ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

Факультет компьютерных наук
Образовательная программа бакалавриата «Программная инженерия»

ПРОЕКТ ПО РАЗРАБОТКЕ МНОГОПОТОЧНОГО ПРИЛОЖЕНИЯ С ИСПОЛЬЗОВАНИЕМ OpenMP

Отчет

Исполнитель
Студентка группы БПИ193(подгруппа 1)
Шевко Марина Николаевна
Вариант 26
30 ноября 2020 г

Москва 2020

СОДЕРЖАНИЕ

1. ВВЕДЕНИЕ

1.1. Постановка задачи на выполнение

Вторая задача об Острове Сокровищ. Шайка пиратов под предводительством Джона Сильвера высадилась на берег Острова Сокровищ. Не смотря на добытую карту старого Флинта, местоположение сокровищ по прежнему остается загадкой, поэтому искать клад приходится практически на ощупь. Так как Сильвер ходит на деревянной ноге, то самому бродить по джунглям ему не с руки. Джон Сильвер поделил остров на участки, а пиратов на небольшие группы. Каждой группе поручается искать клад на нескольких участках, а сам Сильвер ждет на берегу. Группа пиратов, обшарив одну часть острова, переходит к другой, еще необследованной части. Закончив поиски, пираты возвращаются к Сильверу и докладывают о результатах. Требуется создать многопоточное приложение с управляющим потоком, моделирующее действия Сильвера и пиратов. При решении использовать парадигму портфеля задач.

2. ОПИСАНИЕ ИСПОЛЬЗУЕМНЫХ АЛГОРИТМОВ

2.1. Модель взаимодействия

По условию должна использоваться парадигма портфеля задач, что характерно для модели взаимодействия “взаимодействующие равные”. И действительно – наши потоки симулирующие группы пиратов берут невыполненные задачи из портфеля – общей переменной для всех (массив секторов острова).

2.2. Особенности OpenMP

2.1. Для того чтобы установить количество потоков, симулирующим поисковые группы используется

```
// поисковые группы пиратов симулируются количеством созданных потоков
int threadNumber = getNoOfGroups();
omp_set_num_threads(threadNumber);
```

2.2. После этого используем `#pragma omp parallel` и помещаем внутрь основной цикл while с поиском сокровищ. Эта директива указывает на то, что структурный блок внутри должен выполниться в несколько потоков, а именно в количестве, указанном в предыдущем шаге.

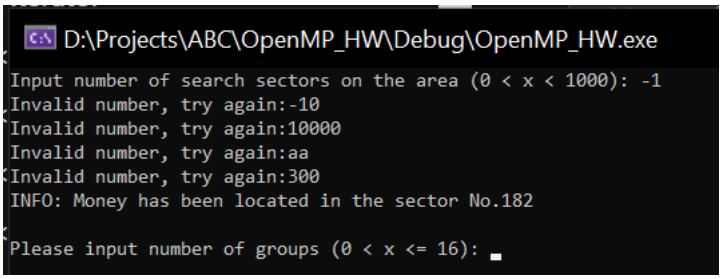
```
#pragma omp parallel
{
    int group = omp_get_thread_num();
    //ищем пока клад не найдн
    while (!moneyFound && currentSector < sectorsNum)
    {
        int i = 0;

        {
            // Свободная группа пиратов берет задачу из портфеля на обыск сектора
            // блокируем для других групп
            #pragma omp critical
            {
                i = currentSector++;
                printf("---> Pirates group %d has got sector No.%d \n", group, i);
            }


            // Имитируем временную задержку при обыске сектора
            double d = uselessFunc(sectors[i].size);  // нагружаем группу объемом работы, зависящим от размера поискового сектора

            if (sectors[i].withMoney)
            {
                // блокируем поток на вывод в консоль
                #pragma omp critical
                {
                    printf("!!!! Pirates group %d has found money in the sector No.%d \n", group, i);
                }
                moneyFound = true;
                groupfound = group;
                sectorfound = i;
            }
            else
            {
                #pragma omp critical
                {
                    printf("   < Pirates group %d has NOT found money in the sector No.%d (%f) \n", group, i, d);
                }
            }

        }
    }
}
```

2.3. С помощью директивны `#pragma omp critical` мы указываем участок кода, который может выполняется только для одного потока, остальные же будут заблокированы. Они ждут своей очереди на выполнение. Но так как в критической

секции мы смещаем указатель на ссылку - на задачу, которая еще не выполнена, то заблокированный поток берет новую задачу. Таким образом, из-за данной критической секции потоки не выполняют весь код каждый, а по сути, распределяют работу между собой – невыполненная задача отдается свободному потоку, что соответствует парадигме портфеля задач. Также критические секции используются для безопасного вывода в консоль.

