ĐẠI HỌC QUỐC GIA TP.HCM

TRƯỜNG ĐẠI HỌC BÁCH KHOA

BÁO CÁO BÀI TẬP LỚN

MÔN

XỬ LÝ SONG SONG VÀ HỆ PHÂN BỐ

Đề bài:

Viết chương trình nhân hai ma trận cấp NxM chạy trên Intel Xeon Phi dùng mô hình Native hoặc Offload.

GV Hướng dẫn	SV Thực hiện	
Nguyễn Quang Hùng	Vũ Duy Quang	51303192
	Hoàng Thành Nhân	51302690

I. Chương trình

- Chương trình thực hiện bằng OpenMP, sử dụng mô hình Native, chạy trên Intel Xeon Phi
- Hệ thống sử dụng:
  - Model name: Intel® Xeon® CPU E5-2680 v3 @ 2.50GHz
  - Architecture: x86_64
  - CPU(s): 48
- Chương trình:
  - Kích thước 2 ma trận 2000x2000
  - 2 ma trận được biểu diễn dưới dạng mảng 1 chiều kích thước 2000x2000

```
#include <stdio.h>
#include <stdlib.h>
#include <omp.h>

void tran(double *A, double *B, int n) {
    int i,j;
    for(i=0; i<n; i++) {
        for(j=0; j<n; j++) {
            B[j*n+i] = A[i*n+j];
        }
    }
}

void mmT(double *A, double *B, double *C, int n)
{
    int i, j, k;
    double *B2;
    B2 = (double*)malloc(sizeof(double)*n*n);
    tran(B,B2, n);
    for (i = 0; i < n; i++) {
        for (j = 0; j < n; j++) {
            double dot  = 0;
            for (k = 0; k < n; k++) {
                dot += A[i*n+k]*B2[j*n+k];
            }
            C[i*n+j ] = dot;
        }
    }
    free(B2);
}

void mmT_omp(double *A, double *B, double *C, int n, int threads)
{
```

```
    double *B2;
    B2 = (double*)malloc(sizeof(double)*n*n);
    tran(B,B2, n);
    #pragma omp parallel num_threads(threads)
    {
        int i, j, k;
        #pragma omp for
        for (i = 0; i < n; i++) {
            for (j = 0; j < n; j++) {
                double dot  = 0;
                for (k = 0; k < n; k++) {
                    dot += A[i*n+k]*B2[j*n+k];
                }
                C[i*n+j ] = dot;
            }
        }

    }
    free(B2);
}

int main() {
    int i, n;
    double *A, *B, *C, dtime_seq, dtime_par;

    //Kích thước ma trận 2000x2000
    n=2000;

    A = (double*)malloc(sizeof(double)*n*n);
    B = (double*)malloc(sizeof(double)*n*n);
    C = (double*)malloc(sizeof(double)*n*n);

    //Thiết lập giá trị ban đầu cho 2 ma trận
    for(i=0; i<n*n; i++) {
        A[i] = 10;
        B[i] = 0.25;
    }

    //Nhân 2 ma trận tuần tự
    dtime_seq = omp_get_wtime();
    mmT(A,B,C, n);
    dtime_seq = omp_get_wtime() - dtime_seq;
    printf("Thoi gian thuc thi tuan tu: %f s\n\n", dtime_seq);

    //Nhân 2 ma trận song song với 2 thread
    dtime_par = omp_get_wtime();
    mmT_omp(A,B,C, n,2);
```

```
    dtime_par = omp_get_wtime() - dtime_par;
    printf("Thoi gian thuc thi song song voi %d threads: %f s\n", 2,
dtime_par);
    printf("Speedup : %f\n", dtime_seq/dtime_par);
    printf("Efficiency: %f\n\n", (dtime_seq/dtime_par)/2);

    //Nhân 2 ma trận song song với 4 thread
    dtime_par = omp_get_wtime();
    mmT_omp(A,B,C, n,4);
    dtime_par = omp_get_wtime() - dtime_par;
    printf("Thoi gian thuc thi song song voi %d threads: %f s\n", 4,
dtime_par);
    printf("Speedup : %f\n", dtime_seq/dtime_par);
    printf("Efficiency: %f\n\n", (dtime_seq/dtime_par)/4);

    //Nhân 2 ma trận song song với 6 thread
    dtime_par = omp_get_wtime();
    mmT_omp(A,B,C, n,6);
    dtime_par = omp_get_wtime() - dtime_par;
    printf("Thoi gian thuc thi song song voi %d threads: %f s\n", 6,
dtime_par);
    printf("Speedup : %f\n", dtime_seq/dtime_par);
    printf("Efficiency: %f\n\n", (dtime_seq/dtime_par)/6);

    //Nhân 2 ma trận song song với 8 thread
    dtime_par = omp_get_wtime();
    mmT_omp(A,B,C, n,8);
    dtime_par = omp_get_wtime() - dtime_par;
    printf("Thoi gian thuc thi song song voi %d threads: %f s\n", 8,
dtime_par);
    printf("Speedup : %f\n", dtime_seq/dtime_par);
    printf("Efficiency: %f\n\n", (dtime_seq/dtime_par)/8);

    //Nhân 2 ma trận song song với 10 thread
    dtime_par = omp_get_wtime();
    mmT_omp(A,B,C, n, 10);
    dtime_par = omp_get_wtime() - dtime_par;
    printf("Thoi gian thuc thi song song voi %d threads: %f s\n", 10,
dtime_par);
    printf("Speedup : %f\n", dtime_seq/dtime_par);
    printf("Efficiency: %f\n\n", (dtime_seq/dtime_par)/10);

    return 0;

}
```

