Lab6

实验环境

OS: Ubuntu 22.04.3 LTS 5.15.153.1-microsoft-standard-WSL2 GNU/Linux

gcc: version 11.4.0 (Ubuntu 11.4.0-1ubuntu1~22.04)

CPU: Intel(R) Core(TM) i5-6300U CPU @ 2.40GHz 2 Cores 4 Threads

	total	used	free	shared	buff/cache	available
Mem	3974020	688848	2106220	3224	1178952	3040812
Swap	1048576		1048576			

代码介绍

• naive_gemm: 经典之作,代码结构清晰,除了跑得慢以外没有缺点。伟大,无需多言。

```
▼ 核心代码

1 for (i = 0; i < m; i++) /* Loop over the rows of C */
2 {
3    for (j = 0; j < n; j++) /* Loop over the columns of C */
4    {
5       for (p = 0; p < k; p++)
6       { /* Update C( i,j ) with the inner product of the ith row of A and the jth column of B */
7       C(i, j) = C(i, j) + A(i, p) * B(p, j);
8       }
9    }
10 }
```

• cblas:调用 OpenBlas 库,很好 blackbox,使我运算飞快。

```
▼ 核心代码

1 cblas_dgemm(CblasColMajor, CblasNoTrans, CblasNoTrans,

2 m, n, k, alpha, a, lda, b, ldb, beta, c, ldc);
```

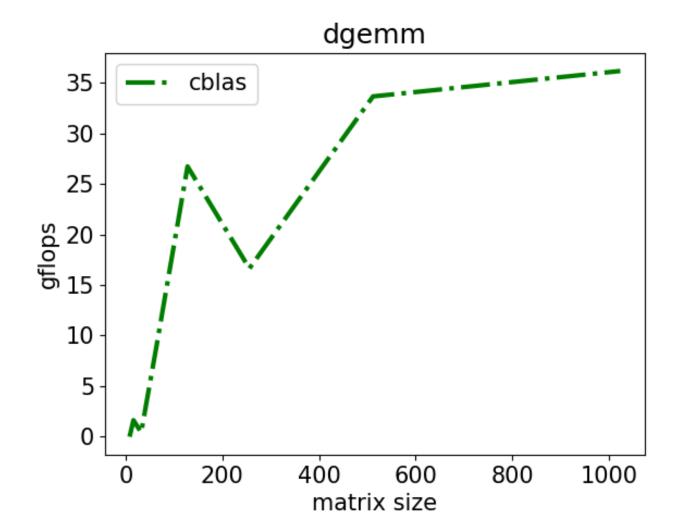
• pthread:调用 pthread 库,创建多线程并手动为每个线程分配任务。

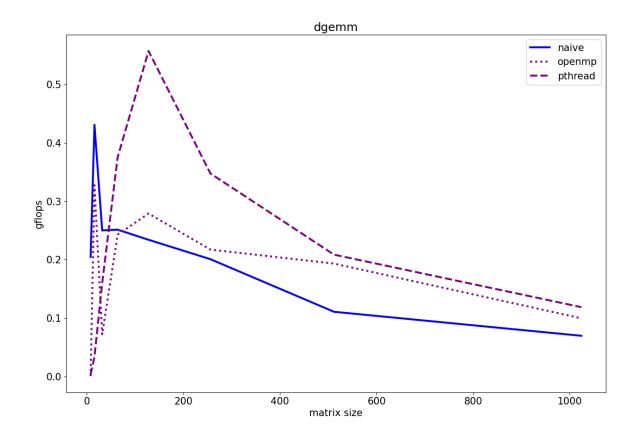
```
▼ 核心代码
 1 void MY MMult(int m, int n, int k,
 2
                 double *a, int lda,
 3
                 double *b, int ldb,
 4
                 double *c, int ldc)
 5 {
     pthread_t p[CPU_CORES];
 6
 7
 8
     int i, rc;
     for(i=0;i<CPU_CORES;i++)</pre>
 9
10
11
     myarg_t args = \{m, n, k,
12
                        a, lda,
                        b, ldb,
13
                        c, ldc,
14
15
                        i*m/CPU CORES,
                        (i+1)*m/CPU CORES};
16
17
       rc = pthread_create(&p[i], NULL, part, &args); assert(rc == 0);
18
19
     for(i=0;i<CPU_CORES;i++)</pre>
20
21
    rc = pthread_join(p[i], NULL); assert(rc == 0);
22
23 }
```

