Multicore proj3-prob1

subject	multicore
professor	손봉수 교수님
date	2023.05.20
department	소프트웨어대학 소프트웨어학부
name	20200641 임수현

Environment

CPU : 11th Gen Intel Core(TM) i7-1195G7//Quard Core
Core : 4
RAM : 16GB
System type : 64 bits
OS : window
Run in WSL

Source Code

```
#include <stdio.h>
#include <stdlib.h>
#define _CRT_SECURE_NO_WARNINGS
#include <stdbool.h>
#include <omp.h>
#include <string.h>
bool isprime(int x){
    int i;
    if (x<=1) return false;
    for (i=2;i<x;i++) {
        if (x%i == 0) return false;
    }
    return true;
}

int main(int argc, char *argv[]) {
        int schedul_type = atoi(argv[1]);
        int thread_num=atoi(argv[2]);
    int count=0;
    double start_time, end_time;

    switch (schedul_type){
    case 1:
        start_time=omp_get_wtime();
        omp_set_num_threads(thread_num);

            #pragma omp parallel for reduction(+:count) schedule(static)
            for (int i=1;i<=200000;i++){
                if (isprime(i)){
                    count++;
                }
            }

        end_time=omp_get_wtime();
        break;
    case 2:
        start_time=omp_get_wtime();
        omp_set_num_threads(thread_num);

            #pragma omp parallel for reduction(+:count) schedule(dynamic)
            for (int i=1;i<=200000;i++){
                if (isprime(i)){
                    count++;
                }
            }
```

```
            end_time=omp_get_wtime();
            break;
        case 3:
            start_time=omp_get_wtime();
            omp_set_num_threads(thread_num);

                #pragma omp parallel for reduction(+:count) schedule(static,10)
                for (int i=1;i<=200000;i++){
                    if (isprime(i)){
                        count++;
                    }
                }

            end_time=omp_get_wtime();
            break;
        case 4:
            start_time=omp_get_wtime();
            omp_set_num_threads(thread_num);

                #pragma omp parallel for reduction(+:count)
schedule(dynamic,10)
                for (int i=1;i<=200000;i++){
                    if (isprime(i)){
                        count++;
                    }
                }

            end_time=omp_get_wtime();
            break;
        default:
            printf("You put wrong number. (input num : 1~4)\n");
            break;
        }



        printf("Number of prime numbers: %d\n", count);
        printf("Execution time: %.2f msec\n", (end_time - start_time)*1000);




}
```

code explain

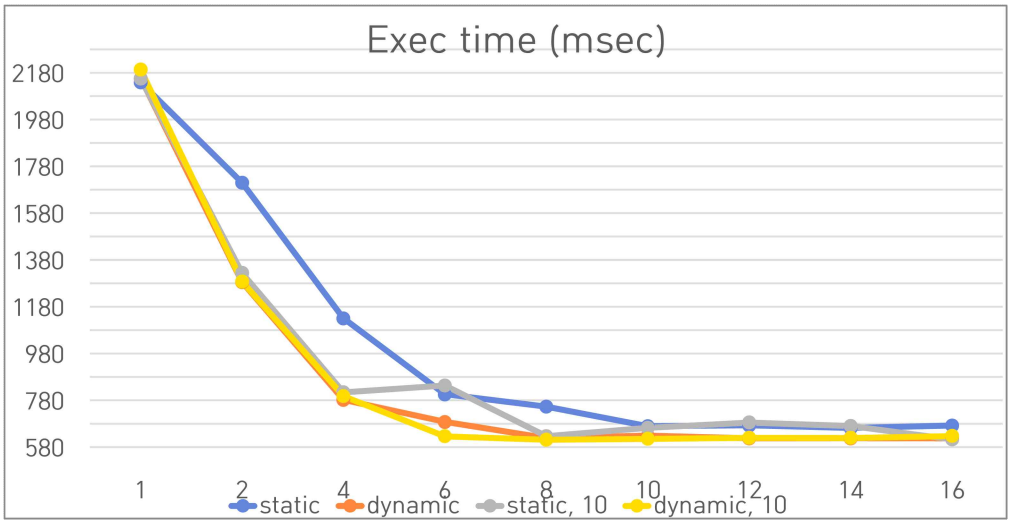
It was executed in four cases using the switch statement. And through reduction (+:count), the count value became a 'public' value.
isprime() was written in the same way as the previous task code.

