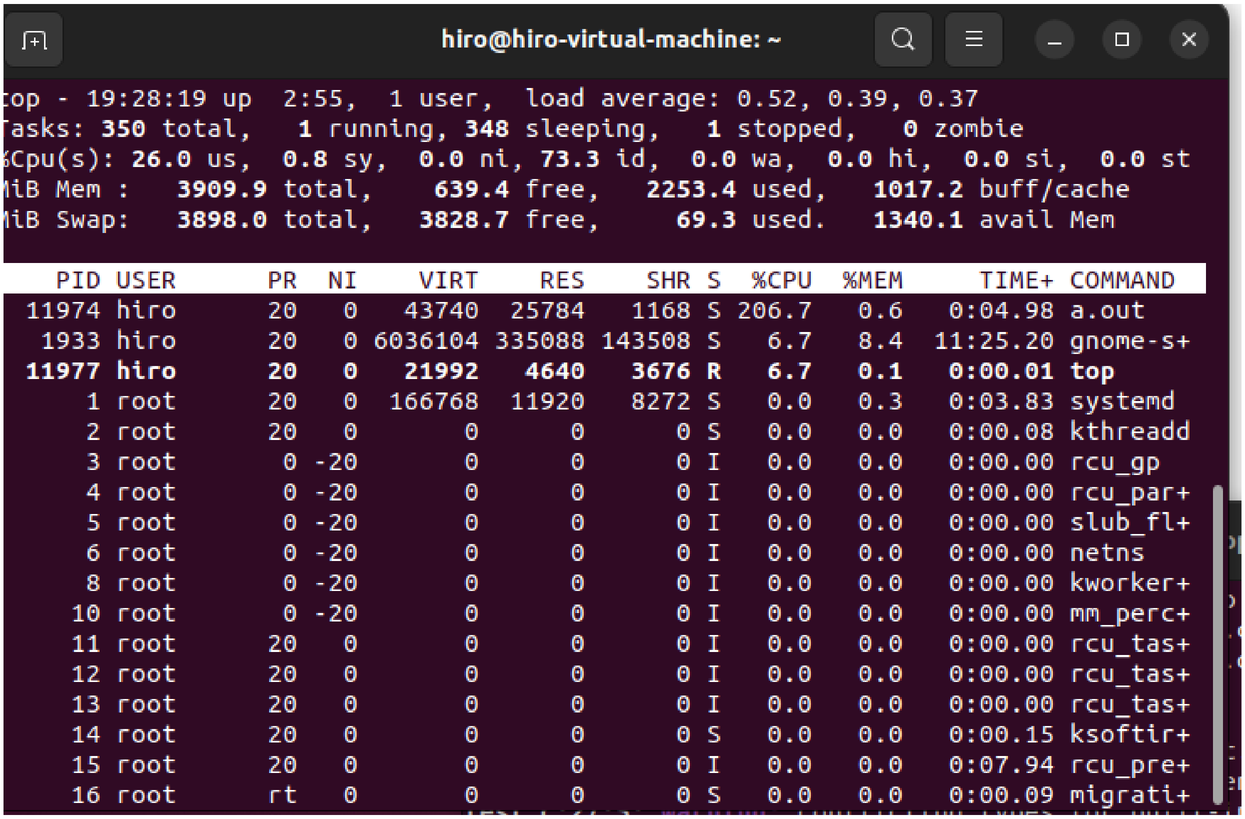
高性能计算实践-实验五

220110519 邢瑞龙 2023秋季

实验环境

1. OS：Linux Ubuntu 22.04
2. gcc: version 11.40(Ubuntu 11.40-1Ubuntu ~22.04)
3. CPU:11th Gen Intel(R) Core(TM) i7-1165G7 @2.80GHz cpu cores:2 (虚拟机)
4. 内存 3911MB

使用top查看cpu情况



```
op - 19:28:19 up  2:55,  1 user,  load average: 0.52, 0.39, 0.37
asks: 350 total,   1 running, 348 sleeping,   1 stopped,   0 zombie
Cpu(s): 26.0 us,  0.8 sy,  0.0 ni, 73.3 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
iB Mem :   3909.9 total,    639.4 free,   2253.4 used,   1017.2 buff/cache
iB Swap:   3898.0 total,   3828.7 free,     69.3 used.   1340.1 avail Mem

   PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
 11974 hiro      20   0   43740  25784   1168 S 206.7   0.6   0:04.98 a.out
  1933 hiro      20   0 6036104 335088 143508 S   6.7   8.4  11:25.20 gnome-s+
 11977 hiro      20   0   21992   4640   3676 R   6.7   0.1   0:00.01 top
     1 root      20   0  166768  11920   8272 S   0.0   0.3   0:03.83 systemd
     2 root      20   0       0      0      0 S   0.0   0.0   0:00.08 kthreadd
     3 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 rcu_gp
     4 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 rcu_par+
     5 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 slub_fl+
     6 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 netns
     8 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 kworker+
    10 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 mm_perc+
    11 root      20   0       0      0      0 I   0.0   0.0   0:00.00 rcu_tas+
    12 root      20   0       0      0      0 I   0.0   0.0   0:00.00 rcu_tas+
    13 root      20   0       0      0      0 I   0.0   0.0   0:00.00 rcu_tas+
    14 root      20   0       0      0      0 S   0.0   0.0   0:00.15 ksoftir+
    15 root      20   0       0      0      0 I   0.0   0.0   0:07.94 rcu_pre+
    16 root      rt   0       0      0      0 S   0.0   0.0   0:00.09 migrati+
```

