Отчёт

Программа Знакомство с OpenACC

Компилятор PGCC

Подготовил Солопов Илья

Группа 21932

Время выполнения циклов

Время выполнения циклов на GPU

С использованием типа данных float были получены следующие показатели.

Type	Time(%)	Time	Calls	Avg	Min	Max	Name
GPU activities:	36.02%	120.61us	1	120.61us	120.61us	120.61us	main_24_gpu
	35.65%	119.36us	1	119.36us	119.36us	119.36us	main_26_gpu
	26.42%	88.479us	1	88.479us	88.479us	88.479us	main_26_gpu__red

С использованием типа данных double были получены следующие показатели

Type	Time(%)	Time	Calls	Avg	Min	Max	Name
GPU activities:	36.25%	131.20us	1	131.20us	131.20us	131.20us	main_26_gpu
	34.37%	124.38us	1	124.38us	124.38us	124.38us	main_24_gpu
	27.62%	99.967us	1	99.967us	99.967us	99.967us	main_26_gpu__red

Время выполнения циклов на CPU

При многопоточном исполнении программы с использованием типа данных float были получены следующие показатели.

Type	Time(%)	Time	Calls	Avg	Min	Max	Name
OpenACC (excl):	98.68%	26.343ms	1	26.343ms	26.343ms	26.343ms	acc_compute_construct@lab1_cpu.c:24
	1.32%	353.65us	1	353.65us	353.65us	353.65us	acc_compute_construct@lab1_cpu.c:26

При многопоточном исполнении программы с использованием типа данных double были получены следующие показатели.

Type	Time(%)	Time	Calls	Avg	Min	Max	Name
OpenACC (excl):	85.20%	20.994ms	1	20.994ms	20.994ms	20.994ms	acc_compute_construct@lab1_cpu.c:24
	14.80%	3.6466ms	1	3.6466ms	3.6466ms	3.6466ms	acc_compute_construct@lab1_cpu.c:26

Утилита nvprof не поддерживает сбор информации о работе CPU в однопоточном режиме. Из данных, представленных выше, можно заметить, что большую часть времени выполнения программы занимает исполнение циклов. Исходя из этого, время для кода, выполняющегося на одном потоке CPU, можно принять близким ко времени работы всей программы, которое представлено в следующем пункте.

Время работы программ

Общее время выполнения на GPU

Общее время исполнения программы на графическом процессоре с использованием типа данных float составило 3426.560767 миллисекунд, а с использованием double 3434.754238 миллисекунд.

