Теория параллелизма

Отчёт

Уравнения теплопроводности

Выполнил Солопов Илья
Группа 21932
07.03.2023

Введение

Цели работы

Реализовать решение уравнения теплопроводности (пятиточечный шаблон) в двумерной области на равномерных сетках. С условиями линейной интерполяции между углами области, а также ограниченными значениями точности и максимального числа итераций.

На вход программе через командную строку должны подаваться параметры: точность, размер сетки, количество итераций.

Вывод программы – количество итераций и достигнутое значение ошибки.

Перенести программу на GPU используя директивы OpenACC.

Сравнить скорость работы для разных размеров сеток на центральном и графическом процессоре.

Произвести профилирование программы и оптимизацию кода.

Используемый компилятор

Использовался компилятор GNU C++ для сборки программы под однопотоковое исполнение центрального процессора и PGC++ для запуска на центральном процессоре в режиме многопоточности и графическом процессоре.

Используемый профилировщик

Использовался профилировщик NVIDIA NsightSystems.

Замер времени работы

Замер времени работы программы производился при помощи библиотеки chrono языка c++ и при помощи nvprof.

Выполнение на CPU

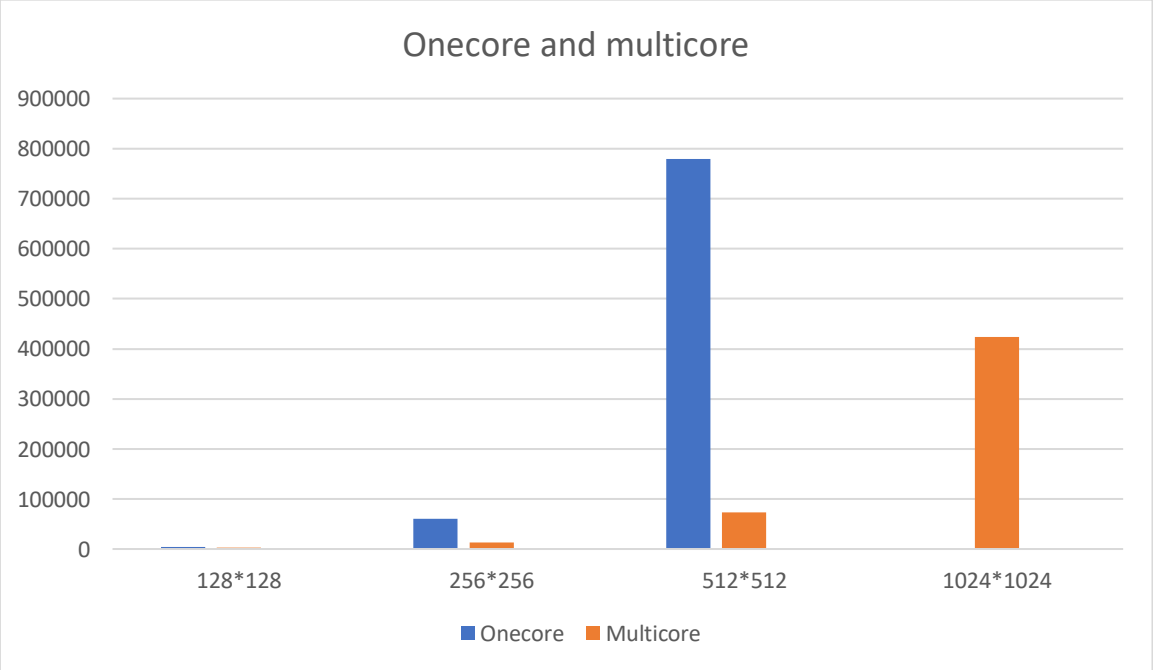
CPU-onecore

Размер сетки	Время выполнения, мс	Точность	Количество итераций
128*128	4290	9.9986e-07	28850
256*256	59831	9.9993e-07	97948
512*512	779286	9.99997e-07	319772

CPU-multicore

Размер сетки	Время выполнения, мс	Точность	Количество итераций
128*128	3310	9.9986e-07	28850
256*256	13087	9.9993e-07	97948
512*512	72766	9.99997e-07	319772
1024*1024	423753	9.99996e-07	987191

Диаграмма сравнения времени работы



.

