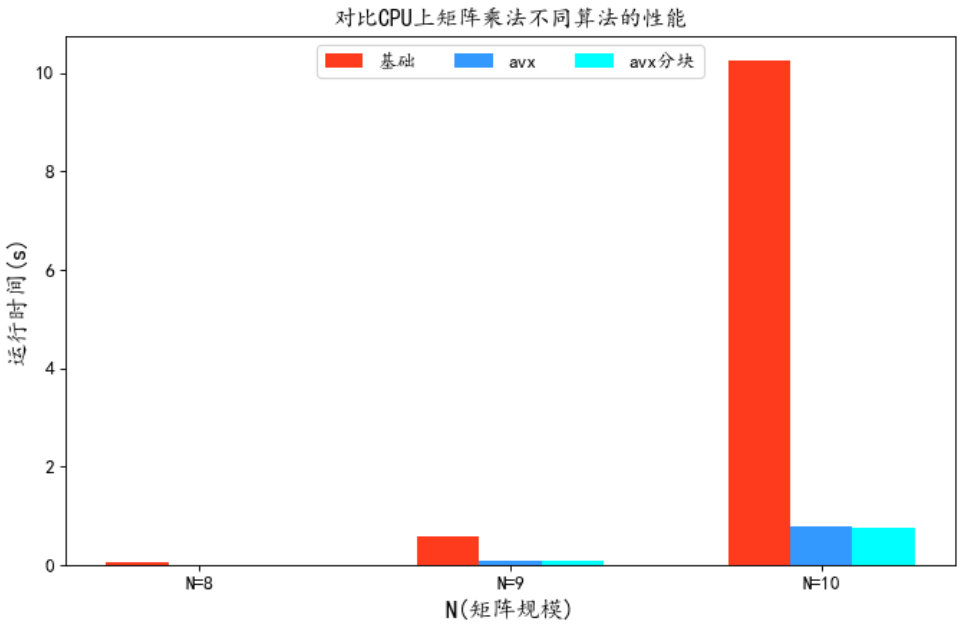
  1. 实现GPU基础矩阵乘法和GPU分块矩阵乘法
  2. 改变输入规模，对比分析两种实现的程序的性能
  3. 不同的gridsize 和 blocksize 对基础矩阵乘法的影响
  4. 不同的gridsize 和 blocksize以及BLOCK对分块矩阵乘法性能的影响

【实验环境】

CPU平台　GPU平台

【实验过程】

- CPU实验
  - 改变输入规模，对比分析三种实现的程序的性能



实验中在矩阵规模为N(8,9,10)下(size大小为1<<N),分别测定了三种算法的性能(具体以计算出初始矩阵A,B的乘积C所消耗的时间为基准),其中分块矩阵此时的blocksize为256

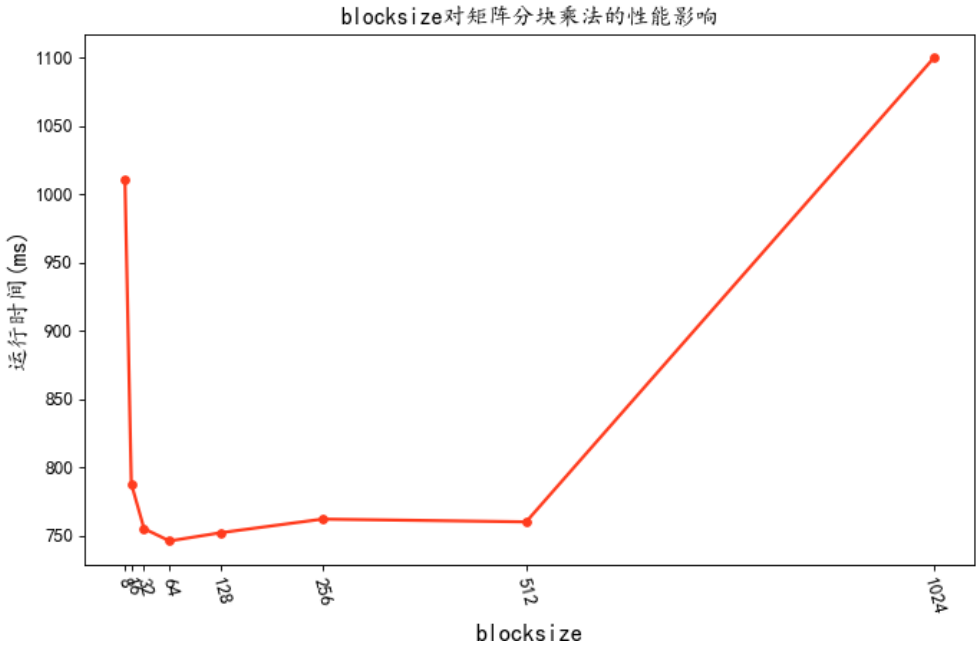
分析：由图可知，三种方案在性能上大体上是avx分块优于avx,avx又优于基础,且从图中可知,随矩阵规模的增大,计算量大幅增长,基础矩阵乘法所消耗的时间远高于另外两种乘法,性能较差

