Отчет по первой работе

([nsuparral/task1 at master · FedorTikunov/nsuparral (github.com)](https://github.com/FedorTikunov/nsuparral/tree/master/task1))

Название кода: task1.c(находиться в папке task1 в github, на сервере в папке Again/task1)

При работы на GPU компилировался через pgcc (пример: pgcc task1.c -acc -Minfo=accel -fast -o problem1\_float\_GPU)

При работы на CPU компилировался через gcc и запускался через gprof (пример:  
>gcc -pg task1.c -lm -o problem1\_float\_CPU

>./problem1\_float\_CPU

>gprof problem1\_float\_CPU gmon.out >> CPU\_float.txt

)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Полное время работы(ms) | Работа цикла FuncArray(ms) | Работа цикла SumArray(ms) | Точность |
| GPU(float) | 331.02 | 0.09523 | 0.070719 | до 4 знака |
| GPU(double) | 335.76 | 0.11078 | 0.10931 | до 12 знака |
| CPU(double) | 264.9 | 40.18 | 30.13 | до 10 знака |
| CPU(float) | 274.97 | 30.13 | 30.13 | До 1 знака |

Как видно по графику при СPU полное время, затраченное на выполнение программы, меньше, однако при GPU точность выводимой суммы выше как при double, так и при float значениях. Тем самым, в зависимости от того важнее вам выше точность или время работы кода, вы берете или GPU(double) или CPU(double) соответственно.

Код в task1.c для float и double почти один и то же, за исключением макросов FORMAT = (double/float) и PFORMAT = ("summa = %0.23lf\n"/"summa = %0.23f\n")

КОД task1float.c:

#include <malloc.h>

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

#include <time.h>

#define FORMAT float

#define PFORMAT "summa = %0.23f\n"

#define MAX\_SIZE 10000000

void funcArray(FORMAT\* arr, size\_t len) {

FORMAT step = 3.141592653589783 \* 2 / MAX\_SIZE;

#pragma acc data copyin(step)

#pragma acc parallel loop gang num\_gangs(2048) vector vector\_length(64), present(arr)

for (size\_t i = 0; i < len; i++)

{

arr[i] = sinf(step \* i);

}

}

FORMAT sumArray(FORMAT\* arr, size\_t len) {

FORMAT sum = 0.0;

#pragma acc data copy(sum)

#pragma acc parallel loop gang num\_gangs(2048) reduction(+:sum) present(arr)

for (size\_t i = 0; i < len; i++)

{

sum += arr[i];

}

return sum;

}

int main() {

clock\_t before = clock();

double sec = 0.0;

FORMAT\* arr = (FORMAT\*)malloc(sizeof(FORMAT) \* MAX\_SIZE);

#pragma acc data create(arr[0:MAX\_SIZE])

{

funcArray(arr, MAX\_SIZE);

printf(PFORMAT, sumArray(arr, MAX\_SIZE));

sec += (FORMAT)(clock() - before)/ CLOCKS\_PER\_SEC;

printf("Time taken: %.5f\n", sec);

free(arr);

}

return 0;

}

КОД task1double.c:

#include <malloc.h>

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

#include <time.h>

#define FORMAT double

#define PFORMAT "summa = %0.23lf\n"

#define MAX\_SIZE 10000000

void funcArray(FORMAT\* arr, size\_t len) {

FORMAT step = 3.141592653589783 \* 2 / MAX\_SIZE;

#pragma acc data copyin(step)

#pragma acc parallel loop gang num\_gangs(2048) vector vector\_length(64), present(arr)

for (size\_t i = 0; i < len; i++)

{

arr[i] = sinf(step \* i);

}

}

FORMAT sumArray(FORMAT\* arr, size\_t len) {

FORMAT sum = 0.0;

#pragma acc data copy(sum)

#pragma acc parallel loop gang num\_gangs(2048) reduction(+:sum) present(arr)

for (size\_t i = 0; i < len; i++)

{

sum += arr[i];

}

return sum;

}

int main() {

clock\_t before = clock();

double sec = 0.0;

FORMAT\* arr = (FORMAT\*)malloc(sizeof(FORMAT) \* MAX\_SIZE);

#pragma acc data create(arr[0:MAX\_SIZE])

{

funcArray(arr, MAX\_SIZE);

printf(PFORMAT, sumArray(arr, MAX\_SIZE));

sec += (FORMAT)(clock() - before)/ CLOCKS\_PER\_SEC;

printf("Time taken: %.5f\n", sec);

free(arr);

}

return 0;

}