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

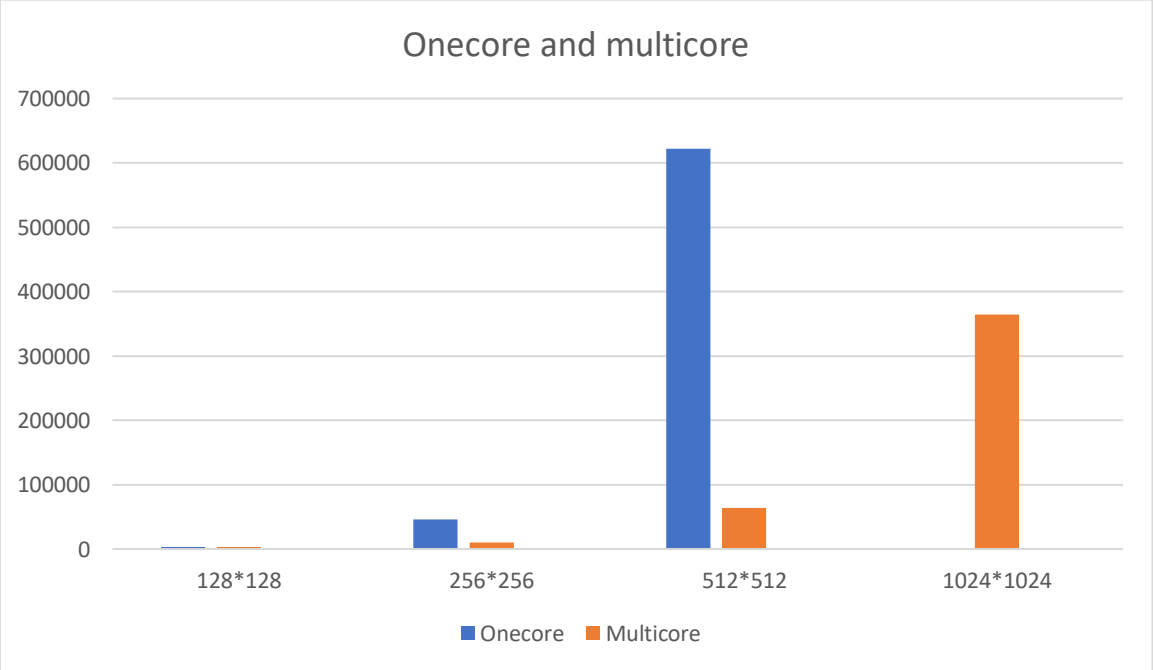
CPU-onecore

Размер сетки	Время выполнения, мс	Точность	Количество итераций
128*128	3378.5	9.97841e-07	30081
256*256	46056.25	9.97808e-07	102913
512*512	622527.5	9.93016e-07	339969

CPU-multicore

Размер сетки	Время выполнения, мс	Точность	Количество итераций
128*128	3431.83	9.97841e-07	30081
256*256	10365.57	9.97808e-07	102913
512*512	63836	9.93016e-07	339969
1024*1024	364049	1.373e-06	1000000

Диаграмма сравнения времени работы



.

