Laborator 4
Execuția suprapusă a buclelor de tip parallel for

Problemă de rezolvat

Să se implementeze înmulțirea a două matrice NxN.

Să se paralelizeze algoritmul secvențial implementat. Să se compare timpii de execuție între algoritmul secvențial și cel paralel și să se calculeze eficiența paralelă.

Algoritm secvențial

```
#include "stdafx.h"
#include <stdlib.h>
#include <time.h>
#include <math.h>
#include <omp.h>

#define N 1000
double A[N][N], B[N][N], C[N][N];

int main()
{
	clock_t start, stop;
	double durata_secv, durata_par;

	srand((unsigned)time(NULL));
	for (int i = 0; i < N; i++) {
		for(int j=0; j<N; j++) {
		A[i][j] = (double)(rand() % 1000);
		B[i][j] = (double)(rand() % 1000);
		C[i][j] = (double)(rand() % 1000);
	}
	}


	start = clock();
	for (int i = 0; i < N; i++) {
		for (int j = 0; j < N; j++) {
			C[i][j] = 0;
```

```
                for (int k = 0; k < N; k ++)
                    C[i][j] += A[i][k] * B[k][i];
            }
        }
        stop = clock();
        durata_secv = (double)(stop - start) / CLOCKS_PER_SEC;
        printf("SECV %2.10f seconds\n", durata_secv);

        getchar();
        return 0;
}
```

Algoritm paralel (varianta 1)

Cea mai simplă soluție de paralelizare a algoritmului de înmulțire este paralelizarea primei bucle for – paralelizarea la nivel de calcul linie matrice rezultat. Se va duplica codul de calcul și se va insera înainte de prima instrucțiune for:

```
#define P 4
…
#pragma omp parallel for num_threads(P)
```

și la final:

```
durata_par = (double)(stop - start) / CLOCKS_PER_SEC;
printf("P_V1 %2.10f seconds\n", durata_par);
printf("P_V1 FA=%2.2f\n", durata_secv / durata_par);
printf("P_V1 E(%d)=%2.2f\n", P, (durata_secv / durata_par) / P);
```

În urma execuției se va observa o scădere semnificativă a timpului de execuție și un factor de accelerare supraunitar apropiat de P (eficiența paralelă apropiată de 1).

```
SECV 7.9510000000 seconds
P_V1 2.0920000000 seconds
P_V1 FA=3.80
P_V1 E(4)=0.95
```

Algoritm paralel (varianta 2)

Dacă se dorește creșterea gradului de paralelism (scăderea granularității) se poate încerca paralelizarea algoritmului la nivel de element al matricei rezultat prin paralelizarea suprapusă a

celor două bucle for. Se va adăuga înainte de al doilea for o clauză de paralelizare similară cu cea anterioară.

Explicați de ce rezultatele sunt mai proaste. Dovediți explicația.

Algoritm paralel (varianta 2+)

Ca alternativă la varianta 2 paralelizați cele două bucle for cu o singură directivă de paralelizare conform algoritmului:

```
#pragma omp parallel for num_threads(P)
      for (int ij = 0; ij < N*N; ij++) {
            int i = ij % N;
            int j = ij / N;
            C[i][j] = 0;
            for (int k = 0; k < N; k++)
                  C[i][j] += A[i][k] * B[k][i];
            }
```

Comparați performanțele de execuție ale algoritmului de mai sus cu cele ale variantei 1. Cum explicați diferența de performanță? Încercați să reduceți diferența utilizând o clauză de tip *schedule*.