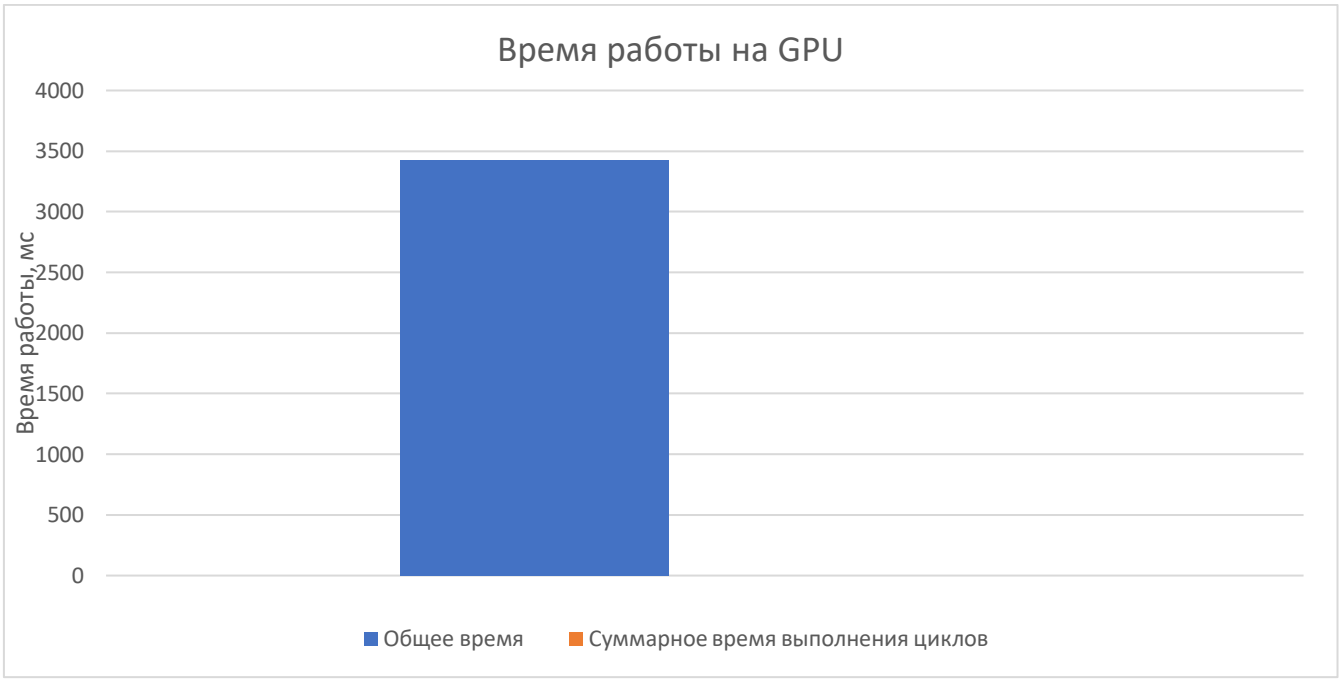
Общее время выполнения на CPU

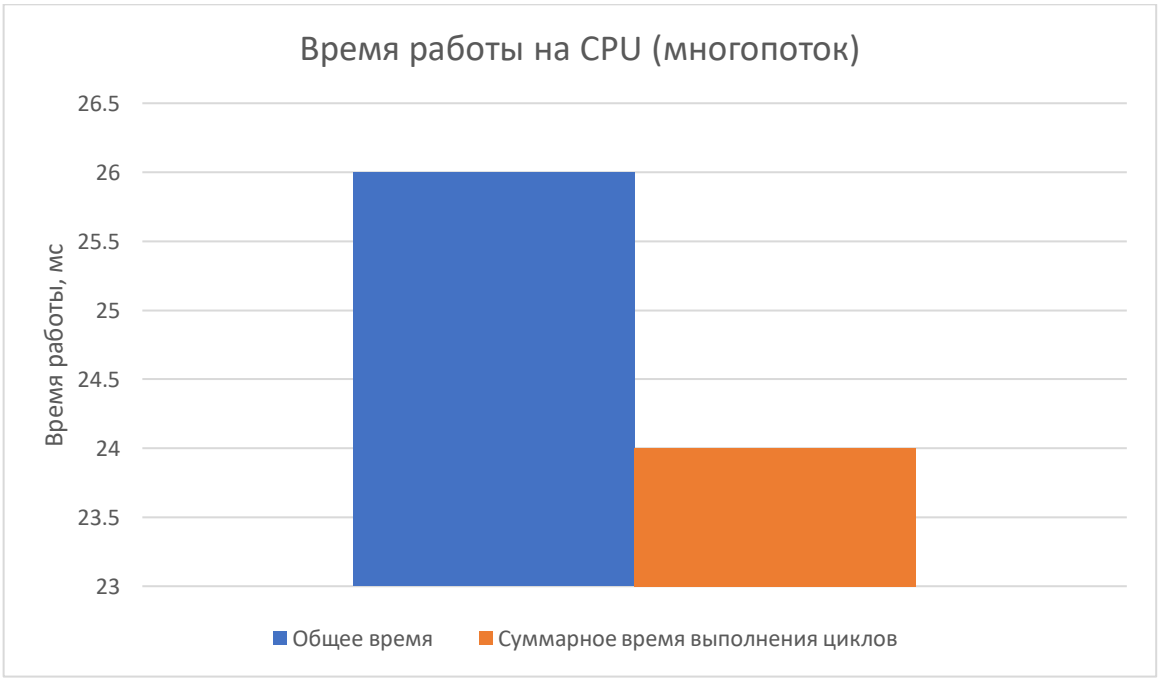
Многопоточность	**Затраченное время, *мс* (float)**	**Затраченное время, *мс* (double)**
Нет	276.890000	267.774000
Да	25.218737	27.805645

Точность расчётов

Процессор	Результат вычислений (float)	Результат вычислений (double)
GPU	-0.035797119140625000000000000	0.000025067563612424237362 6
CPU	-0.014050396159291267395019 5	0.000025005537699573189081 4

Диаграммы


Время работы на GPU
4000
3500
3000
2500
2000
1500
1000
500
0
Время работы, мс
Общее время
Суммарное время выполнения циклов


Время работы на CPU (многопоток)
26.5
26
25.5
25
24.5
24
23.5
23
Время работы, мс
Общее время
Суммарное время выполнения циклов

Коды программы

Код программы на GPU

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

#define N 10000000
#define PI 3.14159

//поддержка double
#define LF_SUP
#ifdef LF_SUP
#define TYPE double
#define SINUS sin
#else
#define TYPE float
#define SINUS sinf
#endif


int main(){
    TYPE *arr = (TYPE*)malloc(sizeof(TYPE) * N), sum =0.0,
    tmp=(PI*2/N);

    #pragma acc enter data create(arr[:N]) copyin(tmp,sum)

    #pragma acc kernels
    for (int i = 0; i < N; ++i){
        arr[i] = SINUS(tmp * i);
    }


    #pragma acc parallel loop reduction(+:sum)
    for (int i = 0; i < N; ++i) {
        sum += arr[i];
    }

    #pragma acc exit data copyout(sum)
    printf("%-32.25lf\n", sum);
    free(arr);
    return 0;
}
```

Код программы на CPU

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <time.h>


#define N 10000000
#define PI 3.14159

//поддержка double
#define LF_SUP
#ifdef LF_SUP
#define TYPE double
#define SINUS sin
#else
#define TYPE float
#define SINUS sinf
#endif

int main(){
    struct timespec start, end;

    clock_gettime(CLOCK_REALTIME, &start);
    TYPE *arr = (TYPE*)malloc(sizeof(TYPE) * N), sum =0.0,
    tmp=(PI*2/N);

    #pragma acc enter data create(arr[:N]) copyin(tmp,sum)

    #pragma acc kernels
    for (int i = 0; i < N; ++i){
        arr[i] = SINUS(tmp * i);
    }

    #pragma acc parallel loop reduction(+:sum)
    for (int i = 0; i < N; ++i) {
        sum += arr[i];
    }

    #pragma acc exit data copyout(sum)
    printf("%-32.25lf\n", sum);
    free(arr);
```

```
    clock_gettime(CLOCK_REALTIME, &end);
     double time = ((end.tv_sec - start.tv_sec) + (end.tv_nsec - start.tv_nsec) / 1000000000.0)*1000;


    printf("%lf ms\n", time);
    return 0;
}
```

Выбор оптимального решения

Полученные результаты говорят нам о том, что время исполнения циклов на графическом процессоре ничтожно мало по сравнению со временем, которое тратится на загрузку данных и аллокацию памяти. В то же время, центральный процессор в любом из режимов работает быстрее (на 1 и 2 порядка в однопотоке и многопотоке соответственно) графического. Исходя из полученных данных, оптимальным решением будет являться использование CPU в многопоточном режиме.