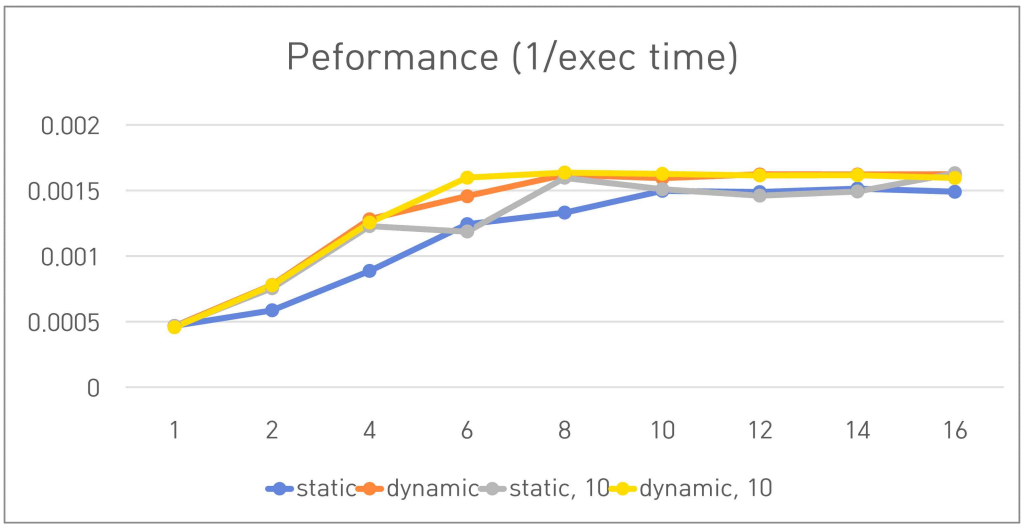
Result

Exec time (msec)	chunk size	1	2	4	6	8	10	12	14	16
static	default	2138.9	1709.7	1129.7	805.13	751.79	668.32	672.02	660.74	670.99
dynamic	default	2155.3	1284.2	781.01	686.91	617.89	628.12	616.5	616.96	616.48
static	10	2154.8	1323.8	813.88	843.2	626.51	662.24	684.67	670.06	612.85
dynamic	10	2193.5	1287.7	797.45	625.78	611.44	615	619.65	618.89	627.05

performance (1/exec time)	chunk size	1	2	4	6
static	default	0.000468	0.000585	0.000885	0.001242
dynamic	default	0.000464	0.000779	0.00128	0.001456
static	10	0.000464	0.000755	0.001229	0.001186
dynamic	10	0.000456	0.000777	0.001254	0.001598

performance (1/exec time)	chunk size	8	10	12	14	16
static	default	0.00133	0.001496	0.001488	0.001513	0.00149
dynamic	default	0.001618	0.001592	0.001622	0.001621	0.001622
static	10	0.001596	0.00151	0.001461	0.001492	0.001632
dynamic	10	0.001635	0.001626	0.001614	0.001616	0.001595





At all cases, as the number of threads increased, the performance gradually increased.the performance is increased gradually. However, the performance is relatively low when it is clearly **static and the chunk size is 1.** The reason is that the load balance is also not good and there is a difference in the amount to be calculated for each thread.

For **Dynamic scheduling,** chunk size does not seem important. Because in the end, threads with a small amount of computation are calculated and calculated from another chunk, so there is no difference in performance.

When **the chunk size of Static is set to 10,** it can be seen that the performance is better than when it is set to 1. This is because by setting the chunk size to 10, the amount calculated for each thread is similar and the load balance is improved.

However, it can be seen that as the number of threads increases (at the point of the number of thread is 16), the performance becomes similar to each other. Due to communication overhead, performance did not continue to increase even if there were more threads.