Lab6 实验报告

郑子炫 2023311724

1. 实验环境

- **操作系统版本**：Ubuntu 22.04.3 LTS
- **编译器版本**：gcc (Ubuntu 11.4.0-1ubuntu1~22.04) 11.4.0
- **CPU 信息**：
  - **物理核数**：10
  - **频率**：2.918398 GHz

2. 各种实现方式简介及核心代码

2.1 Naive 实现

Naive 实现是按照数学矩阵乘法公式计算。

```
void dgemm(int m, int n, int k, int beta,
           double A[][k], double B[][n], double C[][n]){
    for(int i=0; i< m;i ++){    //C[i]
        for(int j=0; j< n; j++){  //C[i][j]
            C[i][j] = beta*C[i][j];
            for(int p=0; p< k; p++){
                C[i][j] += A[i][p]*B[p][j];
            }
        }
    }
}
```

2.2 OpenBLAS 实现

OpenBLAS 提供了 cblas_dgemm 函数，用于计算矩阵乘法。

```
cblas_dgemm(CblasColMajor, CblasNoTrans, CblasNoTrans, M, N, K, alpha, A, lda, B, ldb, beta, C, ldc);
```

2.3 Pthreads 实现

Pthreads 利用多线程库实现并行计算，能够提高矩阵乘法的计算速度。由于 CPU 有 10 个物理核，故设置线程数为 10。

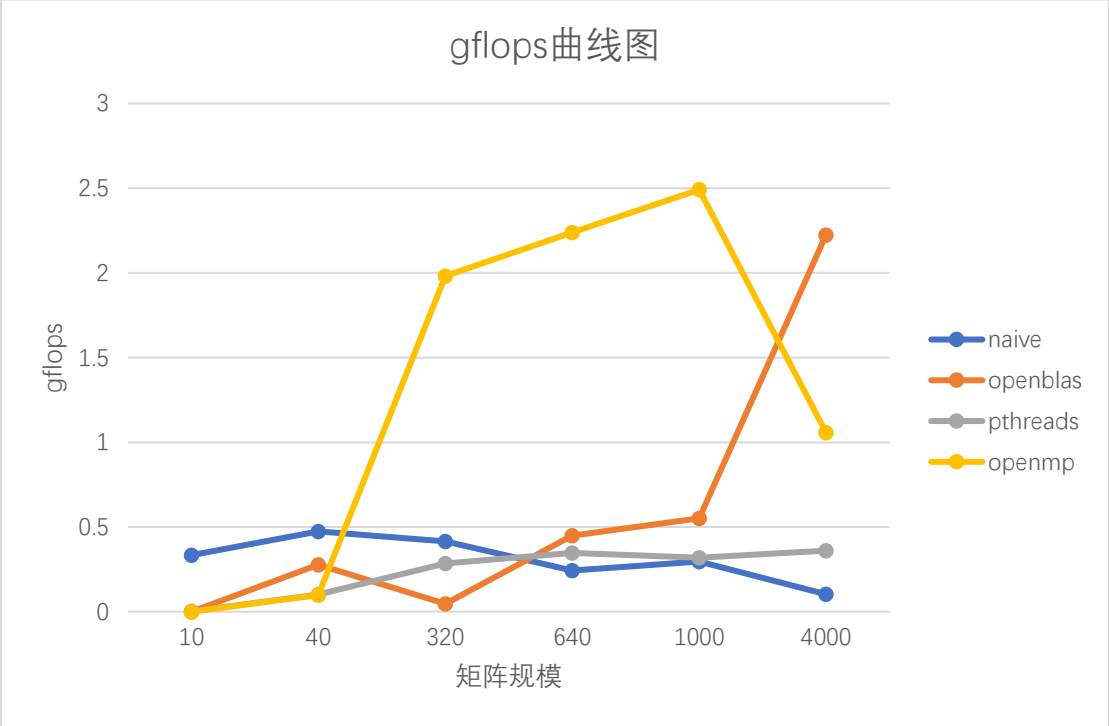
```
for (int i = tid * (n / NUM_THREADS); i < (tid + 1) * (n / NUM_THREADS); i += BLOCK_SIZE) {
    for (int j = 0; j < n; j += BLOCK_SIZE) {
        for (int k = 0; k < n; k += BLOCK_SIZE) {
            for (int ii = i; ii < i + BLOCK_SIZE && ii < n; ii++) {
                for (int jj = j; jj < j + BLOCK_SIZE && jj < n; jj++) {
                    double sum = 0;
                    for (int kk = k; kk < k + BLOCK_SIZE && kk < n; kk++) {
                        sum += A[ii][kk] * B[kk][jj];
                    }
                    C[ii][jj] += sum;
                }
            }
        }
    }
}
```