Выполнение на GPU

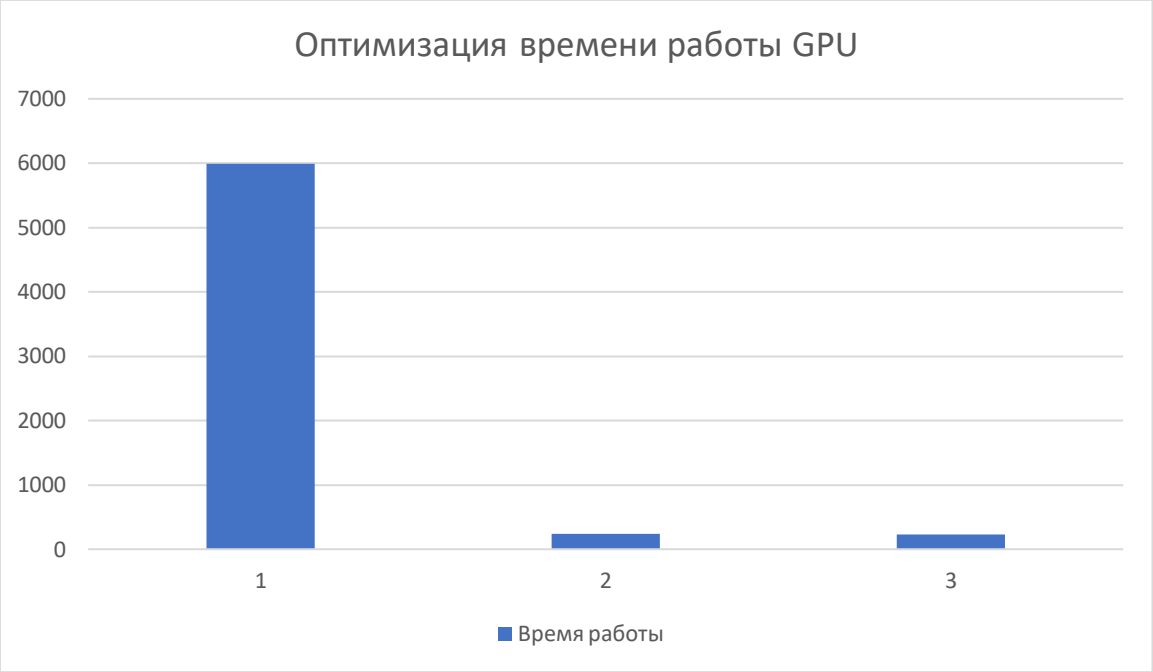
Этапы оптимизации

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	238.67	0.107043	100	Циклы распараллелены с помощью директив parallel loop. Использованы редукции и present. Двойные циклы объединены с помощью collapse
3	224.2	0.0359466	100	Замена поэлементного обмена A и Anew после каждой итерации внешнего цикла на swap через временную переменную. Использование директивы async. Ошибка

				высчитывается не на каждой итерации.

