Report of Assignment 2— OpenMP Programming

Multiprocessor Systems, DV2544

Student: Ye Cai

Personal number: 9503101967

Email address: yeca16@student.bth.se

1 INTRODUCTION

POSIX threads (also called Pthreads) defines a standard for the programming with threads, based on C programming language. The threads of a program share a common address space. Thus, the global variables and dynamically generated data objects can be available by all threads of a process [1].

2 GAUSSIAN ELIMINATION

2.1 Implementation

- Init functions and print function are same with sequential version.
- Basically, in the outer loop (for k=0;k<N;k++) for each k it creates P threads to paralyze the inner loop and eliminate the matrix at inner loop cyclic-rowwisely like Figure 1.

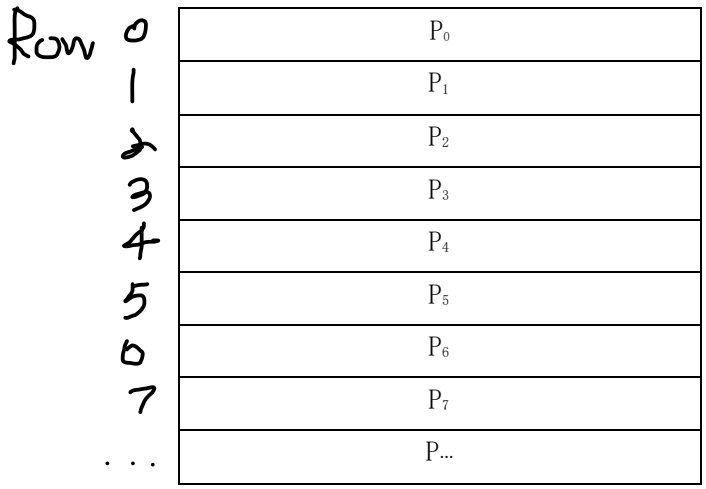

Figure 1: rowwise cyclic data structure

- The elimination steps are separated as two functions elimination1() and elimination2() to deal with two cases one is the inner matrix larger than the

number of threads and one is the inner matrix smaller than the number of threads. So that the elimination1() function could be faster that the sequential version because of using ” i+=p “ to jump over.

```
*when the matrix size is bigger than number of
oid *elimination1(struct Data *data){
    int k=data->k;
    int i,j;
    for(i=data->threadid+k+1;i<N;i+=P){/* Elimi
        for(j=k+1;j<N;j++)
            A[i][j]=A[i][j]-A[i][k]*A[k][j];
        b[i]=b[i]-A[i][k]*y[k];
        A[i][k]=0.0;
    }
```

```
*when the matrix size is smaller than number o
oid *elimination2(struct Data *data){
    int k=data->k;
    int i,j;
    i=data->threadid+k+1;
    for(j=k+1;j<N;j++)
        A[i][j]=A[i][j]-A[i][k]*A[k][j];
    b[i]=b[i]-A[i][k]*y[k];
    A[i][k]=0.0;
```

Figure 2: the two elimination steps.

```
for (i = k + 1; i < N; i++) {
    for (j = k + 1; j < N; j++)
        A[i][j] = A[i][j] - A[i][k] * A[k][j]; /* Elimination step */
    b[i] = b[i] - A[i][k] * y[k];
    A[i][k] = 0.0;
}
```

Figure 3: the sequential version elimination steps

2.2 Measurements

```
yeca16@kraken:~/mp_lab$ time ./gauss_pthread

size      = 2048x2048
maxnum    = 15
Init      = rand
Initializing matrix...done

time elapsed: 35.291472s
real    0m9.387s
user    0m34.191s
sys     0m1.103s
yeca16@kraken:~/mp_lab$ time ./gauss_pthread

size      = 2048x2048
maxnum    = 15
Init      = rand
Initializing matrix...done

time elapsed: 35.767723s
real    0m9.362s
user    0m34.675s
sys     0m1.095s
yeca16@kraken:~/mp_lab$ time ./gauss_pthread

size      = 2048x2048
maxnum    = 15
Init      = rand
Initializing matrix...done

time elapsed: 36.103560s
real    0m9.330s
user    0m35.066s
sys     0m1.040s
```

Figure 4: execution time of Gaussian elimination Pthread version

```
yeca16@kraken:~/omp_lab$ time ./gaussian_seq

size      = 2048x2048
maxnum    = 15
Init      = rand
Initializing matrix...done


real    1m8.915s
user    1m8.550s
sys     0m0.216s
yeca16@kraken:~/omp_lab$ time ./gaussian_seq

size      = 2048x2048
maxnum    = 15
Init      = rand
Initializing matrix...done


real    1m9.059s
user    1m8.632s
sys     0m0.220s
```

Figure 5: execution time of Gaussian elimination sequential version

Gaussian_seq	Gaussian_pthead	Speedup
68.987s	35.720s	1.931

Note: speedup= Sp(n) =T*(n) /Tp(n)= 68.987s/35.720s=1.931 [1].

Reference

1.Rauber, Thomas, and Gudula Rünger. *Parallel programming: For multicore and cluster systems*. Springer Science & Business Media, 2013.