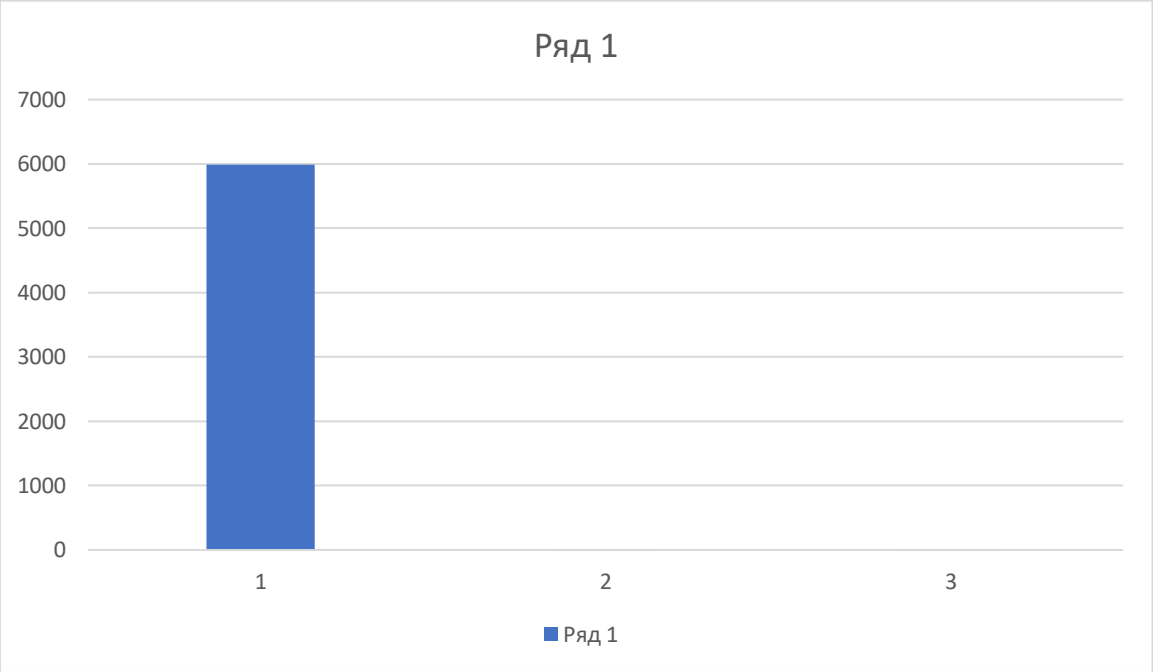
Выполнение на GPU

Этапы оптимизации

Оптимизация проводилась на сетке размером 512x512. Количество итераций при профилировании 100.

Этап №	Время выполнения, мс	Точность	Количество итераций	Комментарии (что было сделано)
1	5989.54	0.0359466	100	Для распараллеливания использованы только директивы kernels
2	6.66	0.0359466	100	Циклы распараллелены с помощью директив parallel loop. Использованы редукции и present. Двойные циклы объединены с помощью collapse
3	6.128	0.0359466	100	Замена поэлементного обмена A и Anew после каждой итерации внешнего цикла на swap через временную переменную

