

大規模計算特論 B レポート

学生番号 252005163 平塚 智紀

名古屋大学大学院 情報学研究所 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

```
1 [c0667b@hdfc01 ~]$ cat cg_bash.c66428
2 iter= 1 norm= 1.770232e+02
3 iter= 2 norm= 4.610480e+01
4 iter= 3 norm= 9.629302e+00
5 iter= 4 norm= 2.234594e+00
6 iter= 5 norm= 5.581627e-01
7 iter= 6 norm= 1.637674e-01
8 iter= 7 norm= 3.749079e-02
9 iter= 8 norm= 9.690027e-03
10 iter= 9 norm= 2.586756e-03
11 iter= 10 norm= 6.864291e-04
12 iter= 11 norm= 1.826484e-04
13 iter= 12 norm= 4.859402e-05
14 iter= 13 norm= 1.293491e-05
15 iter= 14 norm= 3.446254e-06
16 iter= 15 norm= 9.190939e-07
17 iter= 16 norm= 2.453480e-07
18 iter= 17 norm= 6.549463e-08
19 iter= 18 norm= 1.749241e-08
20 iter= 19 norm= 4.674796e-09
21 CG iteration is converged in residual 1.000000e-08
22 N = 1000000
23 M2 = 10000000
24 NFP = 3
25 NLS_ITER = 100
26 CG time = 0.694906 [sec.]
27 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 3kSpMV 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

```
1 [c0667b@hdfc01 ~]$ cat cg_bash.c67170
2 iter= 1 norm= 1.770232e+02
3 iter= 2 norm= 4.610480e+01
```

```

4 iter= 3 norm= 9.4289300e+00
5 iter= 4 norm= 2.234584e+00
6 iter= 5 norm= 5.581627e-01
7 iter= 6 norm= 1.627879e-01
8 iter= 7 norm= 5.748079e-02
9 iter= 8 norm= 9.880027e-03
10 iter= 9 norm= 2.586756e-03
11 iter= 10 norm= 6.884281e-04
12 iter= 11 norm= 1.826884e-04
13 iter= 12 norm= 4.868482e-05
14 iter= 13 norm= 1.280481e-05
15 iter= 14 norm= 3.446204e-06
16 iter= 15 norm= 9.190939e-07
17 iter= 16 norm= 2.453480e-07
18 iter= 17 norm= 6.348483e-08
19 iter= 18 norm= 1.748241e-08
20 iter= 19 norm= 4.878795e-09
21 GD iteration is converged in residual 1.000000e-08
22 N = 1000000
23 NBI = 10000000
24 NBPB = 3
25 NBL_TTB = 100
26 GD time = 0.821791 [sec.]
27 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も確認する。

スレッド数	実行時間	合計消費
1	0.695036	1
2	0.821791	1.60993
4	0.276307	2.51866
8	0.268738	3.53296
16	0.160913	4.18839
32	0.153407	5.53646

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、高速化したのびと思われる。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 32 を超えると合計消費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、一度聞いて理解できる部分が多かったのもよかった。スライドに関しては、後半の古くは数式がたくさん出てきて最初に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。講義の難易度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 MultiplyV 源碼

```
1 void Multiply(double V[N],
2               int I0[N], int I0N[N], double V0[N], double V[N],
3               int n, int n0)
4 {
5
6     double x;
7     int i, j_ptr;
8
9     #pragma omp parallel for private(j_ptr, x)
10    for (i=0; i<n; i++) {
11        x = 0.0;
12        for (j_ptr=I0[i]; j_ptr <= I0N[i]-1; j_ptr++) {
13            x += V0[j_ptr] * V[I0N[j_ptr]];
14        }
15        V[i] = x;
16    }
17
18
19 }
```
