情報科学プロジェクト実験レポート課題

S142063 佐藤涼亮

平成 29 年 1 月 22 日

スレッドプログラミング

1　課題の内容

C++のスレッドプログラミングを用いた円周率の計算

1.1　要点

1. マルチスレッドプログラミングによる並列処理を行う
2. スレッド数は CPU 数に応じて設定する

2　プログラムの説明

逐次処理で計算していたプログラムを、マルチスレッドプログラムに変更し並列処理を行い、実行時間の短縮を図る。円周率の近似計算には、x を [0,1] の範囲で曲線 4/(1+x*x) の下側の面積を数値積分することで求める。

並列処理には、std::thread を使う場合と std::async() を使う場合を作成し、積分を分割し計算を行う。また、積分範囲を分割する intervals 変数の値を 40000000 と INT_MAX の 2 種類それぞれプログラムを作成した。スレッドの数は CPU の数に合わせる。

2.1　目的

マルチスレッドプログラムのプログラム実行の性能を調査する

2.2　方法

シングルスレッドプログラムをマルチスレッドプログラムに変更し、実行時間を比較する

2.3 結果

pi はシングルスレッド、thread は std::thread を用いた、future は std::async() を用いたプログラムになっている。

```
ryousuke@HP-Spectre:~/project/14thread$ time ./pi
Estimation of pi is 3.1415926536

real  0m0.710s
user  0m0.708s
sys 0m0.000s
ryousuke@HP-Spectre:~/project/14thread$ time ./thread
Estimation of pi is 3.1415926536

real  0m0.335s
user  0m1.216s
sys 0m0.000s
ryousuke@HP-Spectre:~/project/14thread$ time ./future
Estimation of pi is 3.1415926536

real  0m0.232s
user  0m0.812s
sys 0m0.000s
ryousuke@HP-Spectre:~/project/14thread$ time ./piIntMax
Estimation of pi is 3.1415926536

real  0m35.986s
user  0m35.984s
sys 0m0.000s
ryousuke@HP-Spectre:~/project/14thread$ time ./threadIntMax
Estimation of pi is 3.1415926536

real  0m15.653s
user  1m0.936s
sys 0m0.008s
ryousuke@HP-Spectre:~/project/14thread$ time ./futureIntMax
Estimation of pi is 3.1415926536

real  0m10.845s
user  0m41.828s
sys 0m0.008s
ryousuke@HP-Spectre:~/project/14thread$
```

CPU コア 4 個の環境で実行し、時間測定には time コマンドを用いた。

プログラム	結果	経過時間	CPU 時間
pi	3.1415926536	0m0.710s	0m0.708s
thread	3.1415926536	0m0.335s	0m1.216s
future	3.1415926536	0m0.232s	0m0.812s

プログラム	結果	経過時間	CPU 時間
piIntMax	3.1415926536	0m35.986s	0m35.984s
threadIntMax	3.1415926536	0m15.653s	1m0.936s
futureIntMax	3.1415926536	0m10,845s	0m41.828s

2.4 考察

結果を見ると、シングルスレッドのプログラムに対して、std::thread の場合は約 2.2 倍、std::async() の場合は約 3.2 倍、処理効率が向上したと言える。

3 感想

今回の実験において、マルチスレッド化することで効率は上がった。結果の user 時間を CPU の数で割ると real 時間に近似していることから、しっかり CPU の個数分スレッドが生成され、並列処理していることがわかる。今まで作成していたプログラムも並列化できるのではないかと、少し楽しくなってきた。もっと、計算に時間を要するプログラムをより多くの CPU を持ったコンピューターやサーバー、他にも MPI の並列処理において PC クラスタを構築し、短い時間で実行できるようにしてみたいと思った。

4 プログラム

ソースコード 1: pi.cpp

```
1  #include <iostream>
2  #include <iomanip>
3  const int intervals = 40000000;
4  const double width = 1.0 / intervals;
5  int main(int argc, char *argv[] )
6  {
7          double sum = 0.0;
8          for (int i = 0; i < intervals; i++ ) {
9                  const double x = (i + 0.5)*width;
10                 sum += 1/(1.0 + x*x);
11         }
12         sum *= 4.0*width;
13         std::cout << "Estimation␣of␣pi␣is␣"
14         << std::fixed << std::setprecision(10)
15         << sum << "\n";
16         return 0;
17 }
```

ソースコード 2: piIntMax.cpp

```
#include <iostream>
#include <iomanip>
#include <climits>
const int intervals = INT_MAX;
const double width = 1.0 / intervals;
int main(int argc, char *argv[] )
{
        double sum = 0.0;
        for (int i = 0; i < intervals; i++ ) {
                const double x = (i + 0.5)*width;
                sum += 1/(1.0 + x*x);
        }
        sum *= 4.0*width;
        std::cout << "Estimation of pi is "
        << std::fixed << std::setprecision(10)
        << sum << "\n";
        return 0;
}
```