3. ТЕСТИРОВАНИЕ

3.1.Валидация входных данных



При вводе некорректных чисел и символов просит повторить ввод, пока не будет введен верный ответ.

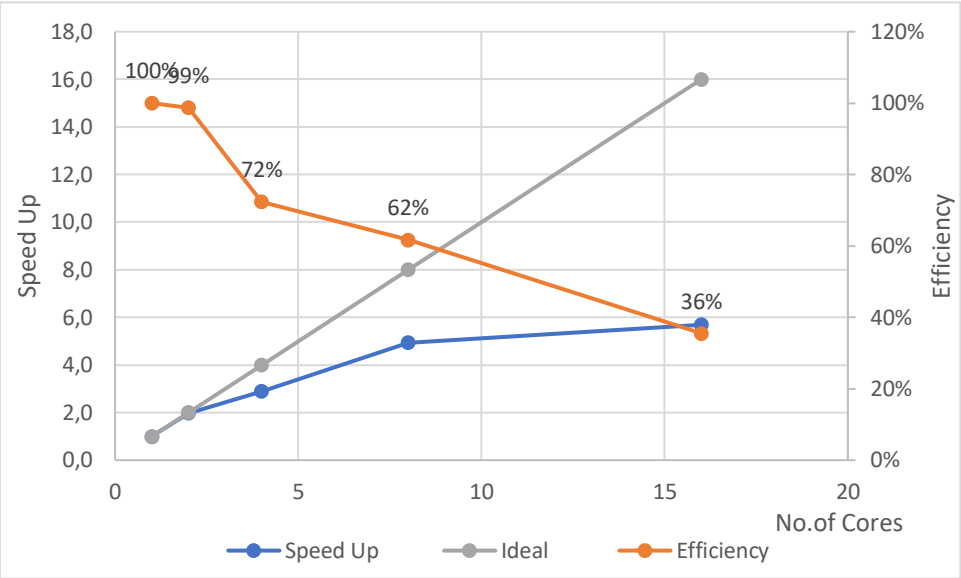
Для имитации различных затрат времени и ресурсов в зависимости от размера сегмента на поиск сокровищ используется функция uselessFunc(). Это позволяет отследить реальную эффективность распараллеливания.

```
// Время- и ресурсно- затратная функция
double uselessFunc(const int size)
{
    double d = 1;
    for (int i = 0; i < size; ++i) {
        for (int j = 0; j < size; ++j)
            d += d * i * j - d;
    }
    return d;
}
```

3.2. На небольших числах

Входные данные: 40 секторов, рандомно выбирается, что в 35 сектора будет находиться клад. Также выбираются 4 поисковые группы (из 16 возможных на данном компьютере). Можно заметить, что группы действительно берет свободные задачи как только закончат с текущей. (Например, 1 группа ничего не нашла во 2 секторе и взяла 5, так как 3 и 4 уже были взяты другими группами). В результатах отображается, что 0-я группа нашла клад в 35 секторе, что соответствует выведенной информации о кладе в начале.

