高性能计算程序设计（3） 秋季2023

**提交格式说明**

按照实验报告模板填写报告，需要提供源代码及代码描述至https://easyhpc.net/course/181。实验报告模板使用PDF格式，命名方式为高性能计算程序设计\_学号\_姓名。如果有问题，请发邮件至韦媛馨（课程、实验）3366875159@qq.com钟赫明（课程、实验）zhonghm5@mail2.sysu.edu.cn询问细节。

1. **通过OpenMP实现通用矩阵乘法**

通过OpenMP实现通用矩阵乘法（Lab1）的并行版本，OpenMP并行线程从1增加至8，矩阵规模从512增加至2048。

通用矩阵乘法（GEMM）通常定义为：

输入：M , N, K三个整数（512 ~2048）

问题描述：随机生成M\*N和N\*K的两个矩阵A,B,对这两个矩阵做乘法得到矩阵C.

输出：A,B,C三个矩阵以及矩阵计算的时间

1. **基于OpenMP的通用矩阵乘法优化**

分别采用OpenMP的默认任务调度机制、静态调度schedule(static, 1)和动态调度schedule(dynamic,1)的性能，实现#pragma omp for，并比较其性能。

1. **构造基于Pthreads的并行for循环分解、分配和执行机制。**
2. 基于pthreads的多线程库提供的基本函数，如线程创建、线程join、线程同步等。构建parallel\_for函数对循环分解、分配和执行机制，函数参数包括但不限于(int start, int end, int increment, void \*(\*functor)(void\*), void \*arg , int num\_threads)；其中start为循环开始索引；end为结束索引；increment每次循环增加索引数；functor为函数指针，指向的需要被并行执行循环程序块；arg为functor的入口参数；num\_threads为并行线程数。
3. 在Linux系统中将parallel\_for函数编译为.so文件，由其他程序调用。
4. 将通用矩阵乘法的for循环，改造成基于parallel\_for函数并行化的矩阵乘法，注意只改造**可被并行执行的for循环**（例如无race condition、无数据依赖、无循环依赖等）。

举例说明：

将串行代码：

for ( int i = 0; i < 10; i++ ){

A[i]=B[i] \* x + C[i]

}

替换为------>

parallel\_for(0, 10, 1, functor, NULL, 2);

struct for\_index {

int start;

int end;

int increment;

}

void \* functor (void \* args){

struct for\_index \* index = (struct for\_index \*) args;

for (int i = index->start; i < index->end; i = i + index->increment){

A[i]=B[i] \* x + C[i];

}

}

==========================

编译后执行阶段：

多线程执行

在两个线程情况下：

Thread0: start和end分别为0，5

void \* funtor(void \* arg){

int start = my\_rank \* (10/2)

int end = start + 10/2;

for(int j = start, j < end, j++)

A[j]=B[j] \* x + C[j];

}

Thread1: start和end分别为5，10

….....