• openmp: 使用 OpenMP API 自动创建并行化线程。

```
▼ 核心代码
1 void MY MMult(int m, int n, int k,
2
                 double *a, int lda,
3
                 double *b, int ldb,
                 double *c, int ldc)
4
5 {
 6 int i;
    #pragma omp parallel for
8
    for(i=0;i<CPU CORES;i++)</pre>
 9
     {
10
      myarg t args = {m, n, k,
11
                       a, lda,
                        b, ldb,
12
13
                        c, ldc,
                        i*m/CPU_CORES,
14
15
                        (i+1)*m/CPU CORES };
16
       part(&args);
```

GFLOPS曲线图





运行截图

```
top - 16:04:04 up 25 min, 1 user, load average: 1.23, 0.75, 0.45
Tasks: 53 total, 2 running, 51 sleeping, 0 stopped, 0 zombie
%Cpu(s): 47.4 us, 1.2 sy, 0.0 ni, 44.3 id, 0.2 wa, 0.0 hi, 6.9 si, 0.0 st
                           2068.4 free,
           3880.9 total,
MiB Mem :
                                          1057.4 used,
                                                        755.1 buff/cache
MiB Swap:
                                                        2564.1 avail Mem
           1024.0 total,
                           1024.0 free,
                                             0.0 used.
   PID USER
                 PR NI
                           VIRT
                                   RES
                                         SHR S %CPU %MEM
                                                               TIME+ COMMAND
  11823 hanami
                          68656 43160
                                         1948 R 200.0
                                                             1:17.37 test MMult.x
   9386 hanami
                 20
                      0
                          21.4g 250084
                                       47852 S
                                                  2.7
                                                       6.3
                                                             0:26.78 node
     1 root
                      0 165984
                 20
                                11144
                                        8124 S
                                                  0.7
                                                       0.3
                                                             0:19.16 systemd
                                                       1.7
   9419 hanami
                 20
                      0 1184200
                                66848
                                        41040 S
                                                 0.7
                                                             0:03.35 node
    693 root
                 20
                          43388 36732
                                        10104 S
                                                       0.9
                                                             0:12.87 python3
                                                 0.3
```

```
hanami@Hanami:~$ pstree
systemd——2*[agetty]
         -cron
         dbus-daemon
                          -SessionLeader----Relay(9307)---sh----sh----sh----node
                                                                                    -12*[{node}]
                                                                              -node
                                                                                               -
-13*[{cpptools}]
                                                                                    -cpptools-
                                                                                     node---6*[{node}]
                                                                                    -bash---make
                                                                                                  -sh---test_MMult.x---3*[{test_MMult.x}]
                                                                                          —pstree
                                                                                     -bash-
                                                                                     -sh----cpuUsage.sh----sleep
                                                                                   -10*[{node}]
```

Lab3

多个c代码中有相同的MY_MMult函数,怎么判断可执行文件调用的是哪个版本的MY_MMult函数?是makefile中的哪行代码决定的?

```
1 $(LINKER) $(OBJS) $(LDFLAGS) -o $@
```

性能数据_data/output_MMult0.m是怎么生成的? c代码中只是将数据输出到终端并没有写入文件。

■ makefile中包含echo指令,将性能数据写入文件

-10*[{node}]

Lab5截图

```
top - 16:14:35 up 36 min, 1 user, load average: 0.88, 0.43, 0.37
Tasks: 55 total, 2 running, 53 sleeping, 0 stopped, 0 zombie
%Cpu(s): 48.3 us, 1.2 sy, 0.0 ni, 45.2 id, 0.1 wa, 0.0 hi, 5.3 si, 0.0 st MiB Mem : 3880.9 total, 1926.1 free, 1177.3 used, 777.4 buff/cache
                                                         2443.8 avail Mem
MiB Swap:
           1024.0 total, 1024.0 free,
                                             0.0 used.
   PID USER
                 PR NI VIRT
                                 RES SHR S %CPU %MEM
                                                                TIME+ COMMAND
  18089 hanami
                  20 0
                          60160 42928 1728 S 200.0 1.1 0:47.02 test MMult.x
   9386 hanami
                 20 0 21.5g 326272 47856 S 2.0 8.2 0:57.60 node
                  20 0 165984 11144
     1 root
                                        8124 S
                                                  1.0 0.3 0:26.11 systemd
                          43388 36732 10104 D 0.3
    693 root
                  20
                                                       0.9 0:16.33 python3
 cpptools-srv—7*[{cpptools-srv}]
_sh-_sh-_sh-_node-_node-_12*[{node}]
                      -node---cpptools---13*[{cpptools}]
                            -2*[node----6*[{node}]]
                           -node ___bash---make---sh---test_MMult.x---2*[{test_MMult.x}]
                            -bash---pstree
                           └<u>-12*[</u>{node}]
```