2.4 OpenMP 实现

OpenMP 是一种用于多平台共享内存多处理器编程的 API，可以通过简单的指令将代码并行化。

```
void dgemm(int m, int n, int k, int beta,
           double A[][k], double B[][n], double C[][n]){
    #pragma omp parallel for collapse(2)
    for(int i=0; i< m;i ++){
        for(int j=0; j< n; j++){
            C[i][j] = beta*C[i][j];
            for(int p=0; p< k; p++){
                C[i][j] += A[i][p]*B[p][j];
            }
        }
    }
}
```

3. gflops 曲线图



分析:

1. **Naive** 的 GFLOPS 在小规模矩阵（10 和 40）时表现相对较高，但随着规模的增加，GFLOPS 值急剧下降，尤其在 4000 规模时，几乎无法提供有效的性能。

2. **OpenBLAS** 在中小规模下表现良好，但在大规模（4000）时，其 GFLOPS 值显著提升，显示出该实现对大矩阵的优化能力。
3. **Pthreads** 和 **OpenMP** 在中等规模下（320 和 640）表现出一定的并发优势，但在更大规模（1000 和 4000）时，GFLOPS 值相对稳定，说明其性能在大规模计算中不如 OpenBLAS。

4. 运行截图

```
top - 21:15:31 up  2:11,  1 user,  load average: 6.06, 1.77, 1.33
Tasks:  62 total,   2 running,  60 sleeping,   0 stopped,   0 zombie
%Cpu0  :  0.9 us,  0.3 sy,  0.0 ni, 88.8 id,  0.0 wa,  0.0 hi, 10.0 si,  0.0 st
%Cpu1  :100.0 us,  0.0 sy,  0.0 ni,  0.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu2  :  0.3 us,  0.7 sy,  0.0 ni, 99.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu3  :100.0 us,  0.0 sy,  0.0 ni,  0.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu4  :  0.0 us,  1.7 sy,  0.0 ni, 97.7 id,  0.7 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu5  :100.0 us,  0.0 sy,  0.0 ni,  0.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu6  :100.0 us,  0.0 sy,  0.0 ni,  0.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu7  :  1.0 us,  1.0 sy,  0.0 ni, 98.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu8  :  0.0 us,  0.3 sy,  0.0 ni, 99.7 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu9  :100.0 us,  0.0 sy,  0.0 ni,  0.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu10 :  0.3 us,  1.3 sy,  0.0 ni, 98.3 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu11 :100.0 us,  0.0 sy,  0.0 ni,  0.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu12 :100.0 us,  0.0 sy,  0.0 ni,  0.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu13 :  1.7 us,  1.3 sy,  0.0 ni, 97.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu14 :100.0 us,  0.0 sy,  0.0 ni,  0.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu15 :  0.7 us,  0.3 sy,  0.0 ni, 99.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu16 :100.0 us,  0.0 sy,  0.0 ni,  0.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu17 :  0.0 us,  1.0 sy,  0.0 ni, 99.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
```