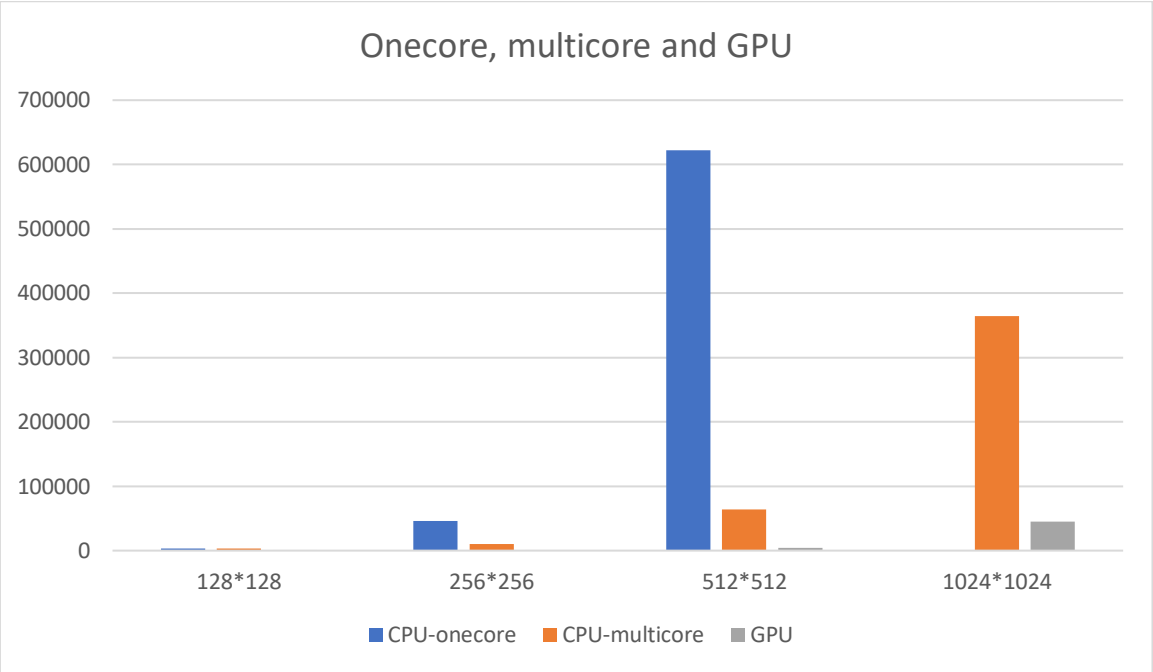
Диаграмма оптимизации



GPU – оптимизированный вариант

Размер сетки	Время выполнения, мс	Точность	Количество итераций
128*128	228.28	9.9998e-07	30081
256*256	821.27	9.9993e-07	102913
512*512	4092.98	9.99984e-07	339969
1024*1024	44710.47	1.373e-06	1000000

Диаграмма сравнения времени работы


Onecore, multicore and GPU
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 и async дают многократный прирост к скорости выполнения.

Приложение

Ссылка на репозиторий

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

Код программы на GPU

```cpp
#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)
{
    #pragma acc kernels
    {
        A[0][0] = 10.0;
        A[0][n - 1] = 20.0;
        A[n - 1][0] = 20.0;
        A[n - 1][n-1] = 30.0;
    }

    #pragma acc parallel loop present(A [0:n] [0:n])
    for (int i{1}; i < n - 1; ++i)
    {
        A[0][i] = 10 + (i * 10.0 / (n - 1));
        A[i][0] = 10 + (i * 10.0 / (n - 1));
        A[n - 1][i] = 20 + (i * 10.0 / (n - 1));
        A[i][n - 1] = 20 + (i * 10.0 / (n - 1));
    }
}
```

```
42  void printArr(TYPE **A, int n)
43  {
44      for (int i {0}; i < n; ++i)
45      {
46          for (int j {0}; j < n; ++j)
47          {
48              #pragma acc kernels present (A[0:n][0:n])
49              printf("%lf ", A[i][j]);
50          }
51          std::cout<<std::endl;
52      }
53
54  }
55
56  void solution(TYPE tol, int iter_max, int n)
57  {
58      TYPE error{1.0};
59      int iter{0};
60      bool flag {true};
61
62      TYPE **A = new TYPE *[n], **Anew = new TYPE *[n];
63      for (int i{0}; i < n; ++i)
64      {
65          A[i] = new TYPE[n];
66          Anew[i] = new TYPE[n];
67      }
68
69      #pragma acc enter data copyin(A [0:n] [0:n], error) create(Anew [0:n] [0:n])
70
71      initArr(A, n);
72      initArr(Anew, n);
73      //printArr(Anew,n);
74
```

```
75      while ( iter < iter_max)
76      {
77          flag = !(iter % n);
78
79          if (flag){
80              #pragma acc kernels present(error)
81              error = 0;
82              //#pragma acc update device(error)
83          }
84
85          #pragma acc parallel loop collapse(2) present(A, Anew, error) reduction(max: error) async(1)
86          for (int j{1}; j < n - 1; ++j)
87          {
88              for (int i{1}; i < n - 1; ++i)
89              {
90                  Anew[j][i] = 0.25 * (A[j][i + 1] + A[j][i - 1] + A[j - 1][i] + A[j + 1][i]);
91                  if (flag)
92                      error = MAX(error, ABS(Anew[j][i] - A[j][i]));
93              }
94          }
95
96          // swap без цикла
97          TYPE **temp = A;
98          A = Anew;
99          Anew = temp;
100
101         ++iter;
102         if (flag){
103             #pragma acc update host(error) wait(1)
104             if (error <= tol)
105                 break;
106         }
107     }
108     #pragma acc wait(1)
109
110     #pragma acc exit data delete (A [0:n] [0:n], error, Anew [0:n] [0:n])
111
112     std::cout << "Iterations: " << iter << std::endl<< "Error: " << error << std::endl;
113
```

```
    for (int i{0}; i < n; i++)
    {
        delete[] A[i];
        delete[] Anew[i];
    }
    delete[] A;
    delete[] Anew;
}

int main(int argc, char *argv[])
{

    TYPE tol{1e-6};
    int iter_max{1000000}, n{128}; // значения для отладки, по умолчанию инициализировать нулями

    std::string tmpStr;
    //-t - точность
    //-n - размер сетки
    //-i - кол-во итераций
    for (int i{1}; i < argc; ++i)
    {
        tmpStr = argv[i];
        if (!tmpStr.compare("-t"))
        {
            tol = CAST(argv[i + 1]);
            ++i;
        }

        if (!tmpStr.compare("-i"))
        {
            iter_max = std::stoi(argv[i + 1]);
            ++i;
        }

        if (!tmpStr.compare("-n"))
        {
            n = std::stoi(argv[i + 1]);
            ++i;
        }
    }

    solution(tol, iter_max, n);
}
```

