BÁO CÁO THỰC HÀNH

Môn học: Phân tích và thiết kế thuật toán

GVHD: Nguyễn Thanh Sơn

Ngày thực hiện: 9/03/2023

Ngày nộp báo cáo: 11/03/2023

1. THÔNG TIN CHUNG:

(Liệt kê tất cả các thành viên trong nhóm)

Lớp: CS112.N21.KHTN

STT	Họ và tên	MSSV	Email
1	Trần Xuân Minh	21520352	21520352@gm.uit.edu.vn
2	Nguyễn Quốc Trường	21521604	21521604@gm.uit.edu.vn

2. ĐÁNH GIÁ KHÁC:

Nội dung	Kết quả
Tổng thời gian thực hiện bài thực hành trung bình	2 ngày
Link Video thực hiện *(nếu có)*	
Ý kiến *(nếu có)* + Khó khăn + Đề xuất …	
Điểm tự đánh giá	10

BÁO CÁO CHI TIẾT

I. Hãy lựa chọn CTDL và thiết kế thuật toán song song để thực hiện sắp xếp dãy tăng dần theo thuật toán merge sort.

- Cấu trúc dữ liệu được lựa chọn cho thuật toán merge sort là mảng (array) vì nó là cấu trúc dữ liệu đơn giản và dễ dàng truy cập phần tử.

- Thuật toán merge sort là một thuật toán phân chia và trị (divide and conquer), trong đó, ta chia dãy cần sắp xếp thành các đoạn con độc lập, tiếp đó sắp xếp và trộn các đoạn con này để tạo ra một dãy đã được sắp xếp. Quá trình trộn (merge) được thực hiện bằng cách so sánh và đẩy các phần tử từ các đoạn con vào một mảng phụ, sau đó ghi đè lại vào mảng ban đầu.

Để thực hiện thuật toán merge sort song song, ta có thể sử dụng phương pháp chia tách dữ liệu và sắp xếp các đoạn con trên các luồng (thread) riêng biệt, sau đó trộn các kết quả lại với nhau. Có thể sử dụng kỹ thuật multi-threading hoặc parallel computing để tăng tốc độ thực thi của thuật toán.

- Thiết kế thuật toán merge sort song song bằng cách sử dụng multi-threading:

+ Chia dữ liệu thành các đoạn con độc lập (tương tự như thuật toán merge sort thông thường).
+ Tạo ra một số lượng các luồng tương đương với số lượng các đoạn con.
+ Chạy mỗi luồng trên một đoạn con, sử dụng thuật toán merge sort thông thường để sắp xếp đoạn con đó.
+ Trộn các kết quả từ các luồng lại với nhau bằng cách sử dụng thuật toán merge sort thông thường.

- Code cài đặt thuật toán merge sort song song bằng Java sử dụng multi-threading:

```
import java.util.Arrays;
import java.util.Scanner;
import java.util.concurrent.*;

public class MergeSortParallel {
    public static void mergeSort(int[] arr) {
        if (arr.length > 1) {
            int mid = arr.length / 2;
            int[] leftArr = Arrays.copyOfRange(arr, 0, mid);
            int[] rightArr = Arrays.copyOfRange(arr, mid, arr.length);

            // Tạo ra một số lượng các luồng tương đương với số lượng các đoạn
con.
            ExecutorService executorService = Executors.newFixedThreadPool(2);
            Future<int[]> leftResult = executorService.submit(() -> {
                // Sắp xếp đoạn con bên trái bằng thuật toán merge sort.
```

```
                mergeSort(leftArr);
                return leftArr;
            });
            Future<int[]> rightResult = executorService.submit(() -> {
                // Sắp xếp đoạn con bên phải bằng thuật toán merge sort.
                mergeSort(rightArr);
                return rightArr;
            });

            try {
                // Lấy kết quả từ các luồng và trộn lại.
                int[] left = leftResult.get();
                int[] right = rightResult.get();
                merge(arr, left, right);
            } catch (InterruptedException | ExecutionException e) {
                e.printStackTrace();
            } finally {
                executorService.shutdown();
            }
        }
    }

    public static void merge(int[] arr, int[] leftArr, int[] rightArr) {
        int i = 0, j = 0, k = 0;
        while (i < leftArr.length && j < rightArr.length) {
            if (leftArr[i] <= rightArr[j]) {
                arr[k++] = leftArr[i++];
            } else {
                arr[k++] = rightArr[j++];
            }
        }

        while (i < leftArr.length) {
            arr[k++] = leftArr[i++];
        }

        while (j < rightArr.length) {
            arr[k++] = rightArr[j++];
        }
    }

    public static void main(String[] args) {
        // int[] arr = {6, 5, 3, 1, 8, 7, 2, 4};
        Scanner scanner = new Scanner(System.in);

        // Nhập số phần tử của mảng
        System.out.print("Nhập số phần tử của mảng: ");
        int n = scanner.nextInt();

        // Khởi tạo mảng
        int[] arr = new int[n];

        // Nhập các phần tử của mảng
        for (int i = 0; i < n; i++) {
            System.out.print("Nhập phần tử thứ " + i + ": ");
            arr[i] = scanner.nextInt();
        }
        // In ra các phần tử của mảng
        System.out.println("Mảng vừa nhập: ");
        System.out.println(Arrays.toString(arr));
```

```
        System.out.println("----------");

        mergeSort(arr);

        // In ra các phần tử của mảng sau khi sắp xếp
        System.out.println("Mảng sau khi sắp xếp: ");
        System.out.println(Arrays.toString(arr));
    }
}
```

Trong ví dụ này, chúng ta sử dụng ExecutorService để tạo ra một số lượng các luồng tương đương với số lượng các đoạn con. Mỗi luồng được chạy trên một đoạn con riêng biệt, sử dụng thuật toán merge sort thông thường. Sau đó, chúng ta sử dụng phương thức get() để lấy kết quả từ các luồng và trộn lại bằng phương thức merge(). Cuối cùng, chúng ta in ra mảng đã được sắp xếp.

II. Link github project:

https://github.com/TxmMinh/CS112.N21.KHTN/tree/main/Parallel-Algorithm