Также была протестирована эта программа с 40 секторами, но с 1, 4, 8 и 16 группами пиратов (потоками) и был составлен график эффективности ускорения в завимисости от количества ядер. (замер временни с помощью `omp_get_wtime()`). По нему видно, что для небольшого числа эффективно лишь ускорение на 4 ядрах, в то время как для 8 и 16 она заметно снижаетя.



```
Input number of search sectors on the area (0 < x < 1000): 40
INFO: Money has been located in the sector No.35

Please input number of groups (0 < x <= 16): 4
---> Pirates group 0 has got sector No.0
   < Pirates group 0 has NOT found money in the sector No.0 (0.000000)
---> Pirates group 0 has got sector No.1
---> Pirates group 1 has got sector No.2
---> Pirates group 3 has got sector No.3
---> Pirates group 2 has got sector No.4
   < Pirates group 1 has NOT found money in the sector No.2 (0.000000)
---> Pirates group 1 has got sector No.5
   < Pirates group 1 has NOT found money in the sector No.5 (0.000000)
---> Pirates group 1 has got sector No.6
   < Pirates group 0 has NOT found money in the sector No.1 (0.000000)
---> Pirates group 0 has got sector No.7
   < Pirates group 2 has NOT found money in the sector No.4 (0.000000)
---> Pirates group 2 has got sector No.8
   < Pirates group 1 has NOT found money in the sector No.6 (0.000000)
---> Pirates group 1 has got sector No.9
   < Pirates group 3 has NOT found money in the sector No.3 (0.000000)
---> Pirates group 3 has got sector No.10
   < Pirates group 3 has NOT found money in the sector No.10 (0.000000)
---> Pirates group 3 has got sector No.11
   < Pirates group 1 has NOT found money in the sector No.9 (0.000000)
---> Pirates group 1 has got sector No.12
   < Pirates group 2 has NOT found money in the sector No.8 (0.000000)
---> Pirates group 2 has got sector No.13
   < Pirates group 0 has NOT found money in the sector No.7 (0.000000)
---> Pirates group 0 has got sector No.14
   < Pirates group 2 has NOT found money in the sector No.13 (0.000000)
---> Pirates group 2 has got sector No.15
   < Pirates group 2 has NOT found money in the sector No.15 (0.000000)
---> Pirates group 2 has got sector No.16
   < Pirates group 0 has NOT found money in the sector No.14 (0.000000)
---> Pirates group 0 has got sector No.17
   < Pirates group 2 has NOT found money in the sector No.16 (0.000000)
---> Pirates group 2 has got sector No.18
   < Pirates group 2 has NOT found money in the sector No.18 (0.000000)
---> Pirates group 2 has got sector No.19
   < Pirates group 2 has NOT found money in the sector No.19 (0.000000)
---> Pirates group 2 has got sector No.20
   < Pirates group 3 has NOT found money in the sector No.11 (0.000000)
---> Pirates group 3 has got sector No.21
   < Pirates group 0 has NOT found money in the sector No.17 (0.000000)
---> Pirates group 0 has got sector No.22
   < Pirates group 0 has NOT found money in the sector No.22 (0.000000)
---> Pirates group 0 has got sector No.23
   < Pirates group 0 has NOT found money in the sector No.23 (0.000000)
---> Pirates group 0 has got sector No.24
   < Pirates group 0 has NOT found money in the sector No.24 (0.000000)
---> Pirates group 0 has got sector No.25
   < Pirates group 1 has NOT found money in the sector No.12 (0.000000)
---> Pirates group 1 has got sector No.26
   < Pirates group 0 has NOT found money in the sector No.25 (0.000000)
---> Pirates group 0 has got sector No.27
   < Pirates group 3 has NOT found money in the sector No.21 (0.000000)
---> Pirates group 3 has got sector No.28
   < Pirates group 1 has NOT found money in the sector No.26 (0.000000)
---> Pirates group 1 has got sector No.29
   < Pirates group 0 has NOT found money in the sector No.27 (0.000000)
---> Pirates group 0 has got sector No.30
   < Pirates group 0 has NOT found money in the sector No.30 (0.000000)
---> Pirates group 0 has got sector No.31
   < Pirates group 3 has NOT found money in the sector No.28 (0.000000)
---> Pirates group 3 has got sector No.32
   < Pirates group 1 has NOT found money in the sector No.29 (0.000000)
---> Pirates group 1 has got sector No.33
   < Pirates group 3 has NOT found money in the sector No.32 (0.000000)
---> Pirates group 3 has got sector No.34
   < Pirates group 3 has NOT found money in the sector No.34 (0.000000)
---> Pirates group 3 has got sector No.35
   < Pirates group 1 has NOT found money in the sector No.33 (0.000000)
---> Pirates group 1 has got sector No.36
   < Pirates group 2 has NOT found money in the sector No.20 (0.000000)
---> Pirates group 2 has got sector No.37
   < Pirates group 0 has NOT found money in the sector No.31 (0.000000)
---> Pirates group 0 has got sector No.38
!!!! Pirates group 3 has found money in the sector No.35
   < Pirates group 0 has NOT found money in the sector No.38 (0.000000)
   < Pirates group 1 has NOT found money in the sector No.36 (0.000000)
   < Pirates group 2 has NOT found money in the sector No.37 (0.000000)

RESULTS: Pirates group 3 has found Money in the sector No.35
        CPU time by using of 4 groups (cores) is 12.702228 sec
```

