



МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

ФАКУЛЬТЕТ ІНФОРМАТИКИ ТА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ
КАФЕДРА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ

ЛАБОРАТОРНА РОБОТА №4
з дисципліни «Паралельні та розподілені обчислення
Паралельне програмування-2»
Тема: «OpenMP. Бар'єри, критичні секції»

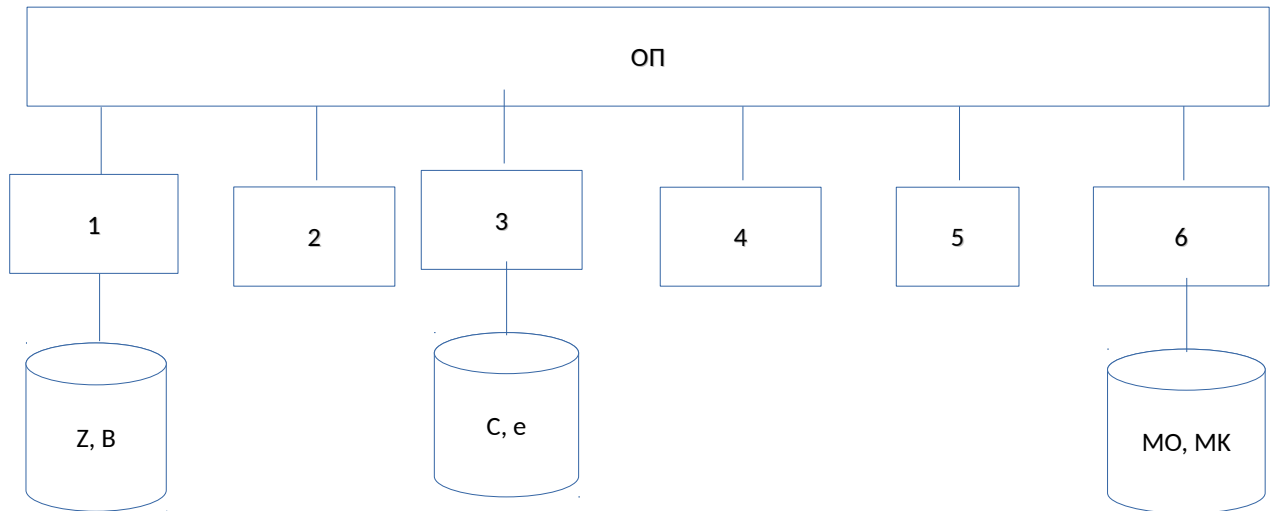
Виконав:
студент 3-го курсу
групи ІП-42
з номер заліковки 4206
Дзюба Влад

Завдання

Мета роботи: розробка програми для ПКС зі СП

Мова програмування: C++

Засоби організації взаємодії процесів: бар'єри, критичні секції OpenMP



Варіант

$$A = \max(Z) \cdot B + e \cdot C \cdot (MO \cdot MK)$$

Математичний паралельний алгоритм:

1. $z_h = \max(Z_h)$
2. $z = \max(z, Z_h)$
3. $A_h = z \cdot B_h + e \cdot C \cdot (MO \cdot MK_h)$