Lab3 的问题：

1)：在 Makefile 中，通常在链接阶段，

```
gcc -o my_program file1.o file2.o file3.o
```

这里的 file1-file3 的顺序决定了最后的链接版本。

2）在运行程序时，可能在命令行中使用了输出重定向，将终端输出重定向到文件。

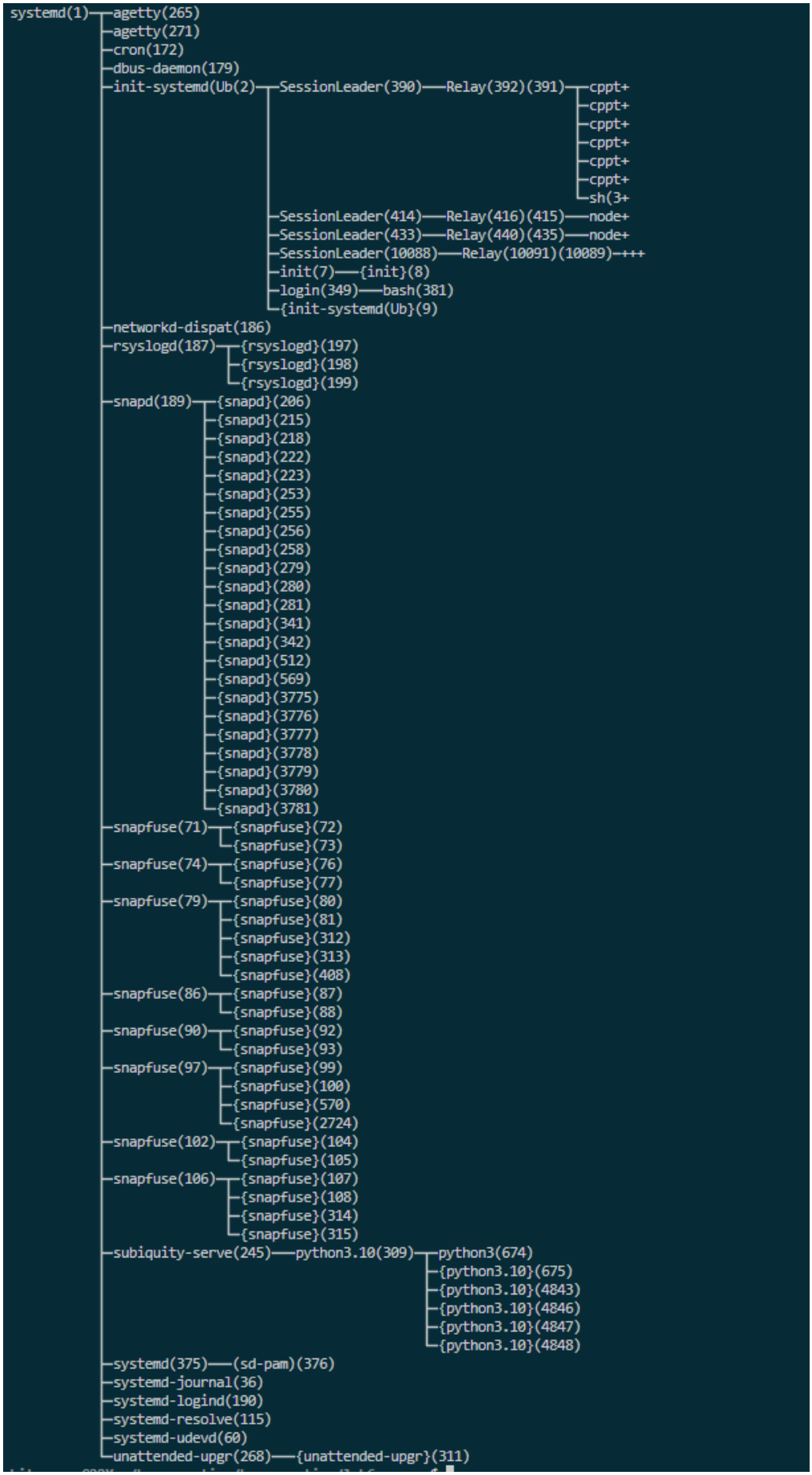
Lab5 的问题：

top

```
top - 21:26:00 up  2:22,  1 user,  load average: 1.23, 4.24, 3.70
Tasks:  59 total,   1 running,  58 sleeping,   0 stopped,   0 zombie
%Cpu(s): 50.5 us,  0.5 sy,  0.0 ni, 48.3 id,  0.0 wa,  0.0 hi,  0.6 si,  0.0 st
MiB Mem :   7806.4 total,   5992.4 free,   1475.5 used,    338.5 buff/cache
MiB Swap:   2048.0 total,   2048.0 free,      0.0 used.   6096.8 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
  62153 hitsz_z+  20   0  459748 376824   1868 S  1003   4.7   0:54.77 pthreads_+
    402 hitsz_z+  20   0 1333980 110888  46188 S  10.0   1.4   0:25.36 node
    483 hitsz_z+  20   0   21.4g 259740  48916 S   4.0   3.2   5:49.66 node
    557 hitsz_z+  20   0 1263664  72896  41884 S   2.0   0.9   0:19.44 node
      1 root      20   0  165852  11248   8308 S   1.7   0.1   1:53.08 systemd
    674 root      20   0   43372  37400  10412 S   1.0   0.5   0:58.42 python3
    416 hitsz_z+  20   0  996884  55024  38204 S   0.7   0.7   0:04.55 node
    309 root      20   0  803772  80880  23064 S   0.3   1.0   0:12.80 python3.10
      2 root      20   0    2476   1436   1320 S   0.0   0.0   0:00.00 init-syst+
      7 root      20   0    2504    144    132 S   0.0   0.0   0:00.00 init
     36 root      19  -1   47752  14524  13500 S   0.0   0.2   0:00.32 systemd-j+
     60 root      20   0   22096   5944   4448 S   0.0   0.1   0:00.82 systemd-u+
     71 root      20   0  152992    180     20 S   0.0   0.0   0:00.00 snapfuse
     74 root      20   0  152992   2228     32 S   0.0   0.0   0:00.00 snapfuse
     79 root      20   0  377284  13648    272 S   0.0   0.2   0:01.18 snapfuse
     86 root      20   0  153124    172      8 S   0.0   0.0   0:00.00 snapfuse
     90 root      20   0  152992    180     20 S   0.0   0.0   0:00.00 snapfuse