ソースコード 3: thread.cpp

```
#include <iostream>
#include <iomanip>
#include <thread>
#include <vector>
#include <climits>
using namespace std;

const int intervals = 40000000;
const double width = 1.0 / intervals;

double func(double head,double tail,double &sum) {
        sum = 0.0;
        for (int i = head; i < tail; i++ ) {
                const double x = (i + 0.5)*width;
                sum += 1/(1.0 + x*x);
        }
}

int main()
{
        const int num = thread::hardware_concurrency();
        vector<thread> a;
        vector<double> total(num);
        for (int i=0; i<num; i++){
                double tmp = i*(intervals/num);
                a.push_back(thread(func, tmp, tmp+(intervals/num),ref(total[i])));
        }
        double sum = 0.0;
        for(auto& x : a) x.join();
        for (double &s : total) sum += s;
        sum *= 4.0*width;
        cout << "Estimation of pi is "
             << fixed << setprecision(10)
             << sum << "\n";
        return 0;
}
```

ソースコード 4: threadIntMax.cpp

```
#include <iostream>
#include <iomanip>
#include <thread>
#include <vector>
#include <climits>
using namespace std;
```

```
7
8  const int intervals = INT_MAX;
9  const double width = 1.0 / intervals;
10
11 double func(double head,double tail,double &sum) {
12         sum = 0.0;
13         for (int i = head; i < tail; i++ ) {
14                 const double x = (i + 0.5)*width;
15                 sum += 1/(1.0 + x*x);
16         }
17 }
18
19 int main()
20 {
21         const int num = thread::hardware_concurrency();
22         vector<thread> a;
23         vector<double> total(num);
24         double head = 0;
25         for (int i=0; i<num; i++){
26                 double tail = head + (intervals/num);
27                 if(i==num-1)tail=intervals;
28                 a.push_back(thread(func, head, tail,ref(total[i])));
29                 head = tail;
30
31         }
32         double sum = 0.0;
33         for(auto& x : a) x.join();
34         for (double &s : total) sum += s;
35         sum *= 4.0*width;
36         cout << "Estimation␣of␣pi␣is␣"
37                 << fixed << setprecision(10)
38                 << sum << "\n";
39         return 0;
40 }
```

ソースコード 5: future.cpp

```
1  #include <iostream>
2  #include <iomanip>
3  #include <future>
4  #include <vector>
5  #include <climits>
6  using namespace std;
7
8  const int intervals = 40000000;
9  const double width = 1.0 / intervals;
10
11 double func(double head,double tail) {
12         double sum=0;
13         for (int i = head; i < tail; i++ ) {
14                 const double x = (i + 0.5)*width;
15                 sum += 1/(1.0 + x*x);
16         }
17         return sum;
18 }
19
20 int main()
21 {
22         const int num = thread::hardware_concurrency();
23         vector<future<double>> a;
24         for (int i=0; i<num; i++){
25                 double tmp = i*(intervals/num);
26                 a.push_back(async(launch::async, func, tmp, tmp+(intervals/num)));
27         }
28         double sum = 0;
29         for(auto& x : a) sum += x.get();
30         sum *= 4.0*width;
31         cout << "Estimation␣of␣pi␣is␣"
```

```
32                  << fixed << setprecision(10)
33                  << sum << "\n";
34          return 0;
35  }
```

ソースコード 6: futureIntMax.cpp

```
1   #include <iostream>
2   #include <iomanip>
3   #include <future>
4   #include <vector>
5   #include <climits>
6   using namespace std;
7
8   const int intervals = INT_MAX;
9   const double width = 1.0 / intervals;
10
11  double func(double head,double tail) {
12          double sum=0;
13          for (int i = head; i < tail; i++ ) {
14                  const double x = (i + 0.5)*width;
15                  sum += 1/(1.0 + x*x);
16          }
17          return sum;
18  }
19
20  int main()
21  {
22          const int num = thread::hardware_concurrency();
23          vector<future<double>> a;
24          double head = 0;
25          for (int i=0; i<num; i++){
26                  double tail = head + (intervals/num);
27                  if(i==num-1)tail=intervals;
28                  a.push_back(async(launch::async, func, head, tail));
29                  head = tail;
30          }
31          double sum = 0;
32          for(auto& x : a) sum += x.get();
33          sum *= 4.0*width;
34          cout << "Estimation of pi is "
35                  << fixed << setprecision(10)
36                  << sum << "\n";
37          return 0;
38  }
```