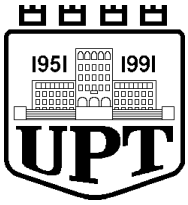

UNIVERSITETI POLITEKNIK I TIRANËS
FAKULTETI I TEKNOLOGJISË DHE INFORMACIONIT
DEPARTAMENTI I INXHINIERISË INFORMATIKE

Punë Laboratori nr. 1

Lënda: Sisteme Te Shperndara

Grupi: III-B

Punoi:
Piro Gjikdhima

Pranoi:
MSc.Megi Tartari

Ushtrimi 1

Llogaritja e shumes se elementeve te nje vektori ne menyre seriale dhe paralele.

Kodi

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <omp.h>

#define MAX_THREADS 6  // Numri maksimal i fijeve që do të testojmë
#define START_SIZE 100  // Madhësia fillestare e vektorit
#define MAX_EXPONENT 5 // Numri i rritjeve (p.sh., 100, 1000, 10000, ...)

double array_sum_parallel(const int* array, size_t size, int num_threads) {
    double sum = 0.0;

    omp_set_num_threads(num_threads);

    #pragma omp parallel for reduction(+:sum)
    for (size_t i = 0; i < size; i++) {
        sum += array[i];
    }
    return sum;
}

int main() {
    printf("Testimi i performancës me ndryshim të madhësisë së vektorit dhe numrit të
threadeve\n");

    // Kalojmë madhësinë e vargut nga 100 në 100*10^MAX_EXPONENT
    for (int exp = 0; exp <= MAX_EXPONENT; exp++) {
        size_t array_size = START_SIZE;
        for (int i = 0; i < exp; i++) {
            array_size *= 10;
        }

        printf("\nMadhësia e vargut: %zu elemente\n", array_size);

        // Alokojmë vargun dhe e inicializojmë me vlera të rastësishme
        int* array = (int*)malloc(array_size * sizeof(int));
        if (!array) {
            fprintf(stderr, "Dështoi alokimi i kujtesës\n");
            return 1;
        }

        srand(time(NULL));
```

```
        for (size_t i = 0; i < array_size; i++) {
            array[i] = rand() % 100 + 1;
        }

        // Ekzekutojmë mbledhjen seriale
        double start_time = omp_get_wtime();
        double serial_sum = array_sum_parallel(array, array_size, 1);
        double serial_time = omp_get_wtime() - start_time;

        printf("Shuma seriale: %.0f (u desh %.6f sekonda)\n", serial_sum,
serial_time);

        // Testojmë për fije nga 2 deri në 6
        for (int threads = 2; threads <= MAX_THREADS; threads++) {
            start_time = omp_get_wtime();
            double parallel_sum = array_sum_parallel(array, array_size, threads);
            double parallel_time = omp_get_wtime() - start_time;

            printf("Shuma paralele (%d threde): %.0f (u desh %.6f sekonda,
përshpejtimi %.2fx)\n",
                   threads, parallel_sum, parallel_time, serial_time /
parallel_time);
        }
        free(array);
    }
    return 0;
}
```

Rezultate

Kur punojmë me vektorë të vegjël (100 - 10,000 elemente), nuk vërejmë ndonjë përmirësim të ndjeshëm në ekzekutimin paralel. Kjo ndodh sepse overhead-i i krijimit dhe menaxhimit të thredeve është më i madh sesa përfitimi i ndarjes së punës. Në këtë rast, koha e shtuar për sinkronizimin dhe koordinimin e thredeve e bën ekzekutimin paralel joefektiv.

Në rastin e vektorëve mesatarë (100,000 elemente), paralelizimi fillon të tregojë përmirësime të dukshme. Procesori shfrytëzon më mirë “multicore” duke ndarë punën në seksione më të mëdha për çdo thred. Kjo redukton kohën totale të ekzekutimit, pasi çdo thred përpunon një pjesë më të madhe të të dhënave në mënyrë të pavarur.

Për vektorë të mëdhenj (1,000,000 elemente), përfitimet e paralelizimit bëhen më të theksuara. Overhead-i i krijimit të thredeve bëhet i papërfillshëm krahasuar me kohën e përgjithshme të përpunimit. Kjo çon në një përshpejtim të ndjeshëm, duke arritur deri në ~4.8x më shpejt me 6 threde në krahasim me ekzekutimin serial.

Në përfundim, paralelizimi është i dobishëm për madhësi të mëdha të të dhënave, ku ndarja e punës sjell një reduktim të ndjeshëm të kohës së ekzekutimit. Megjithatë, për madhësi të vogla, overhead-i i menaxhimit të thredeve e bën ekzekutimin paralel më pak efikas sesa ai serial.