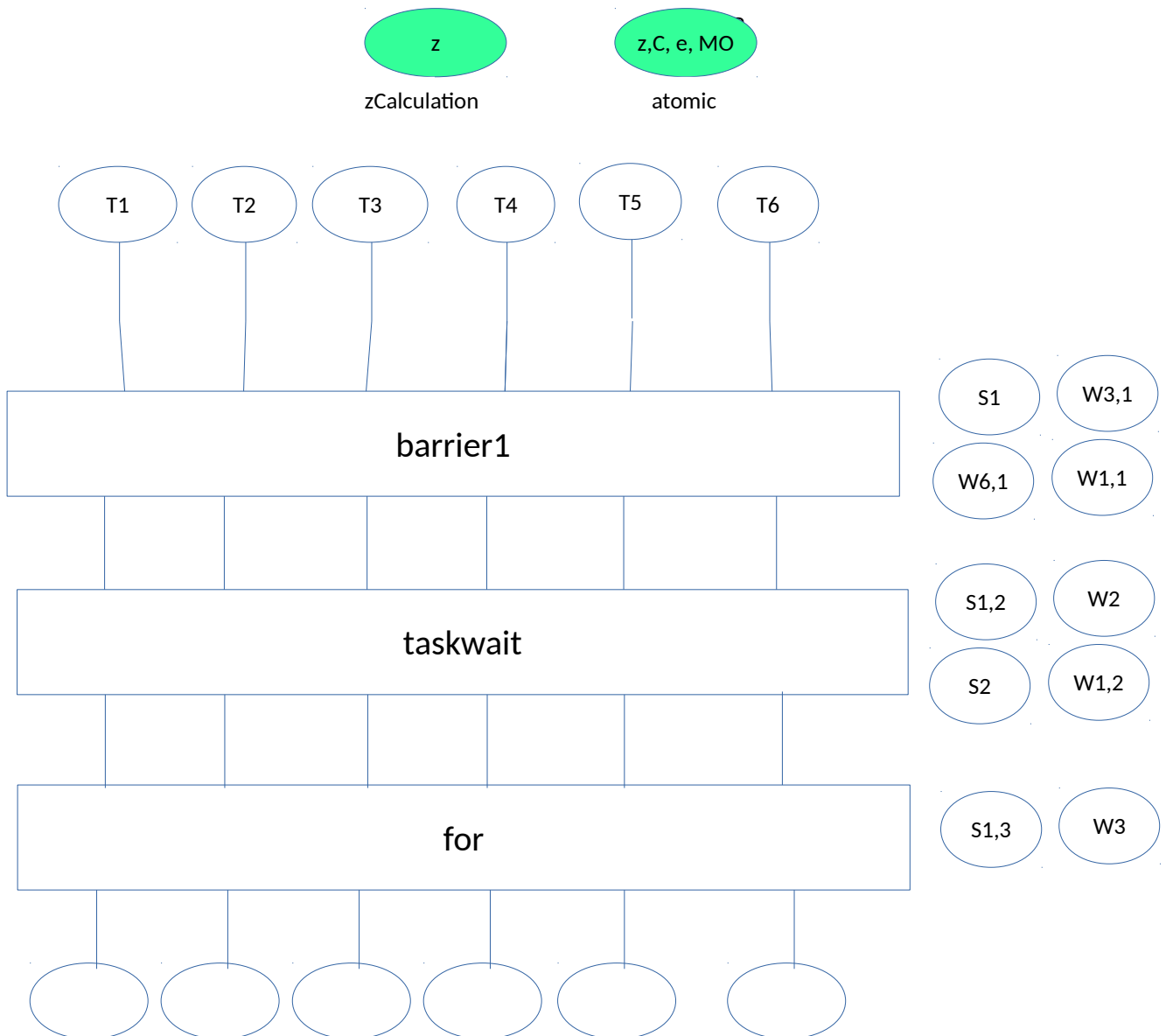
CP: z, e, C, MO.

Алгоритм для кожного процесу:

| | | | |
|--|--|--|---|
| T_1 : 1. Ввести Z, B. 2. Чекати сигнал про завершення вводу з T_3 . 3. Чекати сигнал про завершення вводу з T_6 . 4. Сигнал про початок обрахунку z до всіх $T_i, i \in [2..6]$. 5. Обрахувати $z_h = \max(Z_h)$. 6. Обрахувати $z = \max(z, z_h)$. 7. Чекати сигнал про кінець обрахунку z зі всіх $T_i, i \in [2..6]$. 8. Сигнал про початок обрахунку A до всіх $T_i, i \in [2..6]$. 9. Копіювати $z_1 = z, e_1 = e, C_1 = C, MO_1 = MO$. 10. Обрахувати $A_h = z_1 \cdot B_h + e_1 \cdot C_1 \cdot (MO_1 \cdot MK_h)$. 11. Чекати сигнал про завершення обрахунку A_h з $T_i, i \in [2..6]$. 12. Вивести A . | $W_{3,1}$ $W_{6,1}$ S_1 КУ1 W_2 S_2 КУ2 W_3 | T_3 : 1. Ввести C, e. 2. Сигнал про завершення вводу до T_1 . 3. Чекати сигнал про початок обрахунку z з T_1 . 4. Обрахувати $z_h = \max(Z_h)$. 5. Обрахувати $z = \max(z, z_h)$. 6. Сигнал про кінець обрахунку z до T_1 . 7. Чекати сигнал про початок обрахунку A з T_1 . 8. Копіювати $z_3 = z, e_3 = e, C_3 = C, MO_3 = MO$. 9. Обрахувати $A_h = z_3 \cdot B_h + e_3 \cdot C_3 \cdot (MO_3 \cdot MK_h)$. 10. Сигнал про завершення обрахунку A_h до T_1 . | $S_{1,1}$ $W_{1,1}$ КУ1 $S_{1,2}$ $W_{1,2}$ КУ2 $S_{1,3}$ |
| T_6 : 1. Ввести C, e. 2. Сигнал про завершення вводу до T_1 . 3. Чекати сигнал про початок обрахунку z з T_1 . 4. Обрахувати $z_h = \max(Z_h)$. 5. Обрахувати $z = \max(z, z_h)$. 6. Сигнал про кінець обрахунку z до T_1 . 7. Чекати сигнал про початок обрахунку A з T_1 . 8. Копіювати $z_6 = z, e_6 = e, C_6 = C, MO_6 = MO$. 9. Обрахувати $A_h = z_6 \cdot B_h + e_6 \cdot C_6 \cdot (MO_6 \cdot MK_h)$. 10. Сигнал про завершення обрахунку A_h до T_1 . | $S_{1,1}$ $W_{1,1}$ КУ1 $S_{1,2}$ $W_{1,2}$ КУ2 $S_{1,3}$ | $T_i, i \in \{2, 4, 5\}$: 1. Ввести C, e. 2. Сигнал про завершення вводу до T_1 . 3. Чекати сигнал про початок обрахунку z з T_1 . 4. Обрахувати $z_h = \max(Z_h)$. 5. Обрахувати $z = \max(z, z_h)$. 6. Сигнал про кінець обрахунку z до T_1 . 7. Чекати сигнал про початок обрахунку A з T_1 . 8. Копіювати $z_6 = z, e_6 = e, C_6 = C, MO_6 = MO$. 9. Обрахувати $A_h = z_i \cdot B_h + e_i \cdot C_i \cdot (MO_i \cdot MK_h)$. 10. Сигнал про завершення обрахунку A_h до T_1 . | $S_{1,1}$ $W_{1,1}$ КУ1 $S_{1,2}$ $W_{1,2}$ КУ2 $S_{1,3}$ |