Код программы на CPU

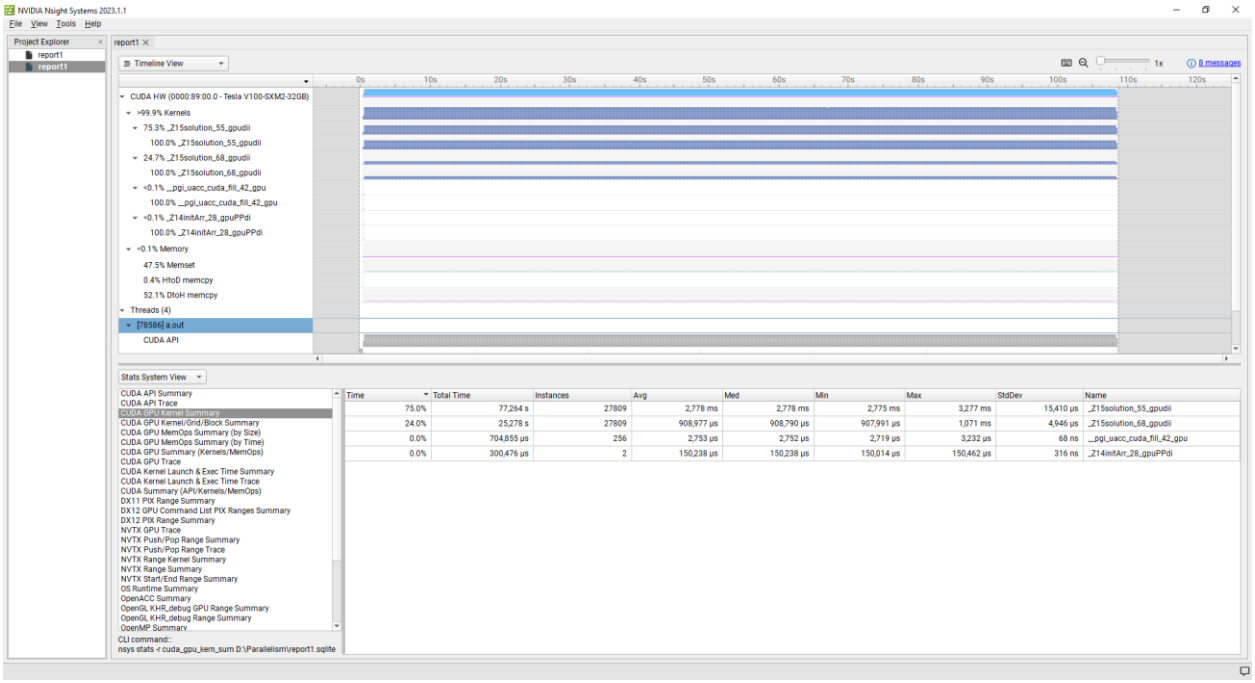
```cpp
#include <iostream>
#include <cmath>
#include <string>
#include <chrono>

//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      }
```

