lab2 实验报告

课程名 高性能计算应用实践

学期 2024年秋季学期

姓名 陈卫喆

学号 2023311F13

一. 实验环境

OS版本

Ubuntu 22.04.3 LTS
5.15.153.1-microsoft-standard-WSL2

gcc版本

gcc (Ubuntu 11.4.0-1ubuntu1~22.04) 11.4.0

cpu型号

13th Gen Intel(R) Core(TM) i5-13500H

频率

cpu MHz : 3187.200

物理核数

Core(s) per socket: 8
Socket(s):1

内存大小

	total	used	free	shared	buff/cache	available
Mem:	8028508	833652	6938612	3268	256244	6958172
Swap:	2097152	0	2097152			

二. test_cblas_dgemm.c

改为行主序前，test_cblas_dgemm输出结果为

-4.000000 11.000000 0.000000 11.000000 -9.000000 5.000000 8.000000 5.000000 6.000000

更改后（更改了CblasColMajor, lda, ldb, ldc四个参数），输出结果为

-4.000000 11.000000 0.000000 11.000000 -9.000000 5.000000 8.000000 5.000000 6.000000

可以看出两次输出结果一致

三. time_dgemm.c

	256	1024	4096	8192
cblas_dgemm duration	0.033572 s	0.045022 s	0.928733 s	7.339119 s
naive_dgemm duration	0.039306 s	4.935227 s	1270.319921 s	11123.249346 s
cblas_dgemm gflops	1.998953 GFLOPS	95.397079 GFLOPS	295.970862 GFLOPS	299.630413 GFLOPS
naive_dgemm gflops	1.707344 GFLOPS	0.870267 GFLOPS	0.216385 GFLOPS	0.197696 GFLOPS

分析如下

cblas算法运行时间随数组规模增大而增大，这是显然的，而gflops值也随数组规模增大而增大，并存在边际效应，输入数组规模越大，算法效率越高
自己编写的naive_dgemm算法的gflops值与cblas相比很低，并随数组规模增大而减小，最后甚至趋近于0，其效率很低

cblas算法在花费时间和效率上明显优于naive_dgemm算法

四. 碰到的问题及解决办法

1. 输入uname -a等命令获取OS版本信息时，显示内容过多，英语能力和专业基础不太过关导致不清楚哪些才是真正需要的信息

询问gpt解决

2. 输入编译、运行命令时，直接复制pdf中的命令，而执行出错

将其中的-和_都改为英文的-和_后解决 (很难排查出的错误，以后遇到类似情况自己敲一遍就好了)

3. 修改time_cblas_dgemm.c为行主序时，执行结果显示错误信息并且结果不同

对接口不够熟悉，仅修改了CblasColMajor这一个参数，更改lda, ldb, ldc后解决

4. time_dgemm.c, 编写并运行naive_dgemm的过程中遇到问题较多

一开始直接将dgemm_naive.c中的dgemm函数粘贴到time_dgemm.c, 并修改接口和main函数中使用的函数，编译时仍出错，
报错信息表明A, B, C三个数组及beta的类型不匹配
复习了二维数组和指针的关系，修改函数及接口后解决，主要是程序跑起来很慢... 因而对规模较小的输入(256, 1024)取了多次的平均值，
而对较大输入(4096, 8192)仅跑了一次

五. 代码和报告已提交至github仓库

https://github.com/Asazuk1/HPC.git