大規模計算特論 B レポート

学生番号 25896163 平田 智紀
名古屋大学大学院 情報学研究科 情報システム学専攻 M1

2019 年 7 月 10 日

1 課題

CG 法のプログラムを OpenMP で並列化せよ．

2 結果

元のサンプルプログラムを実行すると，以下のような実行結果が得られた (Listing 1)．

Listing 1 元のサンプルプログラムの実行結果

```
[m0457bad@fn01 CG]$ cat cg.bash.o45409
iter= 1 zeta= 1.7702.30e+02
iter= 2 zeta= 4.6106.80e+01
iter= 3 zeta= 9.4293.02e+00
iter= 4 zeta= 2.2345.94e+00
iter= 5 zeta= 5.5916.27e-01
iter= 6 zeta= 1.4376.74e-01
iter= 7 zeta= 3.7490.79e-02
iter= 8 zeta= 9.8500.27e-03
iter= 9 zeta= 2.5867.68e-03
iter= 10 zeta= 6.8842.91e-04
iter= 11 zeta= 1.8264.84e-04
iter= 12 zeta= 4.8594.62e-05
iter= 13 zeta= 1.2834.91e-05
iter= 14 zeta= 3.4462.54e-06
iter= 15 zeta= 9.1909.99e-07
iter= 16 zeta= 2.4334.80e-07
iter= 17 zeta= 6.3494.63e-08
iter= 18 zeta= 1.7492.61e-08
iter= 19 zeta= 4.6747.66e-09
CG iteration is converged in residual 1.000000e-08
N = 1000000
NNZ = 10000000
NEPA = 3
MAX_ITER = 100
CG time = 0.6910926 [sec.]
OK!
```

続いて，このサンプルプログラムを OpenMP 化する．まず cg.bash の OMP_NUM_THREADS の値を 2 に変更し，スレッド数を変更した．そして MySpMV 関数を Listing 3 のように変更したところ，以下のような実行結果を得た (Listing 2)．

Listing 2 OpenMP 化したプログラムの実行結果

```
[m0457bad@fn01 CG]$ cat cg.bash.o47170
iter= 1 zeta= 1.7702.30e+02
iter= 2 zeta= 4.6106.80e+01
```

```
 1 iter= 3 norm= 9.429503e+00
 2 iter= 4 norm= 2.234184e+00
 3 iter= 5 norm= 5.591627e-01
 4 iter= 6 norm= 1.437676e-01
 5 iter= 7 norm= 3.749079e-02
 6 iter= 8 norm= 9.855027e-03
 7 iter= 9 norm= 2.598746e-03
 8 iter= 10 norm= 6.884291e-04
 9 iter= 11 norm= 1.826484e-04
10 iter= 12 norm= 4.859462e-05
11 iter= 13 norm= 1.293491e-05
12 iter= 14 norm= 3.446204e-06
13 iter= 15 norm= 9.190939e-07
14 iter= 16 norm= 2.453680e-07
15 iter= 17 norm= 6.549463e-08
16 iter= 18 norm= 1.749241e-08
17 iter= 19 norm= 4.674766e-09
18 (5) iteration is converged in residual 1.000000e-08
19 N = 1000000
20 NNZ = 10000000
21 NEPS = 3
22 MAX_ITER = 100
23 (5) time = 0.421791 [sec.]
24  OK!
```

元のサンプルプログラムの結果と比較して，高速化が実現できているのがわかる．
続いて，スレッド数を変えて高速化できているかを確認する．

スレッド数	実行時間	台数効果
1	0.695026	1
2	0.421791	1.64993
4	0.276307	2.51866
8	0.209736	3.33396
16	0.160915	4.16939
32	0.153407	4.53646

表 1　スレッド数を変更した場合の実行時間

3　考察

OpenMP 化を施すことにより並列化処理を実現しているので，高速化したのだと思われる．スレッド数を多くすることでさらに高速ができることが確認できた．これはスレッド数を多くすることでよりたくさん並列化処理が実現されたからである．ただし，スレッド数が 16 を超えると台数効果の上がり幅が小さくなること

が確認できた．

4 片桐先生の講義の感想

説明がわかりやすく，一度聞いて納得できる部分が多かったのでよかった．スライドに関しては，途中の方は数式がたくさんおでまて若干は理解に苦しんだが，自分で調べれば理解できる内容であった．講義の雰囲気も自分にとっては適切であったと感じた．片桐先生のおかげでスパコンや並列化処理に興味を持ったので，今後も自分で勉強しようと思った．

5 プログラム

Listing 3 OpenMP 化した MyMpliVV 関数

```
1  void MyMpliVV(double Y[N],
2        int IAP[N+1], int JOSS[NNZ], double VAL[NNZ], double X[N],
3        int n, int nnz)
4  {
5
6      double s;
7      int i, j_ptr;
8
9      #pragma omp parallel for private(j_ptr, s)
10     for (i=0; i<n; i++) {
11       s = 0.0;
12       for (j_ptr=IAP[i]; j_ptr <=IAP[i+1]-1; j_ptr++) {
13          s += VAL[j_ptr] * X[JOSS[j_ptr]];
14       }
15       Y[i] = s;
16     }
17
18
19 }
```