原因：

1．直接采用经典算法没有利用数据间的并行性，随着数据规模的增长，所执行的指令大量增加,所需时间会迅速增长。而avx算法进行了向量化，利用了数据间的并行性,大大减少了所需时间。

2．普通avx算法访存跨度较大，并未充分利用cache的局部性，avx分块矩阵算法则弥补了这一点，通过对输入数据进行分块尽可能
使内存访问连续，提高cache命中率，进而提升程序的整体性能。

- 分析分块大小对分块矩阵乘法性能的影响



实验中固定矩阵规模为N=10,更改blocksize的大小,观测程序性能

分析：由图可知,在blocksize过大和过小都会降低程序的性能,在合适的blocksize下,分块矩阵的性能最优

原因：avx分块矩阵乘法优于avx矩阵乘法，主要是利用了cache的局部性原理,在blocksize过小时,每个block中元素较小，利用cache局部性的能力有限,而当blocksize过大时,每个block取代原矩阵成为新的大块,无法利用cache的局部性原理，故合适的blocksize性能最优

- CPU平台上矩阵乘法优化手段调研

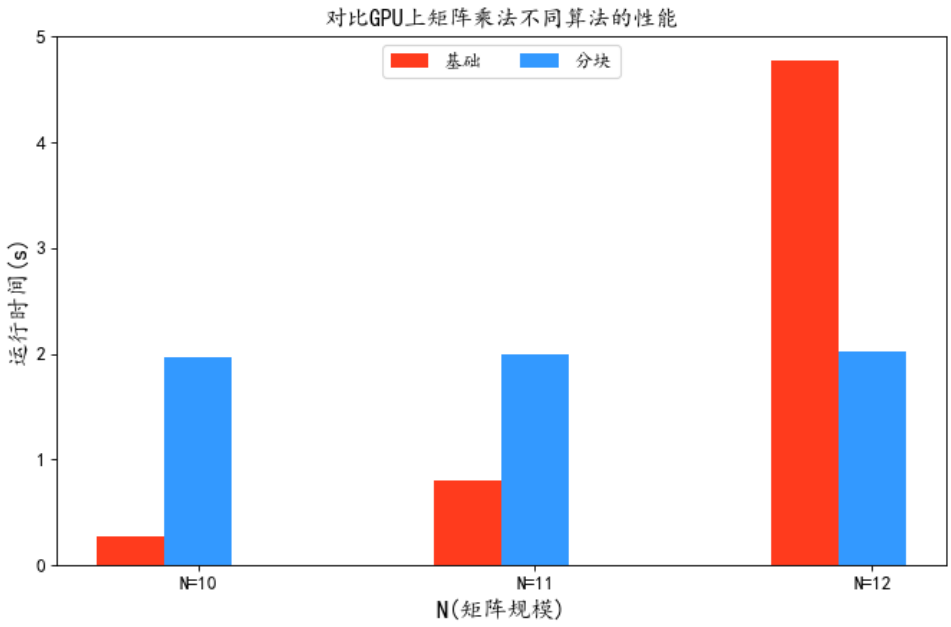
1. 改进运算顺序的矩阵乘法

调整矩阵乘法的运算顺序,更好地利用cache的局部性原理

2. 调用OMP利用cpu并行

