Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

**Лабораторная работа №2 по курсу**

**«Операционные системы»**

**Управление потоками в ОС**

Студент: Эсмедляев Федор Романович

Группа: М8О–212Б–22

Вариант: 5

Преподаватель: Соколов Андрей Алексеевич

Оценка: \_\_\_\_\_\_\_\_\_\_

Дата: \_\_\_\_\_\_\_\_\_\_

Подпись: \_\_\_\_\_\_\_\_\_\_

Москва, 2023.

**Постановка задачи**

## Цель работы

## Целью является приобретение практических навыков в:  Управление потоками в ОС  Обеспечение синхронизации между потоками

## Задание

Составить программу на языке Си, обрабатывающую данные в многопоточном режиме. При обработки использовать стандартные средства создания потоков операционной системы (Windows/Unix). Ограничение максимального количества потоков, работающих в один момент времени, должно быть задано ключом запуска вашей программы.

Так же необходимо уметь продемонстрировать количество потоков, используемое вашей программой с помощью стандартных средств операционной системы.

В отчете привести исследование зависимости ускорения и эффективности алгоритма от входных данных и количества потоков. Получившиеся результаты необходимо объяснить.

Вариант 5(был заменен): Быстрая сортировка (Хоара)

**Общие сведения о программе**

Программа компилируется из файла sort.cpp. В программе используются следующие системные вызовы:

1. **pthread\_create** - создает новый поток
2. **pthread\_join** - присоединяется к потоку
3. **pthread\_exit** - завершает работу потока с указанным кодом

Программа реализует сортировку quicksort. Данная сортировка реализуется в основном через 2 рекурсии, но я сделал реализацию без использования рекурсий, поскольку при их использовании пришлось бы делать еще 2 структуры, что крайне муторно. Весь мой алгоритм заключается в разделении массива на подмассивы, которые получаются путем деления изначального массива на количество потоков, а далее сортируем отдельно каждую часть, все выполняется параллельно, далее у нас будет просто массив, с отсортированными подмассивами, просто берем и соединяем их в 1 итоговый массив.

**Общий метод и алгоритм решения**.

Для реализации задачи необходимо:

1. Написать quicksort + partitions
2. Написать mergeSortedArray
3. Модифицировать quicksort для работы с многопоточностью
4. Произвести анализ зависимости времени работы от количества потоков.

**Основные файлы программы**

#include <bits/stdc++.h>

#include <chrono>

#include <pthread.h>

using namespace std;

using ll = long long;

const int MAX\_THREADS = 7;

typedef struct args\_ins

{

ll\* array;

ll start;

ll end;

} args\_ins\_t;

ll partition(args\_ins\_t args)

{

ll high = args.end;

ll low = args.start;

ll pivot = args.array[high];

ll i = (low - 1);

for (int j = low; j <= high - 1; j++)

{

if (args.array[j] <= pivot)

{

i++;

swap(args.array[i], args.array[j]);

}

}

swap(args.array[i + 1], args.array[high]);

return (i + 1);

}

void\* quickSort(void\* input)

{

args\_ins\_t\* ar = (args\_ins\_t\*)input;

ll\* arr = ar->array;

ll start = ar->start;

ll end = ar->end;

stack<pair<ll, ll>> stack;

stack.push(make\_pair(start, end));

vector<ll> a;

while (!stack.empty())

{

for (int i = start; i < end; i++)

{

a.push\_back(ar->array[i]); // Копируем элементы из ar в arr

}

ll low = stack.top().first;

ll high = stack.top().second;

stack.pop();

if (low < high)

{

args\_ins\_t arg;

arg.array = arr;

arg.end = high;

arg.start = low;

ll pivot\_index = partition(arg);

if (pivot\_index > low)

stack.push(make\_pair(low, pivot\_index - 1));

if (pivot\_index < high)

stack.push(make\_pair(pivot\_index + 1, high));

}

}

}

vector<ll> mergeSortedArrays(const vector<ll>& arr1, const vector<ll>& arr2)

{

vector<ll> merged;

ll i = 0;

ll j = 0;

while (i < arr1.size() and j < arr2.size())

{

if (arr1[i] < arr2[j])

{

merged.push\_back(arr1[i]);

i++;

}

else

{

merged.push\_back(arr2[j]);

j++;

}

}

// Добавляем оставшиеся элементы (если они есть) из обоих массивов

while (i < arr1.size())

{

merged.push\_back(arr1[i]);

i++;

}

while (j < arr2.size())

{

merged.push\_back(arr2[j]);

j++;

}

return merged;

}

void potok(vector<ll>& arr, ll low, ll high)

{

pthread\_t threads[MAX\_THREADS];

ll partitions[MAX\_THREADS + 1];

partitions[0] = low;

for (int i = 0; i < MAX\_THREADS; i++)

{

partitions[i] = i \* arr.size() / MAX\_THREADS;

}

partitions[MAX\_THREADS] = high;

for (int i = 0; i < MAX\_THREADS; i++)

{

ll new\_low = partitions[i];

ll new\_high;

if (i == MAX\_THREADS - 1)

new\_high = high;

else

new\_high = partitions[i + 1] - 1;

if (new\_low < new\_high)

{

args\_ins\_t\* data = (args\_ins\_t\*)malloc(sizeof(args\_ins\_t));

data->array = arr.data(); // Передача указателя на массив

data->start = new\_low; // Установка начального индекса

data->end = new\_high; // Установка конечного индекса

pthread\_create(&threads[i], NULL, quickSort, data);

}

}

for (int i = 0; i < MAX\_THREADS; i++)