II. Giải thuật tuần tự nhân hai ma trận

- Ý tưởng: Trước khi thực hiện nhân 2 ma trận A x B, ta thực hiện chuyển vị ma trận B trước, sau đó khi nhân 2 ma trận ta có thể truy xuất các phần tử trên cả hai ma trận theo hàng. Việc này sẽ làm giảm thời gian truy xuất.

```
void mmT(double *A, double *B, double *C, int n)
{
    int i, j, k;
    double *B2;
    B2 = (double*)malloc(sizeof(double)*n*n);
    tran(B,B2, n);
    for (i = 0; i < n; i++) {
        for (j = 0; j < n; j++) {
            double dot  = 0;
            for (k = 0; k < n; k++) {
                dot += A[i*n+k]*B2[j*n+k];
            }
            C[i*n+j ] = dot;
        }
    }
    free(B2);
}
```

III. Giải thuật song song nhân hai ma trận

```
void mmT_omp(double *A, double *B, double *C, int n, int threads)
{
    double *B2;
    B2 = (double*)malloc(sizeof(double)*n*n);
    tran(B,B2, n);
    #pragma omp parallel num_threads(threads)
    {
        int i, j, k;
        #pragma omp for
        for (i = 0; i < n; i++) {
            for (j = 0; j < n; j++) {
                double dot  = 0;
                for (k = 0; k < n; k++) {
                    dot += A[i*n+k]*B2[j*n+k];
                }
                C[i*n+j ] = dot;
            }
        }

    }
    free(B2);
}
```

IV. Khảo sát speedup và độ hiệu quả (efficient) của giải thuật song song

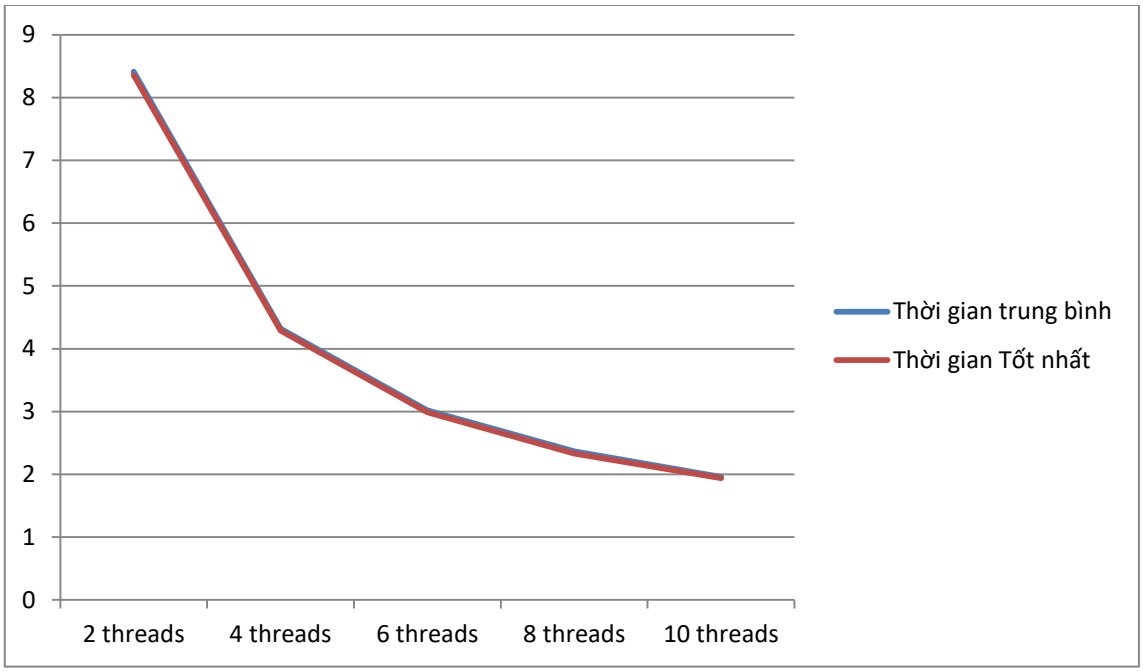
Bảng 1. Thời gian thực hiện giải thuật nhân 2 ma trận tuần tự (giây)

