Operating Systems (327107)

Assignment No. 03

Multithreading Merge Sort

Name: Hassan Ahmed Khan (200901086)

Dept/Batch: BSCS - 01 Section: B

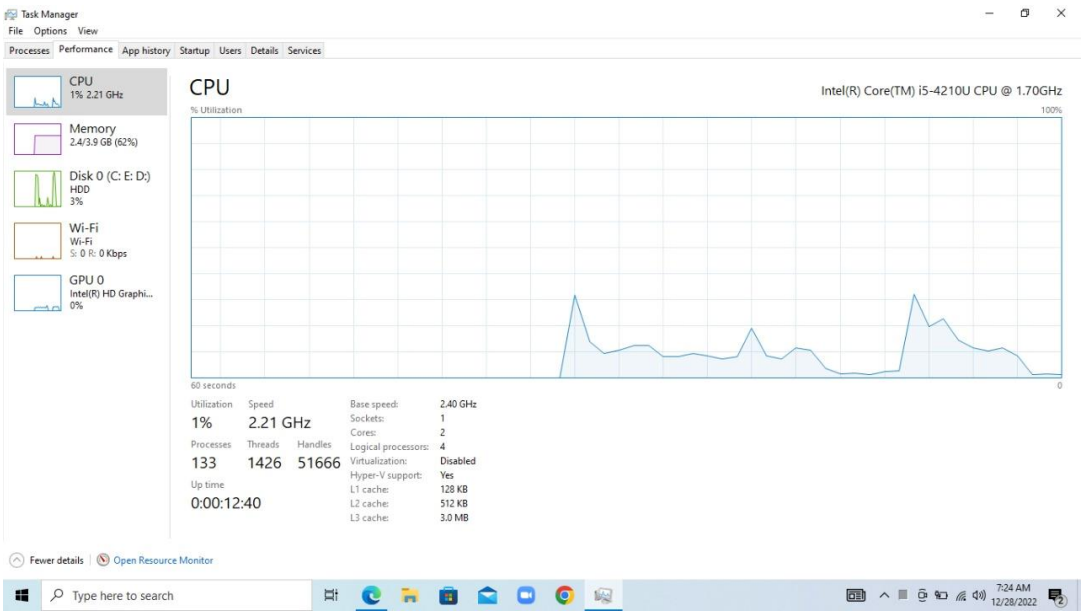
Lecturer: Mam Asia Aman

Date: 28.12.2022

Merge Sort using Multi-Threading

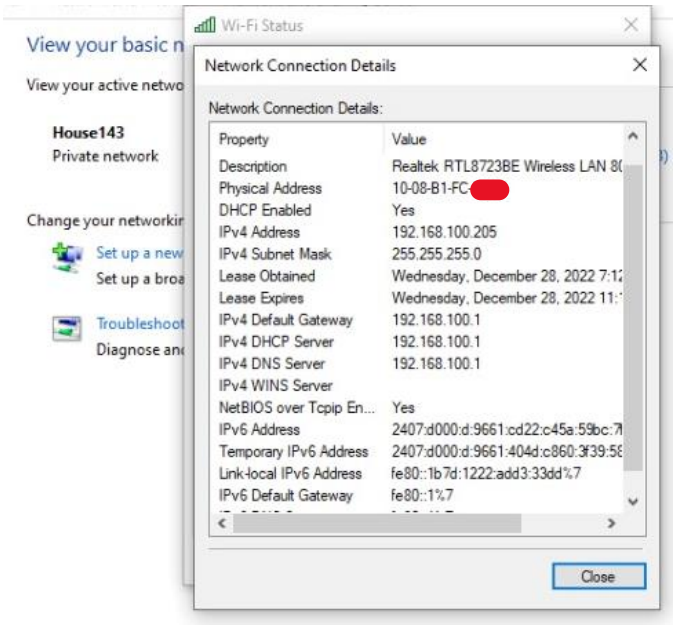
Processor Cores

My laptop's processor has 2 cores, thus I have created 2 threads in the multithreaded merge sort code.



MAC Address

Every device has a unique MAC address, and its knowledge can lead to data leak by hackers, therefore I have hide last 4 digits of the physical address.



Code

```
#include <iostream>
#include <pthread.h>
#include <time.h>

#define THREAD_MAX 2
#define MAX 10

using namespace std;

int a[MAX];
int part = 0;

// Function to merge elements.
void merge(int low, int mid, int high)
{
      //int *a;
      int i, j, k, temp[high - low + 1];
      i = low;
      k = 0;
      j = mid + 1;

      // Merge the two parts into temp[].
      while (i <= mid && j <= high)
      {
            if (a[i] < a[j])
            {
                  temp[k] = a[i];
                  k++;
                  i++;
            }

            else
            {
                  temp[k] = a[j];
                  k++;
                  j++;
            }
      }

      // Insert all the remaining values from i to mid into temp[].
      while (i <= mid)
      {
            temp[k] = a[i];
            k++;
            i++;
      }

      // Insert all the remaining values from j to high into temp[].
      while (j <= high)
```

```
	{
		temp[k] = a[j];
		k++;
		j++;
	}

	// Assign sorted data stored in temp[] to a[].
	for (i = low; i <= high; i++)
	{
		a[i] = temp[i - low];
	}
}

// Function for merge sort.
void merge_sort(int low, int high)
{
	int mid;
	//mid = (low + high) / 2;
	mid = low + (high - low) / 2;

	if (low < high)
	{
		merge_sort(low, mid);
		merge_sort(mid + 1, high);
		merge(low, mid, high);
	}

}

// Thread function for multi-threading.
void* merge_sort(void* arg)
{
	int thread_part = part++;

	int low = thread_part * (MAX / 2);
	int high = (thread_part + 1) * (MAX / 2) - 1;

	int mid = low + (high - low) / 2;

	if (low < high)
	{
		merge_sort(low, mid);
		merge_sort(mid + 1, high);
		merge(low, mid, high);
	}

	return 0;
}

int main()
{
```

```
	int size, i;

	cout << "Enter the size of array: ";
	cin >> size;

	int arr[size];

	cout << "Enter the elements of array: ";
	for (i = 0; i < size; i++)
	{
		cin >> arr[i];
	}

	// t1 and t2 are used for calculating time.
	clock_t t1, t2;
	t1 = clock();
	pthread_t threads[THREAD_MAX];

	for (int i = 0; i < THREAD_MAX; i++)
	{
		pthread_create(&threads[i], NULL, merge_sort, (void*)NULL);
	}

	for (int i = 0; i < 2; i++)
	{
		pthread_join(threads[i], NULL);
	}


	// Merging the final parts.
	merge(0, (size / 2 - 1) / 2, size / 2 - 1);
	merge(size / 2, size / 2 + (size - 1 - size / 2) / 2, size - 1);
	merge(0, (size - 1) / 2, size - 1);


	t2 = clock();

	cout << "\nThe sorted array is: ";
	for (i = 0; i < size; i++)
	{
		cout << " " << arr[i];
	}

	cout << "\nTime taken: " << (t2 - t1) / (double)CLOCKS_PER_SEC <<
endl;

	return 0;
}
```

Reference Link

https://www.geeksforgeeks.org/merge-sort/?ref=lbp