Домашнее задание №3.

Вторая задача об Острове Сокровищ.

Штырба Владислав Анатольевич

НИУ ВШЭ

Факультет Компьютерных наук

Группа БПИ197

Текст задания

26. Шайка пиратов под предводительством Джона Сильвера высадилась на берег Острова Сокровищ. Несмотря на добытую карту старого Флинта, местоположение сокровищ по- 8 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.

Применяемые расчётные методы

В программе была использована парадигма портфеля задач (ссылка http://window.edu.ru/catalog/pdf2txt/971/67971/41350?p_page=20)

Реализация параллельных вычислений возможна с использованием так называемого портфеля задач. Задача здесь представляет независимую единицу работы. Задачи помещаются в “портфель”, разделяемый всеми процессами,

работающими по программе

Парадигма портфеля задач имеет следующие положительные черты:

1) она весьма проста, так как для работы достаточно соблюсти всего лишь несколько требований:

– нужно дать представление задач;

– определить портфель (набор задач);

– дать программу выполнения задачи;

– определить критерий ее окончания;

2) программы с использованием портфеля задач легко масштабируются простым изменением числа процессов (при этом нельзя забывать, что производительность может не измениться, если число процессов больше числа задач);

3) упрощается балансировка нагрузки: освободившиеся процессы берут на себя решение новых задач из портфеля (пока задач в два или три раза больше, чем процессов, загрузка процессов окажется приблизительно одинаковой).

Исходный код

```
#include <iostream>
#include <thread>
#include <math.h>
#include <chrono>
#include <cstdlib>
#include <ctime>
#include <string>
#include <vector>
using namespace std;
using namespace std::chrono;

int result;

void crewFind(bool* islandPart, int size, int islandI) {
    for (int i = 0; i < size; i++)
```

```
        if (islandPart[i])
            result = islandI * size + i;
}

void find(int crewN, bool* island, int size) {
    auto startTime = high_resolution_clock::now();

    cout << "# of threads = " << crewN << endl;
    // thread is a crew group
    int tileCrN = ceil(size * 1.0 / crewN);

    bool** islandParts = new bool* [crewN];
    for (int i = 0; i < crewN; i++)
        islandParts[i] = new bool[tileCrN];

    int begin = 0;
    int end = tileCrN;

    for (int i = 0; i < crewN; i++) {
        for (int j = begin; j < end; j++) {
            if (j >= size)
                islandParts[i][j - begin] = false;
            else islandParts[i][j - begin] = island[j];
        }
        begin = end;
        end = begin + tileCrN;
    }

    for (int i = 0; i < crewN; i++) {
        for (int j = 0; j < tileCrN; j++)
            cout << (islandParts[i][j] ? "1 " : "0 ");
        cout << endl;
    }
    int i = 0;
    thread* crews = new thread[crewN];
    while (true)
    {
        auto t = thread(crewFind, islandParts[i], tileCrN, i);
        t.join();
        if (i == crewN - 1)
        {
            break;
        }
        i++;
    }
   /* for (int i = 0; i < crewN; i++)
        crews[i] = thread(crewFind, islandParts[i], tileCrN, i);
    for (int i = 0; i < crewN; i++)
        crews[i].join();*/

    auto stopTime = high_resolution_clock::now();
    auto duration = duration_cast<microseconds>(stopTime - startTime);

    cout << "Found treasure at " << result << endl;
    cout << "Spent time: \t" << duration.count() * 1.0 / 1000000 << endl;

    for (int i = 0; i < crewN; i++)
        delete[] islandParts[i];
    delete[] islandParts;
}

void print(bool* island, int size) {
    for (int i = 0; i < size; i++)
        cout << to_string(i) << ") " << (island[i] ? "1" : "0") << endl;
    cout <<"\n";
```

```
}

bool check(string line) {
    for (int i = 0; i < line.length(); i++)
        if (line[i] < '0' || line[i] > '9')
            return false;
    return true;
}

int main()
{
    srand(time(NULL));
    string line;
    cout << "Enter the size of the island: ";
    cin >> line;
    if (!check(line)) {
        cout << "Enter a positive integer pls" << endl;
        return 0;
    }
    int islandSize = stoi(line);
    bool* island = new bool[islandSize] {};

    island[rand() % islandSize] = true;

    print(island, islandSize);

    for (int i = 1; i < 9; i++) {
        find(i, island, islandSize);
        cout << "\n";
    }

    delete[] island;
    return 0;
}
```

Тест программы

Тест с неверными введёнными данными:

test1_wrong_input.png

Тест с верным вводом:

test2.png