实验方案

- 采用固定分块16x16的方式

双线程与naive对比

双线程代码：

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include "common.h"
#include <assert.h>
```

```c
#include <pthread.h>
#include <unistd.h>
#include <sys/time.h>

#define N 1024
#define M 16
#define NUM_THREADS 2

double A[N][N];
double B[N][N];
double C[N][N];

typedef struct {
    int n;
    double* a;
    double* b;
    double* c;
    int start_row;
    int end_row;
} myarg_t;

int fmin(int a, int b) {
    if (a < b) return a;
    else return b;
}

// Matrix multiplication function
void *dgemm(void *arg) {
    myarg_t *args = (myarg_t *)arg;
    int n = args->n;
    double *C = args->c;
    double *A = args->a;
    double *B = args->b;
    int start_row = args->start_row;
    int end_row = args->end_row;

    for (int ii = start_row; ii < end_row; ii += M) {
        for (int jj = 0; jj < n; jj += M) {
            for (int kk = 0; kk < n; kk += M) {
                for (int i = ii; i < fmin(ii + M, end_row); i++) {
                    for (int j = jj; j < fmin(jj + M, n); j++) {
                        for (int k = kk; k < fmin(kk + M, n); k++) {
                            C[i * n + j] += A[i * n + k] * B[k * n + j];
                        }
                    }
                }
            }
        }
    }
    return NULL;
}
```

```
int main() {
    // Initialize matrices A and B (for simplicity, assuming random values)
    for (int i = 0; i < N; i++) {
        for (int j = 0; j < N; j++) {
            A[i][j] = (double)rand() / RAND_MAX;
            B[i][j] = (double)rand() / RAND_MAX;
            C[i][j] = 0.0;
        }
    }

    myarg_t args[NUM_THREADS];
    pthread_t threads[NUM_THREADS];
    struct timeval start, finish;


    gettimeofday(&start, NULL);
    int chunk_size = N / NUM_THREADS;
    for (int i = 0; i < NUM_THREADS; i++) {
        args[i].n = N;
        args[i].a = (double *)A;
        args[i].b = (double *)B;
        args[i].c = (double *)C;
        args[i].start_row = i * chunk_size;
        args[i].end_row = (i == NUM_THREADS - 1) ? N : (i + 1) * chunk_size;

        int rc = pthread_create(&threads[i], NULL, dgemm, &args[i]);
        assert(rc == 0);
    }

    for (int i = 0; i < NUM_THREADS; i++) {
        int rc = pthread_join(threads[i], NULL);
        assert(rc == 0);
    }

    gettimeofday(&finish, NULL);
    double duration = ((double)(finish.tv_sec - start.tv_sec) * 1000000 + (double)(finish.tv_usec - start.tv_usec)) / 1000000;
    printf("Total time: %lf seconds\n", duration);

    return 0;
}
```

单线程代码:

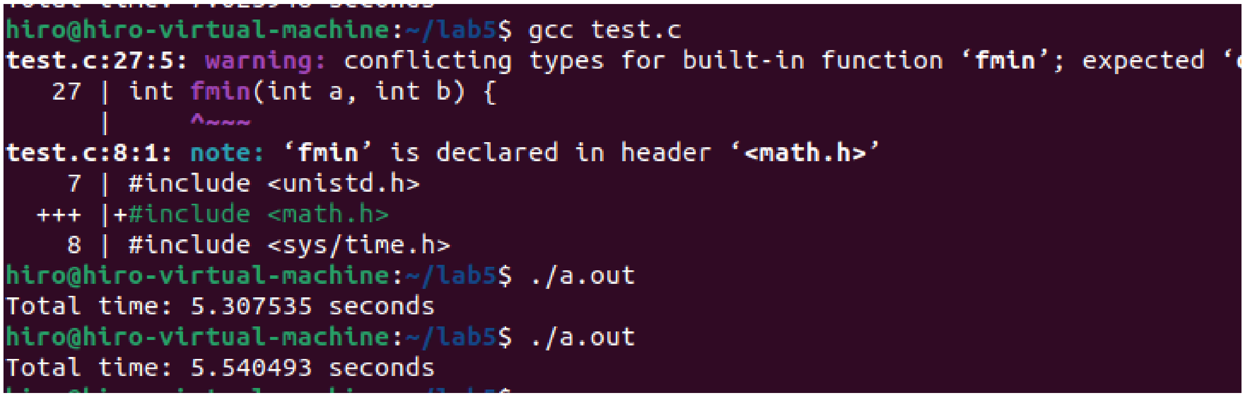
```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include "common.h"
#include <assert.h>
#include <pthread.h>
```

