МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

**“Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики”**

**(НИУ ИТМО)**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Факультет программной инженерии и компьютерной техники

Направление подготовки 09.04.04 Программная инженерия

**Лабораторная работа № 1**

**“** **Автоматическое распараллеливание программ”**

**По дисциплине “Параллельные вычисления”**

       Студент группы Р4114

Шитов Григорий Семенович

         Преподаватель:

Жданов Андрей Дмитриевич

Санкт-Петербург, 2024 г.

Оглавление

[Описание решаемой задачи 3](#_Toc164108910)

[Краткая характеристика системы 4](#_Toc164108911)

[Программа lab1.c 5](#_Toc164108912)

[Результаты 8](#_Toc164108913)

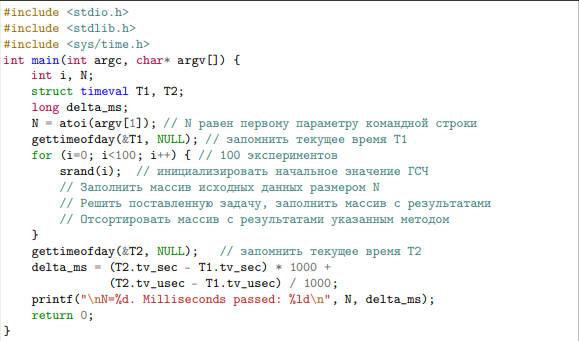
[Сравнение результатов при N1 и N2 8](#_Toc164108914)

[Сравнение результатов в динамике роста N 9](#_Toc164108915)

[Выводы 10](#_Toc164108916)

Описание решаемой задачи

1. На компьютере с многоядерным процессором установить Unix-подобную операционную систему и компилятор GCC версии не ниже 9.x. При невозможности установить Unix-подобную операционную систему или отсутствии компьютера с многоядерным процессором можно выполнять лабораторную работу на виртуальной машине. Минимальное число ядер при использовании виртуальной машины – два. Важным условием является отключение гипертрединга, для того, чтобы выполнить честные замеры времени.
2. На языке Cи написать консольную программу lab1.c, решающую задачу, указанную в п. 5.5 (см. ниже). В программе нельзя использовать библиотечные функции сортировки, выполнения матричных операций и расчёта статистических величин. В программе нельзя использовать библиотечные функции, отсутствующие в стандартных заголовочных файлах stdio.h, stdlib.h, sys/time.h, math.h. Задача должна решаться 100 раз с разными начальными значениями генератора случайных чисел (ГСЧ). Структура программы, примерно следующая:



1. Скомпилировать написанную программу без использования автоматического распараллеливания с помощью следующей команды: /home/user/gcc -O3 -Wall -Werror -lm -o lab1-seq lab1.c
2. Скомпилировать написанную программу, используя встроенное в gcc средство автоматического распараллеливания Graphite, с помощью следующей команды: /home/user/gcc -O3 -Wall -Werror -lm -floop-parallelize-all -ftree-parallelize-loops=K lab1.c -o lab1-par-K (переменной K поочерёдно присвоить хотя бы четыре значения: один, меньше числа физических ядер, равное числу физических ядер и больше числа физических ядер).
3. В результате получится одна нераспараллеленная программа и четыре или более распараллеленных.
4. Закрыть все работающие в операционной системе прикладные программы (включая Winamp, uTorrent, браузеры, Telegram и Skype), чтобы они не влияли на результаты последующих экспериментов. При использовании ноутбука необходимо иметь постоянное подключение к сети питания на время проведения эксперимента.
5. Запускать файл lab1-seq из командной строки, увеличивая значения N до значения N1, при котором время выполнения превысит 0.01 с. Подобным образом найти значение N=N2, при котором время выполнения превысит 5 с.
6. Используя найденные значения N1 и N2, выполнить следующие эксперименты (для автоматизации проведения экспериментов рекомендуется написать скрипт):

* запускать lab1-seq для значений 𝑁 = 𝑁1, 𝑁1 + Δ, 𝑁1 + 2Δ, 𝑁1 + 3Δ, . . . , 𝑁2 и записывать получающиеся значения времени delta\_ms(N) в функцию seq(N);
* запускать lab1-par-K для значений 𝑁 = 𝑁1, 𝑁1 + Δ, 𝑁1 + 2Δ, 𝑁1 + 3Δ, . . . , 𝑁2 и записывать получающиеся значения времени delta\_ms(N) в функцию par-K(N);
* значение Δ выбрать так: Δ = (𝑁2 − 𝑁1)/10

1. Провести верификацию значения X. Добавить в конец цикла вывод значения X и изменить число экспериментов на 5. Сравнить значения X для распараллеленной программы и нераспараллеленной.

