**Федеральное государственное бюджетное образовательное учреждение высшего образования**

**«Уфимский университет науки и технологий»**

**Кафедра** Высокопроизводительных вычислительных технологий и систем

**Дисциплина:** Технологии параллельного программирования

**Отчет по лабораторной работе № 4**

**Тема:** «Параллельное вычисление произведения матриц с использованием OpenACC»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Группа ПМ-357 | Фамилия И.О. | Подпись | Дата | Оценка |
| Студент | Акмурзин М.Э. |  |  |  |
| Принял | Юлдашев А.В. |  |  |  |

**Уфа 2023**

**Цель:** Приобрести навыки распараллеливания циклов с использованием директив OpenACC.

**Теоретический материал**

OpenACC открытый стандарт параллельного программирования гибридных систем с использованием директив для языков C/C++/Fortran. Совместная разработка членов рабочей группы OpenMP Working Group on Accelerators , компаний NVIDIA, Cray, PGI и CAPS. Преимущественно основан на моделях программирования ускорителей, разработанных ранее компаниями PGI и CRAY. Реализован в компиляторах Cray, PGI и др.

**Задание:**

Сделать OpenACC-версии программ, написанных на предыдущих лабораторных и затестить.

**Практическая часть**

Была реализована OpenACC-версия программы, написанная в лабораторной работе №2.

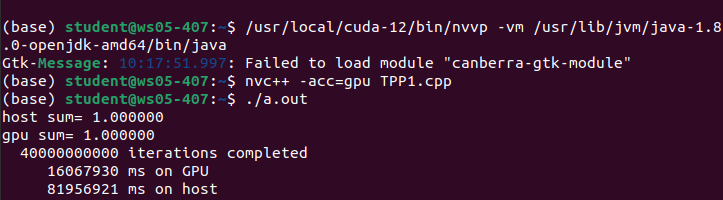
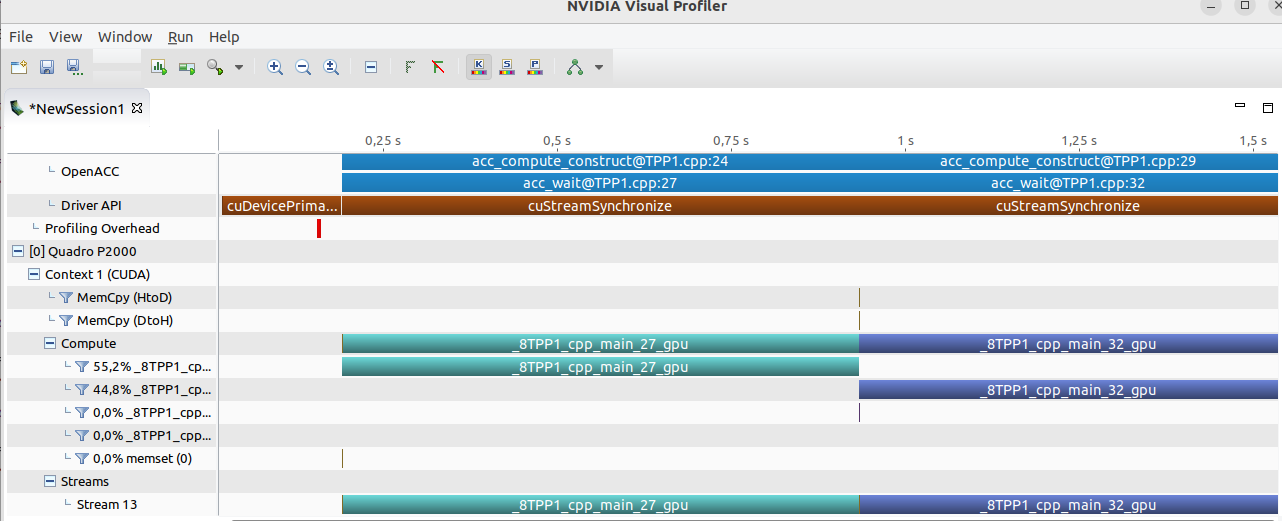


Рисунок 1 результат работы программы для вычисления суммы ряда



|  |  |  |
| --- | --- | --- |
| N | GPU | HOST |
| 150000000 | 0.157 | 0.328 |
| 15000000000 | 6.307 | 30.576 |
| 40000000000 | 16.067 | 81.956 |

Была реализована OpenACC-версия программы, написанная в лабораторной работе №3. Результаты для различных размерностей матриц представлены в таблице 1.

