高性能计算程序设计基础 秋季 2020

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/121。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 lidsh25@mail2.sysu.edu.cn，leong36@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：

$$\mathrm{C} = \mathrm{AB}$$

$$C_{\mathrm{m,n}} = \sum_{n=1}^{N} A_{m,n} B_{n,k}$$

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

2. 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。

3. 构造基于 Pthreads 的并行 for 循环分解、分配和执行机制。

1） 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。

2） 在 Linux 系统中将 parallel_for 函数编译为.so 文件，由其他程序调用。

3） 将基于 OpenMP 的通用矩阵乘法的 omp parallel for 并行，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

Thread1: start 和 end 分别为 5，10

```
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];
}
```