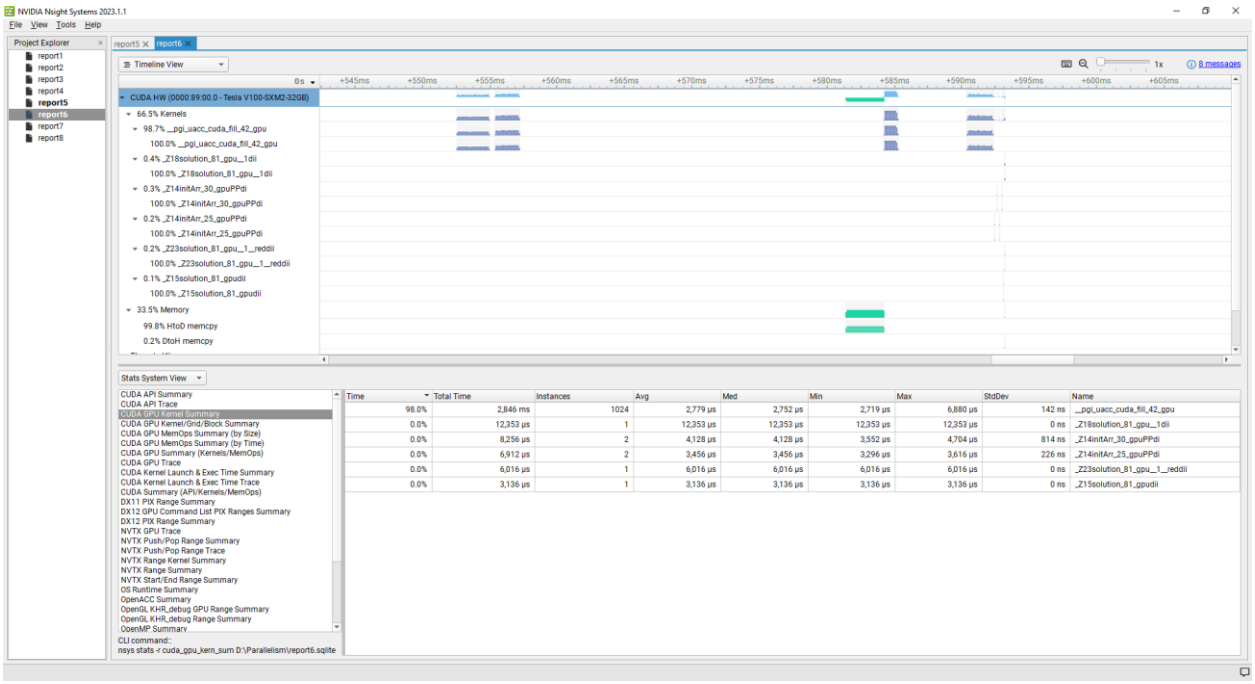
```
72      #pragma acc exit data delete(A [0:n] [0:n],error,Anew [0:n] [0:n])
73
74      std::cout<<"Iterations: "<<iter<<std::endl<<"Error: "<<error<<std::endl;
75
76      for (int i {0}; i < n; i++){
77          delete[] A[i];
78          delete[] Anew[i];
79      }
80      delete [] A;
81      delete [] 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
```