```
Testimi i performancës me ndryshim të madhësisë së vektorit dhe numrit të thredeve

Madhësia e vargut: 100 elemente
Shuma seriale: 5327 (u desh 0.000014 sekonda)
Shuma paralele (2 threde): 5327 (u desh 0.000367 sekonda, përshpejtimi 0.04x)
Shuma paralele (3 threde): 5327 (u desh 0.000628 sekonda, përshpejtimi 0.02x)
Shuma paralele (4 threde): 5327 (u desh 0.000274 sekonda, përshpejtimi 0.05x)
Shuma paralele (5 threde): 5327 (u desh 0.000226 sekonda, përshpejtimi 0.06x)
Shuma paralele (6 threde): 5327 (u desh 0.030641 sekonda, përshpejtimi 0.00x)

Madhësia e vargut: 1000 elemente
Shuma seriale: 51224 (u desh 0.000008 sekonda)
Shuma paralele (2 threde): 51224 (u desh 0.000064 sekonda, përshpejtimi 0.13x)
Shuma paralele (3 threde): 51224 (u desh 0.003398 sekonda, përshpejtimi 0.00x)
Shuma paralele (4 threde): 51224 (u desh 0.000270 sekonda, përshpejtimi 0.03x)
Shuma paralele (5 threde): 51224 (u desh 0.000157 sekonda, përshpejtimi 0.05x)
Shuma paralele (6 threde): 51224 (u desh 0.047844 sekonda, përshpejtimi 0.00x)

Madhësia e vargut: 10000 elemente
Shuma seriale: 508043 (u desh 0.000035 sekonda)
Shuma paralele (2 threde): 508043 (u desh 0.002291 sekonda, përshpejtimi 0.02x)
Shuma paralele (3 threde): 508043 (u desh 0.001035 sekonda, përshpejtimi 0.03x)
Shuma paralele (4 threde): 508043 (u desh 0.000232 sekonda, përshpejtimi 0.15x)
Shuma paralele (5 threde): 508043 (u desh 0.000195 sekonda, përshpejtimi 0.18x)
Shuma paralele (6 threde): 508043 (u desh 0.025423 sekonda, përshpejtimi 0.00x)

Madhësia e vargut: 100000 elemente
Shuma seriale: 5052324 (u desh 0.000387 sekonda)
Shuma paralele (2 threde): 5052324 (u desh 0.009746 sekonda, përshpejtimi 0.04x)
Shuma paralele (3 threde): 5052324 (u desh 0.001441 sekonda, përshpejtimi 0.27x)
Shuma paralele (4 threde): 5052324 (u desh 0.000914 sekonda, përshpejtimi 0.42x)
Shuma paralele (5 threde): 5052324 (u desh 0.000308 sekonda, përshpejtimi 1.26x)
Shuma paralele (6 threde): 5052324 (u desh 0.027271 sekonda, përshpejtimi 0.01x)

Madhësia e vargut: 1000000 elemente
Shuma seriale: 50469268 (u desh 0.003860 sekonda)
Shuma paralele (2 threde): 50469268 (u desh 0.002913 sekonda, përshpejtimi 1.32x)
Shuma paralele (3 threde): 50469268 (u desh 0.001846 sekonda, përshpejtimi 2.09x)
Shuma paralele (4 threde): 50469268 (u desh 0.001209 sekonda, përshpejtimi 3.19x)
Shuma paralele (5 threde): 50469268 (u desh 0.001146 sekonda, përshpejtimi 3.37x)
Shuma paralele (6 threde): 50469268 (u desh 0.026632 sekonda, përshpejtimi 0.14x)

