实验五 矩阵向量乘法

庄清惠 14331401

实验目的： 用 pthread 实现矩阵向量乘法。

代码分析：（完整代码请查看.c 文件）

初始化

```
int N, M;
int *matrix;
int *vector;
int *ans;
int *Index;
void* compute(void *);
int main() {
  int i, j;
  scanf("%d%d", &N, &M);
  matrix = (int *)malloc(sizeof(int) * N * M);
  vector = (int *)malloc(sizeof(int) * M);
  ans = (int *)malloc(sizeof(int) * N);
  Index = (int *)malloc(sizeof(int) * N);
  //initialize the array of index and answer
  memset(ans, 0, sizeof(*ans));
  for (i = 0; i < N; ++i)
    Index[i] = i;
```

N, M 表示矩阵大小。动态分配数组 matrix 存放输入的矩阵；vector 存放输入的向量；ans 存放乘积的结果，并初始化为零；index 表示各个线程要计算的行号。

读取数据

```
//read data
for (i = 0; i < N; ++i) {
  for (j = 0; j < M; ++j) {
    scanf("%d", matrix + i * N + j);
  }
}
for (i = 0; i < M; ++i)
  scanf("%d", vector + i);
```

首先输入矩阵行列数，再输入矩阵及向量。

数据处理

创建线程

```
  //handle data
  pthread_t *tids = (pthread_t *)malloc(sizeof(pthread_t) * N);
  for (i = 0; i < N; ++i) {
    pthread_create(&tids[i], NULL, compute, &Index[i]);
  }
```

每个进程按 index 计算得到 ans 中的一个值

```
void* compute(void* arg) {
  const int index = *(int *)arg;
  int i;
  for (i = 0; i < M; ++i) {
    ans[index] += matrix[index * N + i] * vector[i];
  }
  pthread_exit(NULL);
}
```

打印结果

```
  //print the answer
  printf("The answer of the mutiplication is\n");
  for (i = 0; i < N; ++i)
    printf("%d\n", ans[i]);
```

运行结果：

```
zhuangqh@ubuntu:~/Cpp_homework/CUDA/experiment4$ g++ matrix_mutiply.c -o main -lpthread
zhuangqh@ubuntu:~/Cpp_homework/CUDA/experiment4$ ./main
3 3
1 2 3
4 5 6
7 8 9
1 1 1
The answer of the mutiplication is
6
15
24
```

运行结果正确