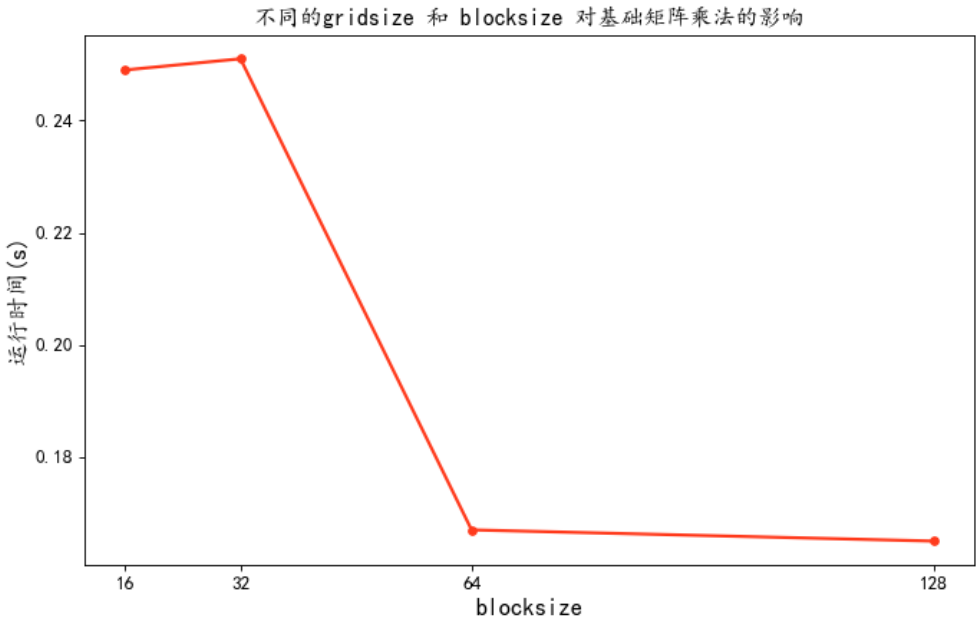
> OpenMP使用FORK-JOIN并行执行模型。所有的OpenMP程序开始于一个单独的主线程。主线程会一直串行地执行，一旦遇到并行域会根据机器情况和编译配置属性实现多线程并行，特别适和对计算密集型任务的优化。

- GPU实验
  - 改变输入规模，对比分析两种实现的程序的性能



> 实验中在矩阵规模为N(10,11,12)下(size大小为1<<N),分别测定了两种算法的性能,其中blocksize为32,gridsize为N/blocksize
>
> 分析：由图可见，当矩阵规模较小时，普通GPU算法性能比分块相乘更好，但当矩阵规模增大时，普通GPU算法性能会急剧下降，而分块算法变化不大。
>
> 原因：GPU分块矩阵加速原理主要是利用局部性原理，使用Shared Memory提高了访存速度。当矩阵规模较小时，需要传输的数据较少，GPU在传输数据上所用时间不多，因此两种算法差别不大。但当矩阵规模增大到一定程度时，GPU用在传输数据上的时间会迅速增长，采用分块矩阵算法时，Shared Memory位于每个Block内部，Block内部的Thread共享该内存，线程计算时无需从Global Memory取数据，节省数据传输时间，大大提高了性能。

