高性能计算实验报告

信息学部 2023311704 王昕远

实验六

1 实验环境

操作系统版本：Ubantu 22.04.3 LTS
CPU型号：AMD Ryzen 7 7745HX with Radeon Graphics 物理核数：8 频率：3600MHz

2 blas

```
void MY_MMult_blas(int m, int n, int k, double *a, int lda,
              double *b, int ldb,
              double *c, int ldc)
{
  cblas_dgemm(CblasRowMajor, CblasNoTrans, CblasNoTrans,
   m, n, k, 1.0, a, lda, b, ldb, 1.0, c, ldc);
}
```

3 pthread

```
#include <stdio.h>
#include <omp.h>
#include "defs.h"
#include <pthread.h>
#include <sys/time.h>

#define NUM_THREADS 4  // 定义线程数量

typedef struct {
    double *A;
    double *B;
    double *C;
    int m, n, k;
    int start_row;
    int end_row;
} thread_data_t;
```

```
// 多线程DGEMM计算部分
void *thread_dgemm(void *arg) {
    thread_data_t *data = (thread_data_t *)arg;
    double *A = data->A;
    double *B = data->B;
    double *C = data->C;
    int m = data->m;
    int n = data->n;
    int k = data->k;
    int start_row = data->start_row;
    int end_row = data->end_row;

    for (int i = start_row; i < end_row; i++) {
        for (int j = 0; j < n; j++) {
            double sum = 0.0;
            for (int p = 0; p < k; p++) {
                sum += A[i * k + p] * B[p * n + j];
            }
            C[i * n + j] = sum;
        }
    }

    pthread_exit(NULL);
}

void MY_MMult_pthread(int m, int n, int k, double *A, int lda, double *B, int ldb, double *C, int l
    pthread_t threads[NUM_THREADS];
    thread_data_t thread_data[NUM_THREADS];
    int rows_per_thread = m / NUM_THREADS;

    // 记录起始时间
    struct timeval start, end;
    gettimeofday(&start, NULL);

    // 分配每个线程负责的行
    for (int i = 0; i < NUM_THREADS; i++) {
        thread_data[i].A = A;
        thread_data[i].B = B;
        thread_data[i].C = C;
        thread_data[i].m = m;
        thread_data[i].n = n;
        thread_data[i].k = k;
        thread_data[i].start_row = i * rows_per_thread;
        thread_data[i].end_row = (i == NUM_THREADS - 1) ? m : (i + 1) * rows_per_thread;
```

```
        pthread_create(&threads[i], NULL, thread_dgemm, (void *)&thread_data[i]);
    }

    // 等待所有线程完成
    for (int i = 0; i < NUM_THREADS; i++) {
        pthread_join(threads[i], NULL);
    }

    // 记录结束时间
    gettimeofday(&end, NULL);

    // 计算时间差
    double time_taken = (end.tv_sec - start.tv_sec) * 1e6;
    time_taken = (time_taken + (end.tv_usec - start.tv_usec)) * 1e-6;

    // printf("MY_MMult (多线程) 完成, 耗时: %f 秒\n", time_taken);
}
```

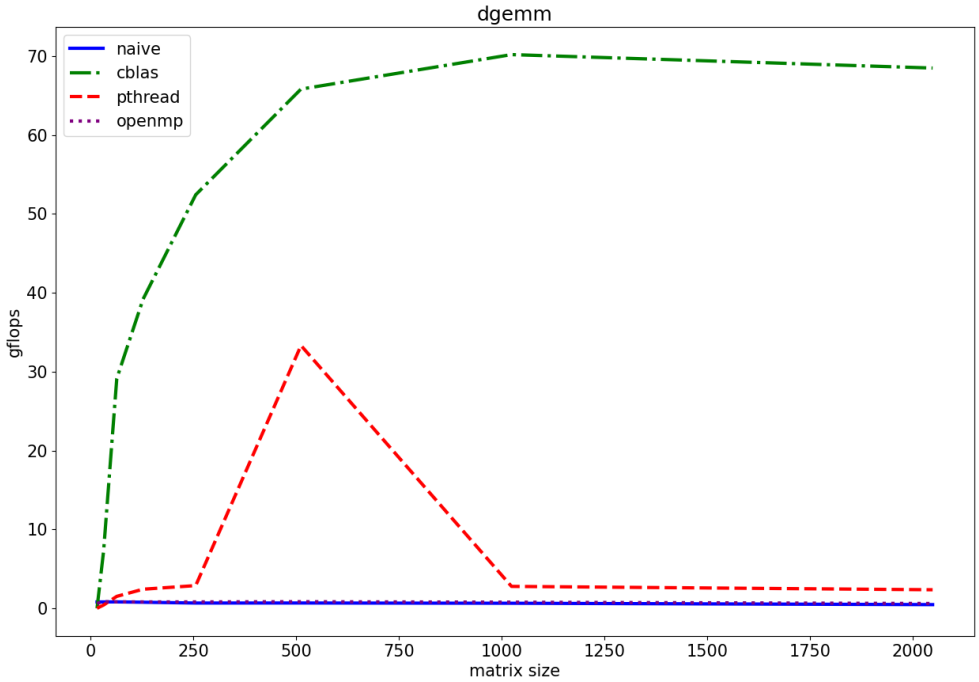
4 openMP

```
#include <stdio.h>
#include <omp.h>
#include "defs.h"

void MY_MMult_openmp(int m, int n, int k, double *A, int lda, double *B, int ldb, double *C, int ld
    int i, j, p;
    double sum;

    // Parallelized outer loops with OpenMP
    #pragma omp parallel for private(i, j, p, sum) shared(A, B, C)
    for (i = 0; i < m; i++) {
        for (j = 0; j < n; j++) {
            sum = 0.0;
            for (p = 0; p < k; p++) {
                sum += A[i * lda + p] * B[p * ldb + j];
            }
            C[i * ldc + j] = sum;
        }
    }
}
```

5 gflops曲线图


dgemm
naive
cblas
pthread
openmp
gflops
matrix size
0
10
20
30
40
50
60
70
0
250
500
750
1000
1250
1500
1750
2000