Скриншоты из профилировщика

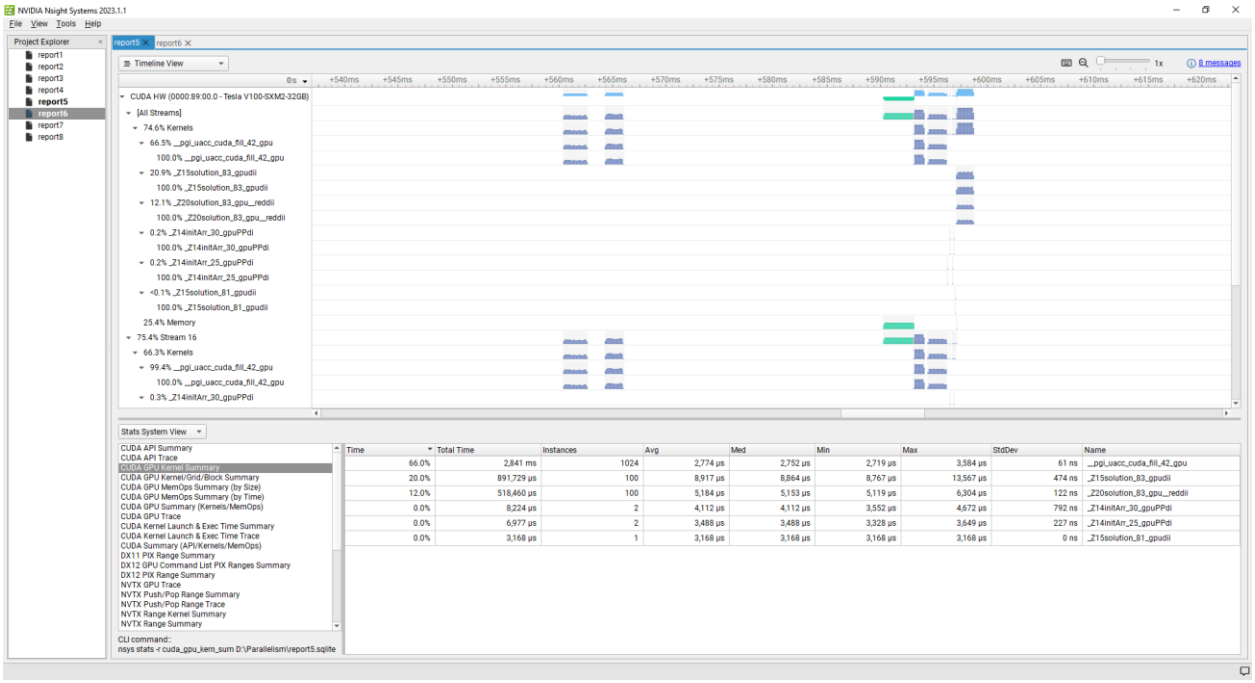
Этап 1


NVIDIA Nsight Systems 2023.1.1
File View Tools Help
Project Explorer
report1
Timeline View
CUDA HW (0000:89:00.0 - Tesla V100-SXM2-32GB)
Stats System View
CUDA API Summary
CUDA API Trace
CUDA GPU Kernel Summary
CUDA GPU Kernel/Grid/Block Summary
CUDA GPU MemOps Summary (by Size)
CUDA GPU MemOps Summary (by Time)
CUDA GPU Summary (Kernels/MemOps)
CUDA GPU Trace
CUDA Kernel Launch & Exec Time Summary
CUDA Kernel Launch & Exec Time Trace
CUDA Summary (API/Kernels/MemOps)
DX11 PIX Range Summary
DX12 GPU Command List PIX Ranges Summary
DX12 PIX Range Summary
NVTX GPU Trace
NVTX Push/Pop Range Summary
NVTX Push/Pop Range Trace
NVTX Range Kernel Summary
NVTX Range Summary
NVTX Start/End Range Summary
OS Runtime Summary
OpenACC Summary
OpenGL KHR_debug GPU Range Summary
OpenGL KHR_debug Range Summary
OpenMP Summary
CLI command::
nsys stats -r cuda_gpu_kern_sum D:\Parallelism\report1.sqlite