Madhësia e vargut: 10000000 elemente
Shuma seriale: 504868044 (u desh 0.034831 sekonda)
Shuma paralele (2 threde): 504868044 (u desh 0.021178 sekonda, përshpejtimi 1.64x)
Shuma paralele (3 threde): 504868044 (u desh 0.012288 sekonda, përshpejtimi 2.83x)
Shuma paralele (4 threde): 504868044 (u desh 0.010174 sekonda, përshpejtimi 3.42x)
Shuma paralele (5 threde): 504868044 (u desh 0.009212 sekonda, përshpejtimi 3.78x)
Shuma paralele (6 threde): 504868044 (u desh 0.007255 sekonda, përshpejtimi 4.80x)
```

Ushtrimi 2

Llogaritja e faktorialit te elementeve te nje vektori ne menyre seriale dhe paralele.

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <omp.h>

#define MAX_THREADS 6  // Numri maksimal i thredeve që do të testojmë
#define START_SIZE 100 // Madhësia fillestare e vektorit
#define MAX_EXPONENT 5 // Numri i rritjeve (p.sh., 100, 1000, 10000, ...)

// Llogarit faktorialin e një numri
unsigned long long factorial(int n) {
    if (n <= 1) return 1;

    unsigned long long result = 1;
    for (int i = 2; i <= n; i++) {
        result *= i;
    }
    return result;
}

// Implementimi serial
void compute_factorials_serial(const int* input, unsigned long long* output, size_t
size) {
    for (size_t i = 0; i < size; i++) {
        output[i] = factorial(input[i]);
    }
}

// Implementimi paralel me OpenMP
void compute_factorials_parallel(const int* input, unsigned long long* output, size_t
size, int num_threads) {
    omp_set_num_threads(num_threads);
    #pragma omp parallel for
    for (size_t i = 0; i < size; i++) {
        output[i] = factorial(input[i]);
    }
}

int main() {
    printf("Testimi i performancës me ndryshim të madhësisë së vektorit dhe numrit të
thredeve\n");

    for (int exp = 0; exp <= MAX_EXPONENT; exp++) {
        size_t array_size = START_SIZE;
```

```
        for (int i = 0; i < exp; i++) {
            array_size *= 10;
        }

        printf("\nMadhësia e vargut: %zu elemente\n", array_size);

        // Alokojmë vektoret
        int* input_array = (int*)malloc(array_size * sizeof(int));
        unsigned long long* output_serial = (unsigned long long*)malloc(array_size *
sizeof(unsigned long long));
        unsigned long long* output_parallel = (unsigned long long*)malloc(array_size
* sizeof(unsigned long long));

        if (!input_array || !output_serial || !output_parallel) {
            fprintf(stderr, "Dështoi alokimi i kujtesës\n");
            return 1;
        }

        srand(time(NULL));
        for (size_t i = 0; i < array_size; i++) {
            input_array[i] = rand() % 20 + 1; // Vlera midis 1 dhe 20 për të shmangur
overflow
        }

        // Ekzekutojmë versionin serial
        double start_time = omp_get_wtime();
        compute_factorials_serial(input_array, output_serial, array_size);
        double serial_time = omp_get_wtime() - start_time;
        printf("Koha seriale: %.6f sekonda\n", serial_time);

        // Testojmë versionin paralel me 2 deri në 6 threde
        for (int threads = 2; threads <= MAX_THREADS; threads++) {
            start_time = omp_get_wtime();
            compute_factorials_parallel(input_array, output_parallel, array_size,
threads);
            double parallel_time = omp_get_wtime() - start_time;

            printf("Koha paralele (%d threde): %.6f sekonda, përshpejtimi %.2fx\n",
                   threads, parallel_time, serial_time / parallel_time);
        }

        free(input_array);
        free(output_serial);
        free(output_parallel);
    }

    return 0;
}
```