{

pthread\_join(threads[i], NULL); // Ожидание завершения потоков

}

// Вывод для отладки (может быть убран)

// for (int j = 0; j <= 32; j++)

// cout << arr[j] << " ";

// cout << '\n';

vector<ll> m;

ll pos = 0;

for (int j = partitions[0]; j < partitions[1]; j++)

{

m.push\_back(arr[j]);

}

for (int i = 1; i < MAX\_THREADS; i++)

{

vector<ll> mas1;

for (int j = partitions[i]; j < partitions[i+1]; j++)

{

mas1.push\_back(arr[j]);

}

m = mergeSortedArrays(m, mas1);

}

for (int i = 0; i < m.size(); i++)

{

cout << m[i] << " ";

}

cout << '\n';

}

int main()

{

ll n;

cin >> n;

vector<ll> arr(n);

for (int i = 0; i < n; i++)

{

cin >> arr[i];

}

auto start = chrono::high\_resolution\_clock::now();

// Создание потоков

potok(arr, 0, n - 1);

auto end = chrono::high\_resolution\_clock::now();

// Ожидание завершения потоков

auto duration = chrono::duration\_cast<chrono::microseconds>(end - start);

cout << duration.count() << '\n';

return 0;

}

**Пример работы**

fedorubuntu@fedorubuntu-ZenBook-UX325JA-UX325JA:~/Документы/OS/lab2/build$ "/home/fedorubuntu/Документы/OS/lab2/build/sort"

100

7 49 73 58 30 72 44 78 23 9 40 65 92 42 87 3 27 29 40 12 3 69 9 57 60 33 99 78 16 35 97 26 12 67 10 33 79 49 79 21 67 72 93 36 85 45 28 91 94 57 1 53 8 44 68 90 24 96 30 3 22 66 49 24 1 53 77 8 28 33 98 81 35 13 65 14 63 36 25 69 15 94 29 1 17 95 5 4 51 98 88 23 5 82 52 66 16 37 38 44

1 1 1 3 3 3 4 5 5 7 8 8 9 9 10 12 12 13 14 15 16 16 17 21 22 23 23 24 24 25 26 27 28 28 29 29 30 30 33 33 33 35 35 36 36 37 38 40 40 42 44 44 44 45 49 49 49 51 52 53 53 57 57 58 60 63 65 65 66 66 67 67 68 69 69 72 72 73 77 78 78 79 79 81 82 85 87 88 90 91 92 93 94 94 95 96 97 98 98

828

fedorubuntu@fedorubuntu-ZenBook-UX325JA-UX325JA:~/Документы/OS/lab2/build$ "/home/fedorubuntu/Документы/OS/lab2/build/sort"

100

7 49 73 58 30 72 44 78 23 9 40 65 92 42 87 3 27 29 40 12 3 69 9 57 60 33 99 78 16 35 97 26 12 67 10 33 79 49 79 21 67 72 93 36 85 45 28 91 94 57 1 53 8 44 68 90 24 96 30 3 22 66 49 24 1 53 77 8 28 33 98 81 35 13 65 14 63 36 25 69 15 94 29 1 17 95 5 4 51 98 88 23 5 82 52 66 16 37 38 44

1 1 1 3 3 3 4 5 5 7 8 8 9 9 10 12 12 13 14 15 16 16 17 21 22 23 23 24 24 25 26 27 28 28 29 29 30 30 33 33 33 35 35 36 36 37 38 40 40 42 44 44 44 45 49 49 49 51 52 53 53 57 57 58 60 63 65 65 66 66 67 67 68 69 69 72 72 73 77 78 78 79 79 81 82 85 87 88 90 91 92 93 94 94 95 96 97 98 99

643

fedorubuntu@fedorubuntu-ZenBook-UX325JA-UX325JA:~/Документы/OS/lab2/build$ "/home/fedorubuntu/Документы/OS/lab2/build/sort"

100

7 49 73 58 30 72 44 78 23 9 40 65 92 42 87 3 27 29 40 12 3 69 9 57 60 33 99 78 16 35 97 26 12 67 10 33 79 49 79 21 67 72 93 36 85 45 28 91 94 57 1 53 8 44 68 90 24 96 30 3 22 66 49 24 1 53 77 8 28 33 98 81 35 13 65 14 63 36 25 69 15 94 29 1 17 95 5 4 51 98 88 23 5 82 52 66 16 37 38 44

1 1 1 3 3 3 4 5 5 7 8 8 9 9 10 12 12 13 14 15 16 16 17 21 22 23 23 24 24 25 26 27 28 28 29 29 30 30 33 33 33 35 35 36 36 37 38 40 40 42 44 44 44 45 49 49 49 51 52 53 53 57 57 58 60 63 65 65 66 66 67 67 68 69 69 72 72 73 77 78 78 79 79 81 82 85 87 88 90 91 92 93 94 94 95 96 97 98 99

854

**Таблица зависимости времени работы от количества потоков:**

|  |  |
| --- | --- |
| **Количество потоков** | **Время работы, микросекунды** |
| **1** | **828** |
| **2** | **672** |
| **3** | **610** |
| **4** | **643** |
| **5** | **725** |
| **6** | **857** |
| **7** | **854** |

**Вывод**

Я научился разделять программу на многопоточную работу. Возникли сложности при попытке реализовать, когда у меня была рекурсия, позже когда я уже убрал рекурсию, были проблемы с отладкой, поскольку не давало войти внутрь работы pthread\_create, поэтому пришлось иметь вторую программу, где я проверял работу функций. Остальное сильных проблем не вызвало, только реализация затратила время, логически все понятно.5