Этап 2


NVIDIA Nsight Systems 2023.1.1
File View Tools Help
Project Explorer
report6
Timeline View
CUDA HW (0000:89:00.0 - Tesla V100-SXM2-32GB)
Stats System View
CUDA API Summary
CUDA API Trace
CUDA GPU Kernel Summary
CUDA GPU Kernel/Grid/Block Summary
CUDA GPU MemOps Summary (by Size)
CUDA GPU MemOps Summary (by Time)
CUDA GPU Summary (Kernels/MemOps)
CUDA GPU Trace
CUDA Kernel Launch & Exec Time Summary
CUDA Kernel Launch & Exec Time Trace
CUDA Summary (API/Kernels/MemOps)
DX11 PIX Range Summary
DX12 GPU Command List PIX Ranges Summary
DX12 PIX Range Summary
NVTX GPU Trace
NVTX Push/Pop Range Summary
NVTX Push/Pop Range Trace
NVTX Range Kernel Summary
NVTX Range Summary
NVTX Start/End Range Summary
OS Runtime Summary
OpenACC Summary
OpenGL KHR_debug GPU Range Summary
OpenGL KHR_debug Range Summary
OpenMP Summary
CLI command::
nsys stats -r cuda_gpu_kern_sum D:\Parallelism\report6.sqlite

Этап 3


NVIDIA Nsight Systems 2023.1.1
File View Tools Help
Project Explorer
report1
report2
report3
report4
report5
report6
report7
report8
report5
report6
Timeline View
1x
8 messages
0s
+540ms +545ms +550ms +555ms +560ms +565ms +570ms +575ms +580ms +585ms +590ms +595ms +600ms +605ms +610ms +615ms +620ms
CUDA HW (0000:89:00.0 - Tesla V100-SXM2-32GB)
[All Streams]
74.6% Kernels
66.5% __pgi_uacc_cuda_fill_42_gpu
100.0% __pgi_uacc_cuda_fill_42_gpu
20.9% _Z15solution_83_gpudii
100.0% _Z15solution_83_gpudii
12.1% _Z20solution_83_gpu__reddii
100.0% _Z20solution_83_gpu__reddii
0.2% _Z14initArr_30_gpuPPdi
100.0% _Z14initArr_30_gpuPPdi
0.2% _Z14initArr_25_gpuPPdi
100.0% _Z14initArr_25_gpuPPdi
<0.1% _Z15solution_81_gpudii
100.0% _Z15solution_81_gpudii
25.4% Memory
75.4% Stream 16
66.3% Kernels
99.4% __pgi_uacc_cuda_fill_42_gpu
100.0% __pgi_uacc_cuda_fill_42_gpu
0.3% _Z14initArr_30_gpuPPdi
Stats System View
CUDA API Summary
CUDA API Trace
CUDA GPU Kernel Summary
CUDA GPU Kernel/Grid/Block Summary
CUDA GPU MemOps Summary (by Size)
CUDA GPU MemOps Summary (by Time)
CUDA GPU Summary (Kernels/MemOps)
CUDA GPU Trace
CUDA Kernel Launch & Exec Time Summary
CUDA Kernel Launch & Exec Time Trace
CUDA Summary (API/Kernels/MemOps)
DX11 PIX Range Summary
DX12 GPU Command List PIX Ranges Summary
DX12 PIX Range Summary
NVTX GPU Trace
NVTX Push/Pop Range Summary
NVTX Push/Pop Range Trace
NVTX Range Kernel Summary
NVTX Range Summary
CLI command:
nsys stats -r cuda_gpu_kern_sum D:\Parallelism\report5.sqlite
Time Total Time Instances Avg Med Min Max StdDev Name
66.0% 2,841 ms 1024 2,774 µs 2,752 µs 2,719 µs 3,584 µs 61 ns __pgi_uacc_cuda_fill_42_gpu
20.0% 891,729 µs 100 8,917 µs 8,864 µs 8,767 µs 13,567 µs 474 ns _Z15solution_83_gpudii
12.0% 518,460 µs 100 5,184 µs 5,153 µs 5,119 µs 6,304 µs 122 ns _Z20solution_83_gpu__reddii
0.0% 8,224 µs 2 4,112 µs 4,112 µs 3,552 µs 4,672 µs 792 ns _Z14initArr_30_gpuPPdi
0.0% 6,977 µs 2 3,488 µs 3,488 µs 3,328 µs 3,649 µs 227 ns _Z14initArr_25_gpuPPdi
0.0% 3,168 µs 1 3,168 µs 3,168 µs 3,168 µs 3,168 µs 0 ns _Z15solution_81_gpudii