Rezultate

```
Testimi i performancës me ndryshim të madhësisë së vektorit dhe numrit të thredeve

Madhësia e vargut: 100 elemente
Koha seriale: 0.000005 sekonda
Koha paralele (2 threde): 0.000477 sekonda, përshpejtimi 0.01x
Koha paralele (3 threde): 0.000369 sekonda, përshpejtimi 0.01x
Koha paralele (4 threde): 0.000272 sekonda, përshpejtimi 0.02x
Koha paralele (5 threde): 0.000198 sekonda, përshpejtimi 0.03x
Koha paralele (6 threde): 0.000201 sekonda, përshpejtimi 0.03x

Madhësia e vargut: 1000 elemente
Koha seriale: 0.000044 sekonda
Koha paralele (2 threde): 0.000027 sekonda, përshpejtimi 1.62x
Koha paralele (3 threde): 0.000053 sekonda, përshpejtimi 0.84x
Koha paralele (4 threde): 0.001457 sekonda, përshpejtimi 0.03x
Koha paralele (5 threde): 0.000191 sekonda, përshpejtimi 0.23x
Koha paralele (6 threde): 0.049177 sekonda, përshpejtimi 0.00x

Madhësia e vargut: 10000 elemente
Koha seriale: 0.000632 sekonda
Koha paralele (2 threde): 0.000674 sekonda, përshpejtimi 0.94x
Koha paralele (3 threde): 0.001136 sekonda, përshpejtimi 0.56x
Koha paralele (4 threde): 0.000359 sekonda, përshpejtimi 1.76x
Koha paralele (5 threde): 0.000321 sekonda, përshpejtimi 1.97x
Koha paralele (6 threde): 0.001198 sekonda, përshpejtimi 0.53x

Madhësia e vargut: 100000 elemente
Koha seriale: 0.004706 sekonda
Koha paralele (2 threde): 0.004312 sekonda, përshpejtimi 1.09x
Koha paralele (3 threde): 0.001864 sekonda, përshpejtimi 2.53x
Koha paralele (4 threde): 0.001361 sekonda, përshpejtimi 3.46x
Koha paralele (5 threde): 0.001121 sekonda, përshpejtimi 4.20x
Koha paralele (6 threde): 0.001127 sekonda, përshpejtimi 4.17x

Madhësia e vargut: 1000000 elemente
Koha seriale: 0.052551 sekonda
Koha paralele (2 threde): 0.031274 sekonda, përshpejtimi 1.68x
Koha paralele (3 threde): 0.015868 sekonda, përshpejtimi 3.31x
Koha paralele (4 threde): 0.014093 sekonda, përshpejtimi 3.73x
Koha paralele (5 threde): 0.011335 sekonda, përshpejtimi 4.64x
Koha paralele (6 threde): 0.016878 sekonda, përshpejtimi 3.11x

Madhësia e vargut: 10000000 elemente
Koha seriale: 0.455387 sekonda
Koha paralele (2 threde): 0.224162 sekonda, përshpejtimi 2.03x
Koha paralele (3 threde): 0.164328 sekonda, përshpejtimi 2.77x
Koha paralele (4 threde): 0.117795 sekonda, përshpejtimi 3.87x
Koha paralele (5 threde): 0.105858 sekonda, përshpejtimi 4.30x
Koha paralele (6 threde): 0.087169 sekonda, përshpejtimi 5.22x
```

Të njëjtat rezultate si ushtrimi 1 përftojmë dhe në ushtrimin 2.

Kur punojmë me vektorë të vegjël (100 - 10,000 elemente), paralelizimi nuk tregon përmirësime të dukshme. Kjo ndodh sepse overhead-i i krijimit dhe menaxhimit të thredeve është më i madh sesa përfitimi i ndarjes së punës. Në këtë rast, koha e shtuar për sinkronizimin dhe koordinimin e thredeve e bën ekzekutimin paralel joefektiv.

Në rastin e vektorëve mesatarë (100,000 elemente), paralelizimi fillon të tregojë përmirësime të dukshme. Procesori shfrytëzon më mirë shumëbërthamësinë duke ndarë punën në seksione më të mëdha për çdo thred. Kjo redukton kohën totale të ekzekutimit, pasi çdo thred përpunon një pjesë më të madhe të të dhënave në mënyrë të pavarur.

Për vektorë të mëdhenj (1,000,000 elemente), përfitimet e paralelizimit bëhen më të theksuara. Overhead-i i krijimit të thredeve bëhet i papërfillshëm krahasuar me kohën e përgjithshme të përpunimit. Kjo çon në një përshpejtim të ndjeshëm, duke arritur deri në ~3.11x më shpejt me 6 threde në krahasim me ekzekutimin serial.

Për vektorë shumë të mëdhenj (10,000,000 elemente), përfitimet e paralelizimit bëhen jashtëzakonisht të dukshme. Overhead-i i krijimit të thredeve bëhet i papërfillshëm dhe koha e përpunimit zvogëlohet në mënyrë të konsiderueshme. Kjo çon në një përshpejtim të jashtëzakonshëm, duke arritur deri në ~5.22x më shpejt me 6 threde në krahasim me ekzekutimin serial.

Ky është një konfirmim i parimit të përgjithshëm që paralelizimi është më efektiv kur madhësia e problemit është mjaft e madhe për të justifikuar koston e krijimit dhe menaxhimit të thredeve.