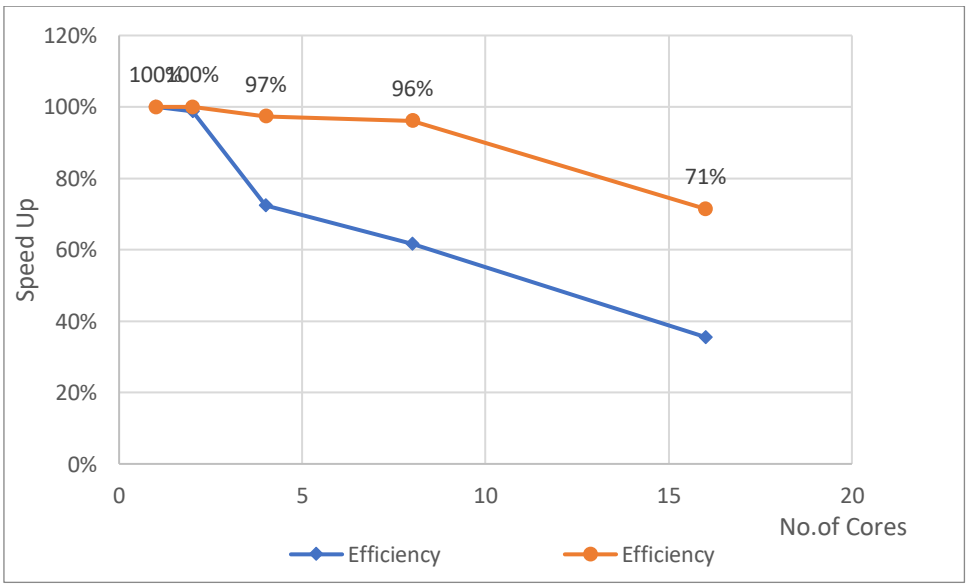
3.3. Для большого числа

Входные данные: 400 секторов, рандомно выбирается, что в 216 сектора будет находиться клад. Также выбираются 16 поисковые группы (из 16 возможных на данном компьютере).

```
Input number of search sectors on the area (0 < x < 1000): 400
INFO: Money has been located in the sector No.216

Please input number of groups (0 < x <= 16):
16
---> Pirates group 9 has got sector No.0
---> Pirates group 0 has got sector No.1
---> Pirates group 1 has got sector No.2
---> Pirates group 7 has got sector No.3
---> Pirates group 12 has got sector No.4
---> Pirates group 13 has got sector No.5
---> Pirates group 8 has got sector No.6
---> Pirates group 6 has got sector No.7
---> Pirates group 4 has got sector No.8
   < Pirates group 9 has NOT found money in the sector No.0 (0.000000)
```

```
---> Pirates group 4 has got sector No.218
!!!! Pirates group 5 has found money in the sector No.216
   < Pirates group 4 has NOT found money in the sector No.218 (0.000000)
   < Pirates group 15 has NOT found money in the sector No.206 (0.000000)
   < Pirates group 1 has NOT found money in the sector No.203 (0.000000)
   < Pirates group 12 has NOT found money in the sector No.202 (0.000000)
   < Pirates group 14 has NOT found money in the sector No.192 (0.000000)
   < Pirates group 9 has NOT found money in the sector No.185 (0.000000)
   < Pirates group 7 has NOT found money in the sector No.198 (0.000000)
   < Pirates group 10 has NOT found money in the sector No.204 (0.000000)
   < Pirates group 8 has NOT found money in the sector No.214 (0.000000)
   < Pirates group 0 has NOT found money in the sector No.208 (0.000000)
   < Pirates group 13 has NOT found money in the sector No.217 (0.000000)
   < Pirates group 11 has NOT found money in the sector No.209 (0.000000)
   < Pirates group 2 has NOT found money in the sector No.207 (0.000000)
   < Pirates group 6 has NOT found money in the sector No.213 (0.000000)
   < Pirates group 3 has NOT found money in the sector No.215 (0.000000)

RESULTS: Pirates group 5 has found Money in the sector No.216
         CPU time by using of 16 groups (cores) is 24.253263 sec
```

Для этой программы также был построен график эффективности ускорения в зависимости от количества потоков. На графике ниже можно сравнить эффективность для маленького числа (35) и большого (216). Видно, что для большего числа распараллеливание на 4 и 8 потоков ускоряет выполнение практически в 4 и 8 раз соответственно. Для маленького числа распараллеливание более чем на 4 ядра не дает такого эффекта.