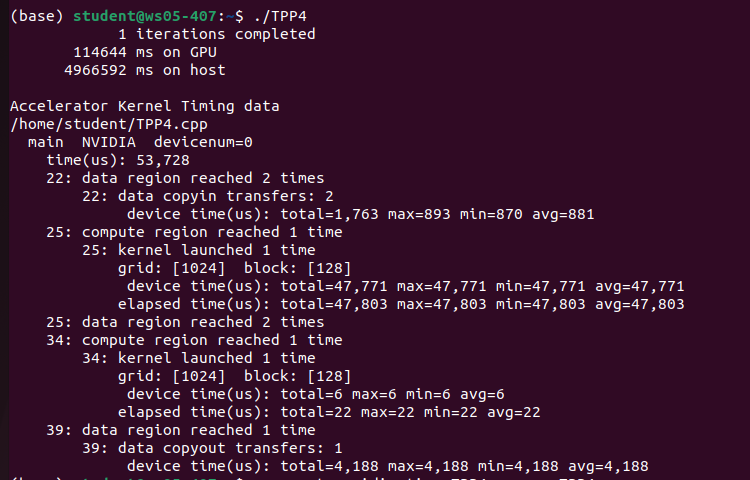
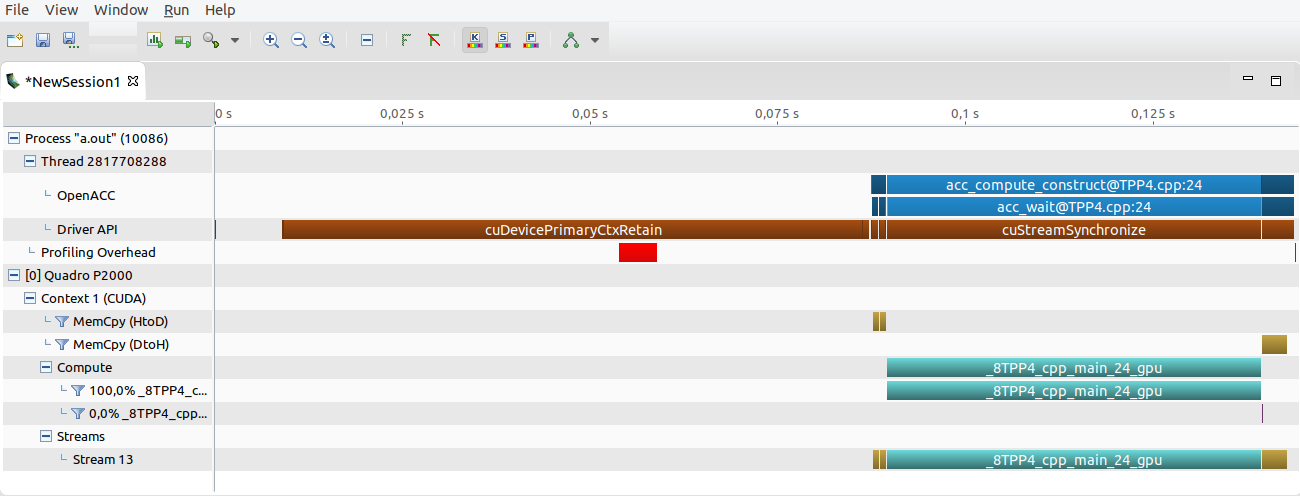


Рисунок 2 результат работы программы для вычисления произведения матриц

Таблица 1

|  |  |  |
| --- | --- | --- |
| N | GPU | HOST |
| 256 | 0.057418 | 0.023162 |
| 1024 | 0.114644 | 4.966592 |
| 4096 | 3.873619 | 504.996788 |

Рисунок 3: Nvidia Visual Profiler

**Вывод**

Было изучено распараллеливание циклов с использованием директив OpenACC, cделаны OpenACC-версии программ, написанных на предыдущих лабораторных.

**Листинг программы**

#define \_USE\_MATH\_DEFINES

#include <math.h>

#include <openacc.h>

#include <time.h>

int **main**()

{

long long N = 15000000000;

double sum = 0.0;

clock\_t t1, t2, t3;

long long cgpu, chost;

t1=clock();

for (long long i = 1; i < N; i += 2) {

sum += (2.0 \* i + 1) / i / (i + 1.0);

}

for (long long i = 2; i < N; i += 2) {

sum -= (2.0 \* i + 1) / i / (i + 1.0);

}

t2=clock();

printf("host sum= ");

printf("%f\n",sum);

sum=0;

chost = t2-t1;

t2=clock();

//#pragma acc parallel loop reduction(+:sum)

#pragma acc kernels

for (long long i = 1; i < N; i += 2) {

sum += (2.0 \* i + 1) / i / (i + 1.0);

}

//#pragma acc parallel loop reduction(+:sum)

#pragma acc kernels

for (long long i = 2; i < N; i += 2) {

sum -= (2.0 \* i + 1) / i / (i + 1.0);

}

t3=clock();

printf("gpu sum= ");

printf("%f\n",sum);

cgpu = t3-t2;

printf( "%13ld iterations completed\n", N );

printf( "%13ld ms on GPU\n", cgpu );

printf( "%13ld ms on host\n", chost );

return 0;

}

Умножение матриц

#include <stdlib.h>

#include <openacc.h>

#include <time.h>

const int N = 256;

double A[N][N];

double B[N][N];

double C[N][N];

int main() {

int size=N;

int q = 10;

srand(time(NULL));

//scanf("%d",&q);

clock\_t t1, t2;

long long cgpu;

for (int i = 0; i < N; i++)

for (int j = 0; j < N; j++) {

A[i][j] = (rand() % 256) / 256.0 - 0.5;

B[i][j] = (rand() % 256) / 256.0 - 0.5;

}

////////////////////////////////////

t1=clock();

#pragma acc enter data copyin(A[0:size][0:size],B[0:size][0:size])

#pragma acc enter data create(C[0:size][0:size])

for (int l = 0; l < q; l++) {

#pragma acc parallel loop

for (int i = 0; i < N; i++)

for (int j = 0; j < N; j++) {

C[i][j] = 0;

for (int k = 0; k < N; k++) C[i][j] += A[i][k] \* B[k][j];

}

}

t2=clock();

double s = 0.0;

#pragma acc parallel loop

for (int i = 0; i < N; i++)

for (int j = 0; j < N; j++) s += C[i][j] \* C[i][j];

cgpu = t2-t1;

/////////////////////////////////////

#pragma acc exit data copyout(C[0:size][0:size])

//#pragma acc exit delete(A[0:size][0:size],B[0:size][0:size],C[0:size][0:size])

printf( "%13ld iterations completed\n", q );

printf( "%13ld ms on GPU\n", cgpu );

}