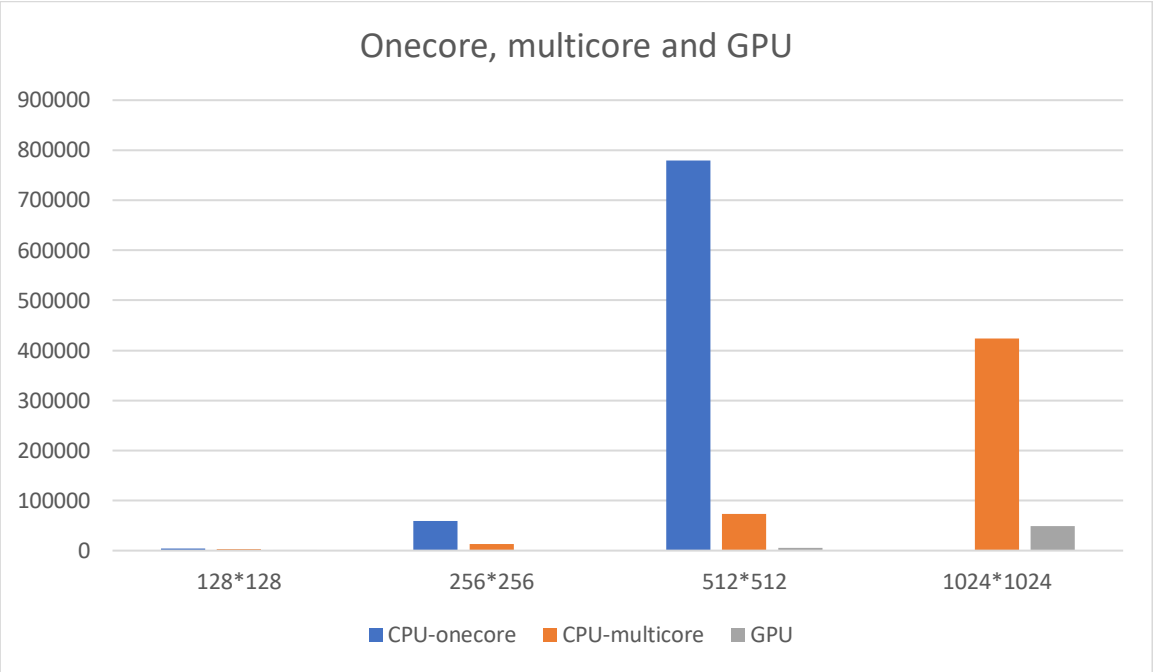
Диаграмма оптимизации



GPU – оптимизированный вариант

Размер сетки	Время выполнения, мс	Точность	Количество итераций
128*128	276.3	9.99899e-07	27809
256*256	1154.49	9.99987e-07	93751
512*512	5433.69	9.99982e-07	302922
1024*1024	49137.18	9.99995e-07	919656

Диаграмма сравнения времени работы


Onecore, multicore and GPU
900000
800000
700000
600000
500000
400000
300000
200000
100000
0
128*128
256*256
512*512
1024*1024
CPU-onecore
CPU-multicore
GPU

Вывод

Полученные результаты говорят нам о том, что для сетки размером 128*128 результаты будут примерно одни и те же, но при её увеличении GPU работает намного быстрее. Также, распараллеливание с помощью директив parallel loop с использованием редукции даёт многократный прирост к скорости выполнения.

Приложение

Ссылка на репозиторий

https://github.com/IlS0/Parallelism-tasks

Код программы на GPU

```
#include <iostream>
#include <cmath>
#include <string>

//поддержка double
#define LF_SUP

#ifdef LF_SUP
#define TYPE double
#define ABS fabs
#define MAX fmax
#define CAST std::stod
#else
#define TYPE float
#define ABS fabsf
#define MAX fmaxf
#define CAST std::stof
#endif

//инициализация сетки
void initArr(TYPE** A,int n){
    A[0][0]=10.0;
    A[0][n-1] = 20.0;
    A[n-1][0]=30.0;
    A[n-1][n-1]=20.0;

    #pragma acc parallel loop present(A[0:n][0:n])
    for (int i{1};i<n-1;++i){
        A[0][i]=10+((A[0][n-1] - A[0][0])/ (n-1))*i;
        A[i][0]=10+((A[n-1][0] - A[0][0])/ (n-1))*i;
        A[n-1][i]=10+((A[n-1][0] - A[n-1][n-1])/ (n-1))*i;
        A[i][n-1]=10+((A[n-1][n-1] - A[n-1][0])/ (n-1))*i;
    }
}

```

```
36 void solution(TYPE tol,int iter_max,int n){
37     TYPE error {1.0};
38     int iter{0};
39     TYPE** A = new TYPE*[n],**Anew = new TYPE*[n];
40     for ( int i {0}; i < n; ++i){
41         A[i] = new TYPE[n];
42         Anew[i] = new TYPE[n];
43     }
44 
45     #pragma acc enter data copyin(A [0:n] [0:n],error) create (Anew[0:n][0:n])
46 
47     initArr(A,n);
48     initArr(Anew,n);
49 
50     while (error > tol && iter < iter_max){
51         error = 0.0;
52         #pragma acc update device(error)
53 
54         #pragma acc parallel loop collapse(2) present(A[0:n][0:n]) reduction(max : error)
55         for (int j {1}; j < n - 1; ++j){
56             for (int i {1}; i < n - 1; ++i){
57                 Anew[j][i] = 0.25 * (A[j][i + 1] + A[j][i - 1] + A[j - 1][i] + A[j + 1][i]);
58                 error = MAX(error, ABS(Anew[j][i] - A[j][i]));
59             }
60         }
61 
62         //swap без цикла
63         TYPE** temp = A;
64         A = Anew;
65         Anew = temp;
66 
67         ++iter;
68         #pragma acc update host(error)
69     }
70 
71     #pragma acc exit data delete(A [0:n] [0:n],error,Anew [0:n] [0:n])
72 
73     std::cout<<"Iterations: "<<iter<<std::endl<<"Error: "<<error<<std::endl;
74 
75     for (int i {0}; i < n; i++){
76         delete[] A[i];
77         delete[] Anew[i];
78     }
79     delete [] A;
80     delete [] Anew;
81 }
```

```
82
83  int main(int argc, char *argv[]){
84      TYPE tol{1e-6};
85      int iter_max{1000000},n{128}; //значения для отладки, по умолчанию инициализировать нулями
86
87      std::string tmpStr;
88      //-t - точность
89      //-n - размер сетки
90      //-i - кол-во итераций
91      for (int i{1};i<argc;++i){
92          tmpStr = argv[i];
93          if(!tmpStr.compare("-t")){
94              tol = CAST(argv[i + 1]);
95              ++i;
96          }
97
98          if(!tmpStr.compare("-i")) {
99              iter_max = std::stoi(argv[i + 1]);
100             ++i;
101         }
102
103         if(!tmpStr.compare("-n")) {
104             n = std::stoi(argv[i + 1]);
105             ++i;
106         }
107     }
108     solution(tol,iter_max,n);
109 }
110
111
```