Структурна схема взаємодії процесів

Запис « $T_i, i \leftarrow \{ 2, 4, 5 \}$ » для позначення того, що даний блок та його зв'язки є ідентичними для потоків T2, T4 та T5.



Лістинг програми:

main.cpp

```
/*
 * ИП-42, Дзюба Влад
 */
#include <omp.h>
#include <iostream>

using namespace std;

#define N 12
#define P 6

int* initVector(int n);
int* initMatrix(int n);
int max(int *arr, int b, int e);

int main() {
    // Initialize data
    int *Z, *B, e, *C, *M0, *MK, *A = new int[N];
    int z = 0;

#pragma omp parallel num_threads(P)
    {
        int thread_num = omp_get_thread_num();
        cout << "Thread " << thread_num + 1 << " started" << endl;
        // Initializes first thread input
        switch (thread_num) {
            case 0: {
                Z = initVector(N);
                B = initVector(N);
            } break;
            // Initializes third thread input
            case 2: {
                C = initVector(N);
                e = 1;
            } break;
            // Initializes sixth thread input
            case 5: {
                M0 = initMatrix(N);
                MK = initMatrix(N);
            } break;
        }
        // Finds maximum of Z
#pragma omp barrier
        for (int i = 0; i < N; i++) {
#pragma omp task shared(z)
            {
#pragma omp critical(zCalculation)
                z = Z[i] > z ? Z[i] : z;
            }
        }
    }
#pragma omp taskwait
```

```

        // Calculates A
#pragma omp for
        for (int i = 0; i < N; i++) {
            int zi, ei, *Ci, *MOi;
            // Copies shared data
#pragma omp read atomic
            zi = z;
#pragma omp read atomic
            ei = e;
#pragma omp read atomic
            Ci = C;
#pragma omp read atomic
            MOi = MO;
            // Calculates A
            A[i] = zi * B[i];
            for (int j = 0; j < N; j++) {
                int ok = 0;
                for (int k = 0; k < N; k++) {
                    ok += MOi[j * N + k] * MK[k * N + i];
                }
                A[i] += ei * ok * Ci[j];
            }
        }
#pragma omp master
        if (N < 20) {
            for (int i = 0; i < N; i++) {
                cout << A[i] << " ";
            }
            cout << endl;
        }
        cout << "Thread " << thread_num + 1 << " ended" << endl;
    }
}

// Initializes vectors
int* initVector(int n) {
    int* res = new int[n];
    for (int i = 0; i < n; i++) {
        res[i] = 1;
    }
    return res;
}

// Initializes matrices
int* initMatrix(int n) {
    int* res = new int[n * n];
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            res[i * n + j] = 1;
        }
    }
    return res;
}

```