  - 不同的gridsize 和 blocksize 对基础矩阵乘法的影响

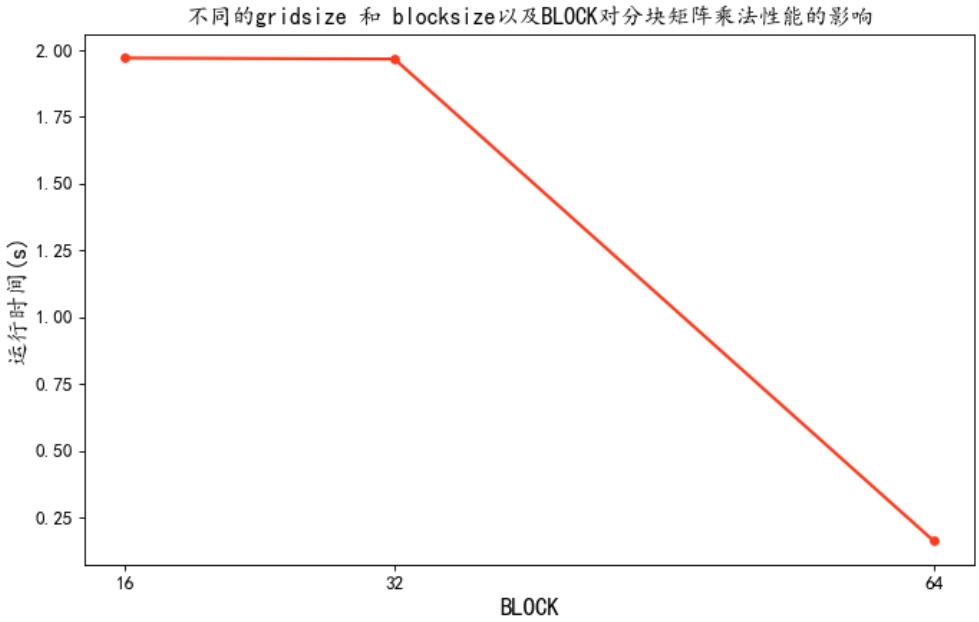


实验中,blocksize*gridsize=N,改变blocksize观测程序的性能变化,设定N=10

分析：由图可见，当 blocksize 小于等于32时，程序执行时间较长，当 blocksize 大于32后，程序执行时间明显下降。

原因：cuda程序中的每个block是运行在GPU中的一个SM上的，每个线程块的线程数太少，那么由于SM同时执行的线程块数量有限，这就导致SM同时执行的线程数不够，这将导致GPU资源占用率低，从而降低性能。由于thread以warp（32个thread）为单位跑在SM上，因此一般而言,blocksize 应当大于32

- 不同的gridsize 和 blocksize以及BLOCK对分块矩阵乘法性能的影响



> 实验中,每个block处理一个分块矩阵的运算,参数BLOCK的大小等于分块矩阵的大小,设定为blocksize,另外，由于受到Shared Memory大小限制， blocksize 最大只能达到64，因此 blocksize 取16，32，64进行测试,设定N=10
>
> 分析：当 blocksize 小于等于32时，程序执行时间较长，当blocksize到64后，程序执行时间明显下降
>
> 原因：分块矩阵算法中，每个thread不直接从global memory中读取数据，而是从sharedmemory中读取数据，因此在 blocksize 较小时，访存时间变化不大。同时由于 blocksize 小于等于32时，每个线程块的线程数太少，那么由于SM同时执行的线程块数量有限，这就导致SM同时执行的线程数不够，这将导致GPU资源占用率低，从而降低性能。当 blocksize 大于32后，GPU资源占用率得到提高，因此性能较高。

【总结与思考】

- 问题

  1. 使用AVX库的编译链接问题

  > 查阅网上资料要加上特定的编译选项

  2. 使用GPU的程序时间测量方法

  > 网上调研使用nsys的方法

- 收获

  通过本次实验，在CPU的实验中,了解到使用CPU实现数据级并行的方法,熟悉了多媒体扩展指令集实现数据级并行的模型;在GPU的实验中使用CUDA编程，加深了对GPU结构的理解，更了解GPU如何组织多个thread执行、GPU的内存结构等。

  除此之外,体会到数据级并行在数据量庞大的应用程序中的性能优化能力,提高了动手与思考能力。

【备注】

实验过程中的截图附带在文件夹下