Lần	Thời gian
1	14.694810
2	14.563277
3	14.655281
4	14.689238
5	14.678332
TB	14.678332
Tốt Nhất	14.563277

Bảng 2. Thời gian thực hiện giải thuật nhân 2 ma trận song song (giây)

Lần	2 threads	4 threads	6 threads	8 threads	10 threads
1	8.356431	4.287107	2.987914	2.332173	1.942622
2	8.381213	4.319026	3.005273	2.343676	1.958057
3	8.430519	4.325084	3.002167	2.372348	1.958010
4	8.474687	4.335791	3.093113	2.444044	1.990546
5	8.403506	4.325510	3.016488	2.350119	1.939670
TB	8.409271	4.318504	3.020991	2.368472	1.957781
Tốt Nhất	8.356431	4.287107	2.987914	2.332173	1.939670

Đồ thị 1. Thời gian thực thi giải thuật nhân 2 ma trận song song



Bảng 3. Speadup và efficient của giải thuật nhân 2 ma trận song song (giây)

- Speedup: $S = \frac{T_{seq}}{T_{par}}$

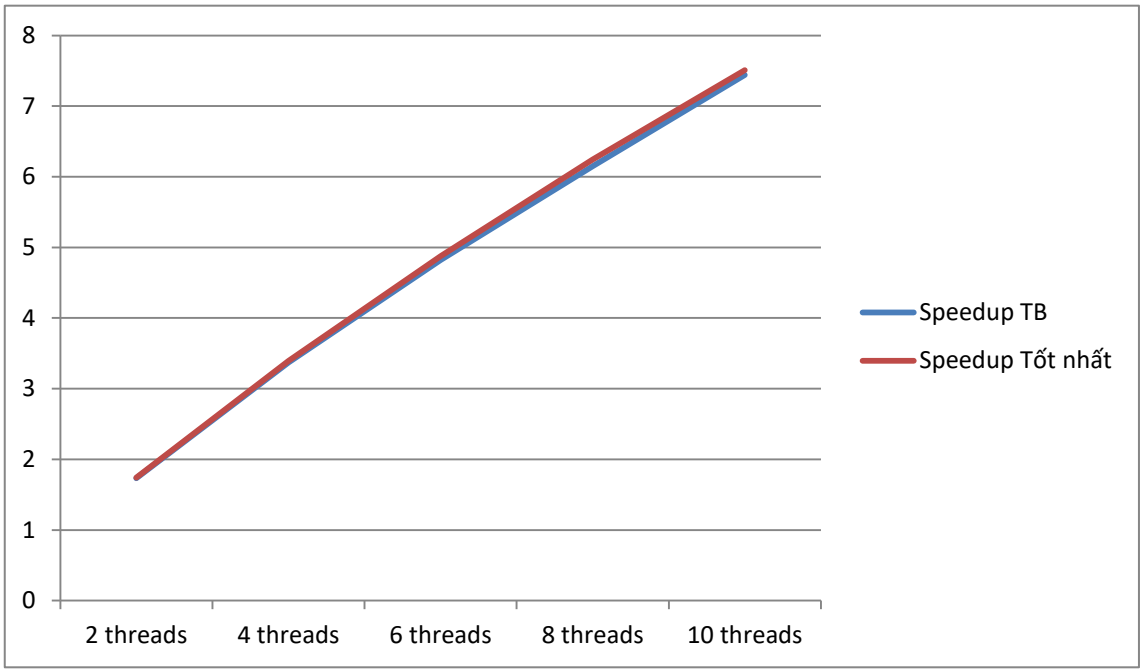
  T_{seq}: thời gian thực hiện giải thuật tuần tự

  T_{par}: thời gian thực hiện giải thuật song song
- Efficiency: E = $\frac{S}{N}$

  N: số thread

	2 threads	**4 threads**	**6 threads**	**8 threads**	**10 threads**
Speedup TB	1.731812027	3.372297061	4.820695262	6.14880691	7.438664999
Speedup Tốt nhất	1.742762790	3.396994057	4.874061636	6.244509734	7.508120969
Efficiency TB	0.865906013	0.843074265	0.803449210	0.768600864	0.743866500
Efficiency Tốt nhất	0.871381395	0.849248514	0.812343606	0.780563717	0.750812097

Đồ thị 2. Speedup trung bình và Speedup Tốt nhất



Đồ thị 2. Efficiency trung bình và Efficiency Tốt nhất