Foster 并行程序设计

第四次作业 罗仁良 18340126

1. 实验内容

利用Foster并行程序设计方法计算1000 × 1000 的矩阵和1000×1的向量之间的乘积，要求清晰地呈现Foster并行程序设计的四个步骤。

2. 程序设计

2.1 问题分解

记矩阵为$A=[a_1,a_2,a_3,\ldots,a_n]$，其中$a_i$ 是矩阵A第i行构成的行向量，再记向量$b=(b_1,b_2,b_3,\ldots,b_n)$, 所以$A\cdot b=c=(c_1,c_2,c_3,\ldots,c_n)$，其中$c_i=a_i\cdot b$，将$a_i\cdot b$的运算作为一个任务。

```
/* * *
 *  将矩阵与向量乘积分解为 A的行向量与向量b相乘的结果
 *  eg. A*b = c (A为矩阵，b、c为向量) A= (a1, a2, a3, ..., an) , b =
(b1,b2,b3,...,bn)
 *  c=(c1,c2,c3,...cn) , ci= ai*b;
 * * */
void vector_mul(mul_set* A) {
    A->result = 0;
    for(int i=0;i < A->n; i++) {
        A->result += A->vector_row[i]*A->vector_col[i];
    }
}
```

2.2 通信

采用2.1中的方式进行问题划分，不会存在不同进程间的数据通信，只需要把每次计算的结果赋值给保存最后结果的数组即可，每次计算的结果都可以独立计算出结果向量c中的一项，不存在数据依赖。

```
typedef struct mul_set
{
    int * vector_row;
    int * vector_col;
    int n;
    int result;      //保存计算结果
}mul_set;
```

2.3 归并组合

归并组合的主要目的将任务合理重组，减少通信开销等，加快计算，考虑到适应电脑的核心数和线程数，将1000个任务平均分为4组，也就是用四个线程并行执行。

```
typedef struct thread_argv
{
    int **mat;
    int *vector;
    int row;
    int size;
}targv;

void * thread_task(void * argv) {
    targv * p = (targv *) argv;
    mul_set A;
    for(int i=0; i < p->size; i++) {
        A.vector_row = p->mat[p->row+i];
        A.vector_col = p->vector;
        A.n = MAX;
        vector_mul(&A);
        result[p->row+i] = A.result;      //result是一个全局共享的区域用于保存最后计
算结果，分析可知                                    //不会存在数据竞争
    }
    return NULL;
}
```

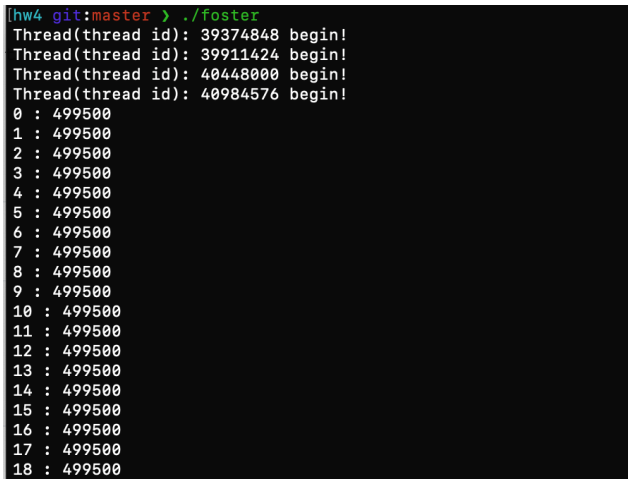
等待所有线程执行结束。

```
    pthread_join(t[0],NULL);
    pthread_join(t[1],NULL);
    pthread_join(t[2],NULL);
    pthread_join(t[3],NULL);
```

2.4 Mapping

实现过程采用多线程计算，最后线程的执行具体是在CPU的哪个核心执行有操作系统分配决定。

3. 运行结果



4. 实验总结

本次实验的难度不是特别大，主要的重点是熟悉了解Foster并行程序设计的方法，加深并行程序设计的理解。在最开始设计的时候，分解任务是按照将矩阵A分解为列向量，在和b中的每一项项成，最后再求和。分析发现这种计算方式需要保存所有任务计算的结果最后等待所有结束之后再求和，不论是内存还是等待结束都有一定的额外开销，存在一定的数据依赖。进一步分析问题，最后采取了本次实验实现的方法，必然了数据依赖带来的额外开销。可见问题的分解，对并行程序设计是一个关键的步骤。