```
#include <unistd.h>
#include <sys/time.h>

#define N 1024
#define M 16

double A[N][N];
double B[N][N];
double C[N][N];

typedef struct {
    int n;
    double* a;
    double* b;
    double* c;
    int start_row;
    int end_row;
} myarg_t;

int fmin(int a, int b) {
    if (a < b) return a;
    else return b;
}

// Matrix multiplication function
void *dgemm(void *arg) {
    myarg_t *args = (myarg_t *)arg;
    int n = args->n;
    double *C = args->c;
    double *A = args->a;
    double *B = args->b;
    int start_row = args->start_row;
    int end_row = args->end_row;

    for (int ii = start_row; ii < end_row; ii += M) {
        for (int jj = 0; jj < n; jj += M) {
            for (int kk = 0; kk < n; kk += M) {
                for (int i = ii; i < fmin(ii + M, end_row); i++) {
                    for (int j = jj; j < fmin(jj + M, n); j++) {
                        for (int k = kk; k < fmin(kk + M, n); k++) {
                            C[i * n + j] += A[i * n + k] * B[k * n + j];
                        }
                    }
                }
            }
        }
    }
    return NULL;
}

int main() {
    // Initialize matrices A and B (for simplicity, assuming random values)
    for (int i = 0; i < N; i++) {
        for (int j = 0; j < N; j++) {
            A[i][j] = (double)rand() / RAND_MAX;
            B[i][j] = (double)rand() / RAND_MAX;
            C[i][j] = 0.0;
        }
    }

    myarg_t args;
    struct timeval start, finish;


    gettimeofday(&start, NULL);

    pthread_t p1;
    args.n = N;
        args.a = (double *)A;
        args.b = (double *)B;
        args.c = (double *)C;
        args.start_row = 0;
        args.end_row = N;
    int rc = pthread_create(&p1, NULL, dgemm, &args);
        assert(rc == 0);


    rc=pthread_join(p1, NULL);
    assert(rc==0);

    gettimeofday(&finish, NULL);
    double duration = ((double)(finish.tv_sec - start.tv_sec) * 1000000 + (double)(finish.tv_usec - start.tv_usec)) / 1000000;
    printf("Total time: %lf seconds\n", duration);

    return 0;
}
```

naive代码:

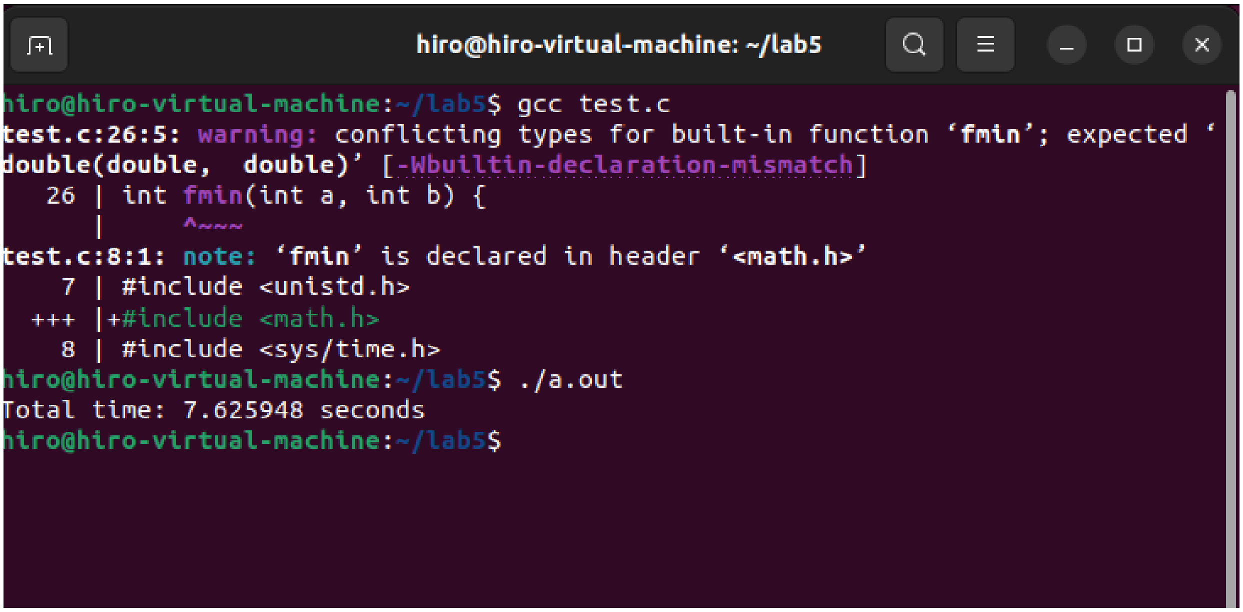
```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include "common.h"
#include <assert.h>
#include <pthread.h>
#include <unistd.h>
#include <sys/time.h>