Краткая характеристика системы

Операционная система: Ubuntu 22.04

Тип системы: UNIX

Процессор: AMD Ryzen 3

Оперативная память: 8ГБ

Количество физических ядер: 2

Количество логических ядер: 4

gcc version 11.4.0 (Ubuntu 11.4.0-1ubuntu1~22.04)

Программа lab1.c

#include <stdio.h>

#include <stdlib.h>

#include <sys/time.h>

#define \_USE\_MATH\_DEFINES

#include <math.h>

#define M1SIZE (N)

#define M2SIZE (N / 2)

#define RFACTOR (1.24733)

void gnomeSort(double \*array, int size);

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

{

    int i, N, j;

    unsigned int seed;

    double sum = 0;

    struct timeval T1, T2;

    long delta\_ms;

    FILE \*resultOfTest = fopen("Results.txt", "a");

    N = atoi(argv[1]); // N равен первому параметру командной строки

    double \*restrict M1 = (double \*)calloc(M1SIZE, sizeof(double));

    double \*restrict M2 = (double \*)calloc(M2SIZE, sizeof(double));

    double \*restrict M2temp = (double \*)calloc(M2SIZE, sizeof(double));

    gettimeofday(&T1, NULL); // запомнить текущее время T1

    for (i = 0; i < 100; i++)

    { // 100 экспериментов

      // инициализировать начальное значение ГСЧ

        seed = 50;

        srand(seed);

        // Заполнить массив исходных данных размером N

        for (j = 0; j < M1SIZE; j++)

        {

            M1[j] = 1 + rand\_r(&seed) % 360;

            if (j >= M2SIZE)

            {

                continue;

            }

            else

            {

                M2[j] = 360 + rand\_r(&seed) % (10 \* 360 - 361);

            }

        }

        // Решить поставленную задачу, заполнить массив с результатами

        for (j = 0; j < M1SIZE; j++)

        {

            M1[j] = 1.0 / tanh(sqrt(M1[j]));

        }

        // copy m2 array

        for (j = 0; j < M2SIZE; j++)

        {

            M2temp[j] = M2[j];

        }

        // new value in M2

        for (j = 1; j < M2SIZE; j++)

        {

            M2[j] += M2temp[j - 1];

            M2[j] = sqrt(M2[j] \* M\_E);

        }

        // Merge

        for (j = 0; j < M2SIZE; j++)

        {

            M2[j] = M1[j] / M2[j];

        }

        // Отсортировать массив с результатами указанным методом

       gnomeSort(M2, M2SIZE);

        // REDUCE

        double min = 10 \* 360 + 1;

        for (j = 0; j < M2SIZE; j++)

        {

            if (M2[j] < min)

            {

                min = M2[j];

            }

        }

        for (j = 0; j < M2SIZE; j++)

        {

            if ((int)(M2[j] / min) % 2 == 0)

            {

                sum += sin(M2[j]);

            }

        }

    }

    printf("X: %f\n", sum);

    gettimeofday(&T2, NULL); // запомнить текущее время T2

    delta\_ms = (T2.tv\_sec - T1.tv\_sec) \* 1000 +

               (T2.tv\_usec - T1.tv\_usec) / 1000;

    printf("\nN=%d. Milliseconds passed: %ld\n", N, delta\_ms);

    fprintf(resultOfTest, "\n%d %ld\n", N, delta\_ms);

    free(M1);

    free(M2);

    free(M2temp);

    fclose(resultOfTest);

    return 0;

}

void gnomeSort(double \*array, int size)

{

    int index = 0;

    while (index < size)

    {

        if (index == 0)

            index++;

        if (array[index] >= array[index - 1])

            index++;

        else

        {

            int temp = array[index];

            array[index] = array[index - 1];

            array[index - 1] = temp;

            index--;

        }

    }

}

Результаты

Сравнение результатов при N1 и N2

|  |  |  |  |
| --- | --- | --- | --- |
|  | Paralleling | N1 time,ms | N2 time,ms |
| 1 | Without | 20 | 5053 |
| 2 | K=1 | 12 | 5062 |
| 3 | K=2 | 15 | 5086 |
| 4 | K=4 | 25 | 5115 |
| 5 | K=8 | 20 | 5054 |

Таблица 1 - Результаты при N1 и N2

Рисунок 1 – сравнение результатов при N1

Рисунок 2 – сравнение результатов при N2

Сравнение результатов в динамике роста N

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | N | K=0 | K=1 | K=2 | K=4 | K=8 |
| 1 | 153900 | 588 | 598 | 625 | 770 | 610 |
| 2 | 286800 | 1141 | 1155 | 1186 | 1359 | 1182 |
| 3 | 419700 | 1651 | 1643 | 1732 | 1917 | 1686 |
| 4 | 552600 | 2152 | 2158 | 2158 | 2429 | 2162 |
| 5 | 685500 | 2776 | 2660 | 2675 | 2941 | 2652 |
| 6 | 818400 | 3130 | 3184 | 3109 | 3383 | 3124 |
| 7 | 951300 | 3617 | 3645 | 3657 | 3885 | 3633 |
| 8 | 1084200 | 4142 | 4132 | 4116 | 4407 | 4148 |
| 9 | 1217100 | 4634 | 4622 | 4633 | 4893 | 4606 |
| 10 | 1350000 | 5077 | 5125 | 5120 | 5354 | 5093 |

Таблица 2 - Результаты при изменении N

Рисунок 3 – Cравнение результатов при изменяющемся N

Хоть графики и представляют ломанные линии с различными данными, в итоге можно сказать, что время, за которое выполняется программа не значительно отличается от примера, к примеру.

Выводы

Исходя из графиков можно сказать, что автоматическое распараллеливание не дает ожидаемого ускорения выполнения программы. Полученные графики также подтверждают то, что использование автоматического распараллеливания является не лучшим выбором для ускорения выполнения задачи.