     97 root      20   0  302520   9208    356 S   0.0   0.1   0:00.16 snapfuse
    102 root      20   0  152992    180     16 S   0.0   0.0   0:00.00 snapfuse
    106 root      20   0  302520  13476    316 S   0.0   0.2   0:01.53 snapfuse
    115 systemd+  20   0   25540  12428   8236 S   0.0   0.2   0:00.20 systemd-r+
    172 root      20   0    4308   2752   2516 S   0.0   0.0   0:00.03 cron
```

pstree


systemd(1)
agetty(265)
agetty(271)
cron(172)
dbus-daemon(179)
init-systemd(Ub(2)
SessionLeader(390)
Relay(392)(391)
cppt+
cppt+
cppt+
cppt+
cppt+
cppt+
sh(3+
SessionLeader(414)
Relay(416)(415)
node+
SessionLeader(433)
Relay(440)(435)
node+
SessionLeader(10088)
Relay(10091)(10089)
+++
init(7)
{init}(8)
login(349)
bash(381)
{init-systemd(Ub}(9)
networkd-dispat(186)
rsyslogd(187)
{rsyslogd}(197)
{rsyslogd}(198)
{rsyslogd}(199)
snapd(189)
{snapd}(206)
{snapd}(215)
{snapd}(218)
{snapd}(222)
{snapd}(223)
{snapd}(253)
{snapd}(255)
{snapd}(256)
{snapd}(258)
{snapd}(279)
{snapd}(280)
{snapd}(281)
{snapd}(341)
{snapd}(342)
{snapd}(512)
{snapd}(569)
{snapd}(3775)
{snapd}(3776)
{snapd}(3777)
{snapd}(3778)
{snapd}(3779)
{snapd}(3780)
{snapd}(3781)
snapfuse(71)
{snapfuse}(72)
{snapfuse}(73)
snapfuse(74)
{snapfuse}(76)
{snapfuse}(77)
snapfuse(79)
{snapfuse}(80)
{snapfuse}(81)
{snapfuse}(312)
{snapfuse}(313)
{snapfuse}(408)
snapfuse(86)
{snapfuse}(87)
{snapfuse}(88)
snapfuse(90)
{snapfuse}(92)
{snapfuse}(93)
snapfuse(97)
{snapfuse}(99)
{snapfuse}(100)
{snapfuse}(570)
{snapfuse}(2724)
snapfuse(102)
{snapfuse}(104)
{snapfuse}(105)
snapfuse(106)
{snapfuse}(107)
{snapfuse}(108)
{snapfuse}(314)
{snapfuse}(315)
subiquity-serve(245)
python3.10(309)
python3(674)
{python3.10}(675)
{python3.10}(4843)
{python3.10}(4846)
{python3.10}(4847)
{python3.10}(4848)
systemd(375)
(sd-pam)(376)
systemd-journal(36)
systemd-logind(190)
systemd-resolve(115)
systemd-udevd(60)
unattended-upgr(268)
{unattended-upgr}(311)