#define N 1024
#define M 16

double A[N][N];
double B[N][N];
double C[N][N];

typedef struct {
    int n;
    double* a;
    double* b;
    double* c;
    int start_row;
    int end_row;
} myarg_t;

int fmin(int a, int b) {
    if (a < b) return a;
    else return b;
}

// Matrix multiplication function
void *dgemm(void *arg) {
    myarg_t *args = (myarg_t *)arg;
    int n = args->n;
    double *C = args->c;
    double *A = args->a;
    double *B = args->b;
    int start_row = args->start_row;
    int end_row = args->end_row;

    for(int i=0;i<n;i++)
        for(int j=0;j<n;j++)
            for(int k=0;k<n;k++)
            {
                C[i*n+j]+=A[i*n+k]*B[k*n+j];
            }
    return NULL;
}

int main() {
    // Initialize matrices A and B (for simplicity, assuming random values)
    for (int i = 0; i < N; i++) {
        for (int j = 0; j < N; j++) {
            A[i][j] = (double)rand() / RAND_MAX;
            B[i][j] = (double)rand() / RAND_MAX;
            C[i][j] = 0.0;
        }
    }

    myarg_t args;
    struct timeval start, finish;


    gettimeofday(&start, NULL);
    pthread_t p1;
    args.n = N;
        args.a = (double *)A;
        args.b = (double *)B;
        args.c = (double *)C;
        args.start_row = 0;
        args.end_row = N;
    int rc = pthread_create(&p1, NULL, dgemm, &args);
        assert(rc == 0);


    rc=pthread_join(p1, NULL);
    assert(rc==0);

    gettimeofday(&finish, NULL);
    double duration = ((double)(finish.tv_sec - start.tv_sec) * 1000000 + (double)
(finish.tv_usec - start.tv_usec)) / 1000000;
    printf("Total time: %lf seconds\n", duration);

    return 0;
}
```

双线程运行时间:



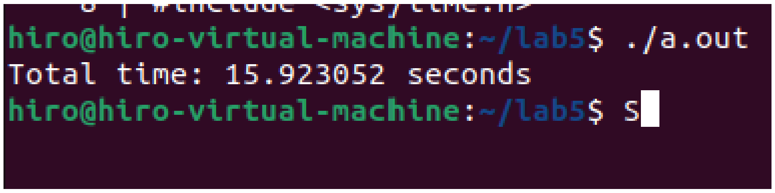
```
hiro@hiro-virtual-machine:~/lab5$ gcc test.c
test.c:27:5: warning: conflicting types for built-in function ‘fmin’; expected ‘
   27 | int fmin(int a, int b) {
      |     ^~~~
test.c:8:1: note: ‘fmin’ is declared in header ‘<math.h>’
    7 | #include <unistd.h>
  +++ |+#include <math.h>
    8 | #include <sys/time.h>
hiro@hiro-virtual-machine:~/lab5$ ./a.out
Total time: 5.307535 seconds
hiro@hiro-virtual-machine:~/lab5$ ./a.out
Total time: 5.540493 seconds
```

单线程运行时间:



```
hiro@hiro-virtual-machine:~/lab5$ gcc test.c
test.c:26:5: warning: conflicting types for built-in function ‘fmin’; expected ‘
double(double,  double)’ [-Wbuiltin-declaration-mismatch]
   26 | int fmin(int a, int b) {
      |     ^~~~
test.c:8:1: note: ‘fmin’ is declared in header ‘<math.h>’
    7 | #include <unistd.h>
  +++ |+#include <math.h>
    8 | #include <sys/time.h>
hiro@hiro-virtual-machine:~/lab5$ ./a.out
Total time: 7.625948 seconds
hiro@hiro-virtual-machine:~/lab5$
```

不分块naive版本运行时间：



```
    8 | #include <sys/time.h>
hiro@hiro-virtual-machine:~/lab5$ ./a.out
Total time: 15.923052 seconds
hiro@hiro-virtual-machine:~/lab5$ S
```

- 总结：运行速度:双线程分块>单线程分块>单线程不分块

遇到的问题

- 中间动态分配又遇到栈溢出的问题，采用全局变量解决
- 对分块矩阵运算过程不清楚，上网学习才了解