ТЕКСТ ПРОГРАММЫ

```
#include <iostream>
#include <vector>
#include <omp.h>

using namespace std;

int currentSector = 0;
int sectorsNum;
bool moneyFound = false;
int group = -1;

struct Sector
{
    int size = 0;
    bool withMoney = false;
};

Sector* sectors; // портфель задач

// Время- и ресурсно- затратная функция
double uselessFunc(const int size)
{
    double d = 1;
    for (int i = 0; i < size; ++i) {
        for (int j = 0; j < size; ++j)
            d += d * i * j - d;
    }
    return d;
}

//Запрос числа секторов от пользователя
int getNoOfSectors()
{
    cout << "Input number of search sectors on the area (0 < x < 1000): ";
    int num;
    cin >> num;
    while (num <= 0 or num > 1000)
    {
        // not a numeric character, probably
        if (cin.fail())
        {
            cin.clear();
            string c;
            cin >> c;
        }
        cout << "Invalid number, try again:";
        cin >> num;
    }
    return num;
}

//Запрос числа групп от пользователя
int getNoOfGroups()
{
    // Кол-во одновременных потоков, поддерживаемых системой
    unsigned int n = omp_get_max_threads();

    printf("Please input number of groups (0 < x <= %d): ", n);

    int num;
    cin >> num;
    while (num <= 0 or num > n)
    {
        // может быть не числом
        if (cin.fail())
        {
            cin.clear();
```

```
            string c;
            cin >> c;
        }
        cout << "Invalid number, try again:";
        cin >> num;
    }
    return num;
}

int main()
{

    //получаем количество секторов
    int sectorsNum = getNoOfSectors();

    //инициалищируем секторы острова
    sectors = new Sector[sectorsNum];

    // генерируем сектора острова - у каждого свой размер.
    for (int i = 0; i < sectorsNum; i++)
    {
        sectors[i] = Sector{ rand(), false };
    }

    // рандомно выбираем сектор с кладом
    int sectorId = (double)rand() / RAND_MAX * sectorsNum;
    sectors[sectorId].withMoney = true;

    // выводим информацию о номере выбранного сектора с кладом
    printf("INFO: Money has been located in the sector No.%d \n\n", sectorId);


    double itime, ftime, exec_time;
    itime = omp_get_wtime();


    // поисковые группы пиратов симулируются количеством созданных потоков
    int threadNumber = getNoOfGroups();
    omp_set_num_threads(threadNumber);
    int sectorfound = -1;
    int groupfound = -1;

    #pragma omp parallel
    {
        int group = omp_get_thread_num();
        //ищем пока клад не найдн
        while (!moneyFound && currentSector < sectorsNum)
        {
            int i = 0;

            {
                // Свободная группа пиратов берет задачу из портфеля на обыск сектора
                // блокируем для других групп
                #pragma omp critical
                {
                    i = currentSector++;
                    printf("---> Pirates group %d has got sector No.%d \n", group, i);
                }


                // Имитируем временную задержку при обыске сектора
                double d = uselessFunc(sectors[i].size);  // нагружаем группу объемом
работы, зависящим от размера поискового сектора

                if (sectors[i].withMoney)
                {
                    // блокируем поток на вывод в консоль
                    #pragma omp critical
                    {
                        printf("!!!! Pirates group %d has found money in the sector
No.%d \n", group, i);
```

```
                    }
                    moneyFound = true;
                    groupfound = group;
                    sectorfound = i;
                }
                else
                {
                    #pragma omp critical
                    {
                        printf("   < Pirates group %d has NOT found money in the
sector No.%d (%f) \n", group, i, d);
                    }
                }

            }
        }
    }

    ftime = omp_get_wtime();
    exec_time = ftime - itime;
    printf("\nRESULTS: Pirates group %d has found Money in the sector No.%d \n",
groupfound, sectorfound);
    printf("         CPU time by using of %d groups (cores) is %f sec\n",
threadNumber, exec_time);

    return 0;
}
```

СПИСОК ЛИТЕРАТУРЫ

1) Модели взаимодействия процессов. Управляющий – рабочие (распределенный портфель задач) [Электронный ресурс] // Режим доступа: https://l.wzm.me/_coder/custom/parallel.programming/009.htm , свободный (Дата обращения: 30.11.2020)

2) Потоки, блокировки и условные переменные в C++ [Электронный ресурс] // Режим доступа: https://habr.com/ru/post/182610/ , свободный (Дата обращения: 30.11.2020)

3) Парадигмы параллельного программирования [Электронный ресурс] // Режим доступа: https://pro-prof.com/forums/topic/parallel-programming-paradigms, свободный (Дата обращения: 30.11.2020)

4) Методы и алгоритмы параллельных вычислений [Электронный ресурс] // Режим доступа: openmp.pdf (petrsu.ru) , свободный (Дата обращения: 30.11.2020)

5) Параллельное программирование на OpenMP вычислений [Электронный ресурс] // Режим доступа: openmp.pdf (nsu.ru) , свободный (Дата обращения: 30.11.2020)