Код программы на CPU

```cpp
#include <iostream>
#include <cmath>
#include <string>
#include <chrono>

//поддержка double
#define LF_SUP

#ifdef LF_SUP
#define TYPE double
#define ABS fabs
#define MAX fmax
#define CAST std::stod
#else
#define TYPE float
#define ABS fabsf
#define MAX fmaxf
#define CAST std::stof
#endif

//инициализация сетки
void initArr(TYPE** A,int n){
    A[0][0]=10.0;
    A[0][n-1] = 20.0;
    A[n-1][0]=30.0;
    A[n-1][n-1]=20.0;

    #pragma acc parallel loop present(A[0:n][0:n])
    for (int i{1};i<n-1;++i){
        A[0][i]=10+((A[0][n-1] - A[0][0])/ (n-1))*i;
        A[i][0]=10+((A[n-1][0] - A[0][0])/ (n-1))*i;
        A[n-1][i]=10+((A[n-1][0] - A[n-1][n-1])/ (n-1))*i;
        A[i][n-1]=10+((A[n-1][n-1] - A[n-1][0])/ (n-1))*i;
    }
}

```

```
37  void solution(TYPE tol,int iter_max,int n){
38      TYPE error {1.0};
39      int iter{0};
40      TYPE** A = new TYPE*[n],**Anew = new TYPE*[n];
41      for ( int i {0}; i < n; ++i){
42          A[i] = new TYPE[n];
43          Anew[i] = new TYPE[n];
44      }
45
46      #pragma acc enter data copyin(A [0:n] [0:n],error) create (Anew[0:n][0:n])
47
48      initArr(A,n);
49      initArr(Anew,n);
50
51      while (error > tol && iter < iter_max){
52          error = 0.0;
53          #pragma acc update device(error)
54
55          #pragma acc parallel loop collapse(2) present(A[0:n][0:n]) reduction(max : error)
56          for (int j {1}; j < n - 1; ++j){
57              for (int i {1}; i < n - 1; ++i){
58                  Anew[j][i] = 0.25 * (A[j][i + 1] + A[j][i - 1] + A[j - 1][i] + A[j + 1][i]);
59                  error = MAX(error, ABS(Anew[j][i] - A[j][i]));
60              }
61          }
62
63          //swap без цикла
64          TYPE** temp = A;
65          A = Anew;
66          Anew = temp;
67
68          ++iter;
69          #pragma acc update host(error)
70      }
```

```
72      #pragma acc exit data delete(A [0:n] [0:n],error,Anew [0:n] [0:n])
73
74      std::cout<<"Iterations: "<<iter<<std::endl<<"Error: "<<error<<std::endl;
75
76      for (int i {0}; i < n; i++){
77          delete[] A[i];
78          delete[] Anew[i];
79      }
80      delete [] A;
81      delete [] Anew;
82  }
83
84  int main(int argc, char *argv[]){
85      auto start = std::chrono::high_resolution_clock::now();
86      TYPE tol{1e-6};
87      int iter_max{1000000},n{128}; //значения для отладки, по умолчанию инициализировать нулями
88
89      std::string tmpStr;
90      //-t - точность
91      //-n - размер сетки
92      //-i - кол-во итераций
93      for (int i{1};i<argc;++i){
94          tmpStr = argv[i];
95          if(!tmpStr.compare("-t")){
96              tol = CAST(argv[i + 1]);
97              ++i;
98          }
99
100         if(!tmpStr.compare("-i")) {
101             iter_max = std::stoi(argv[i + 1]);
102             ++i;
103         }
104
105         if(!tmpStr.compare("-n")) {
106             n = std::stoi(argv[i + 1]);
107             ++i;
108         }
109     }
109     }
110     solution(tol,iter_max,n);
111     auto end = std::chrono::high_resolution_clock::now() - start;
112     long long microseconds = std::chrono::duration_cast<std::chrono::microseconds>(end).count();
113     std::cout<<"Time (ms): "<<microseconds/1000<<std::endl;
114 }
115
116
```