Задача 5. Распределенное умножение матриц. Более эффективная сеть передачи данных используется эффективнее. Уменьшение времени ожидания поступления данных

Исследование провёл студент группы 22207 Гордеев Никита

Дата выполнения работы 18.12.2022 (Вариант 2)

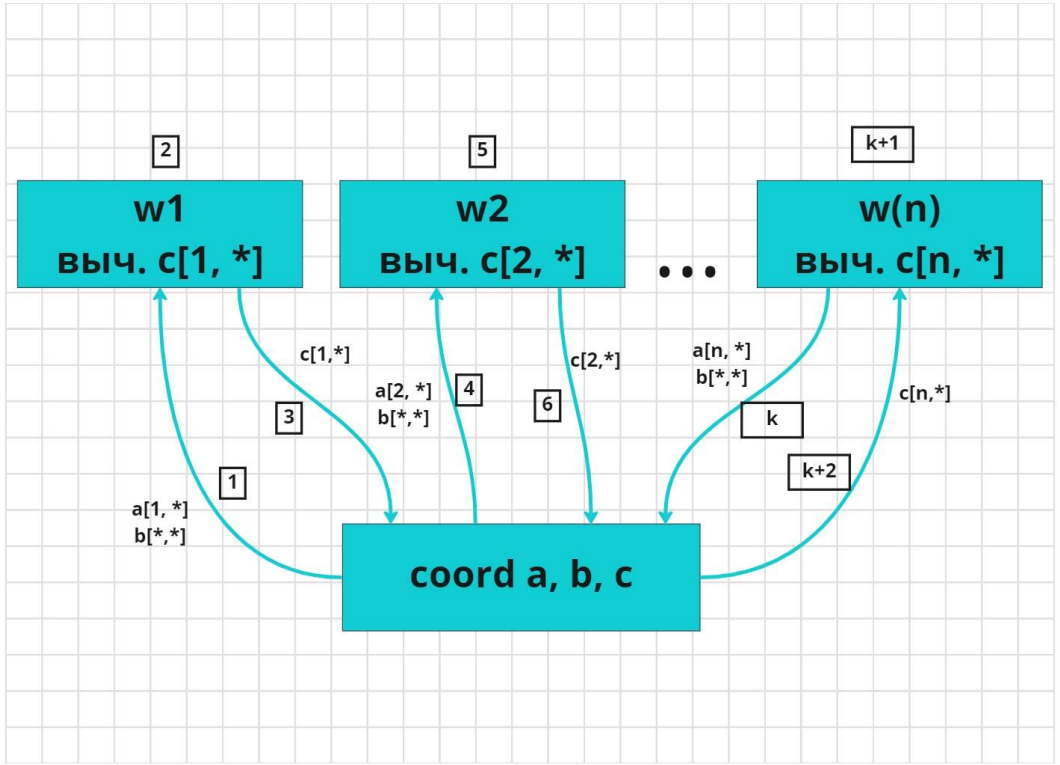
Рассмотренная программа распределенного умножения матриц

```
process w [i = 1 to n] {
    double A[n], B[n, n], C[n];
    receive (coord, A[*]);
    receive (coord, B[*]);
    for (j = 1 to n) {
        C[j] = 0.0;
        for [k = 1 to n]
            C[j] += A[k] * B[k, j]'
     }
    send(Coord, C[*]);
}
```

```
process coord {
    double A[n, n], B[n, n], C[n, n];
    init(A, B);
    for [i = 1 to n - 1] {
        send (w[i], A[i, *];
        send (w[i], B[*, *];
    }
    for [i = 1 to n]
        receive[w[i], C[i,*];
}
```

Анализ рассмотренной программы распределенного умножения матриц

1) **Взаимодействие управляющий-рабочие**
2) **Управляющий процесс отправляет рабочим необходимые им данные для вычисления строки i конечной матрицы**
3) **Рабочие процессы отправляют управляющему результат**



Проблема

1) **Рабочие процессы не начинают вычисления одновременно, поскольку передача исходных данных выполняется последовательно**
2) **Не эффективно используется сеть, когда пытается передать O(N*N) данных и нельзя начать вычисления пока не получены все эти данные**

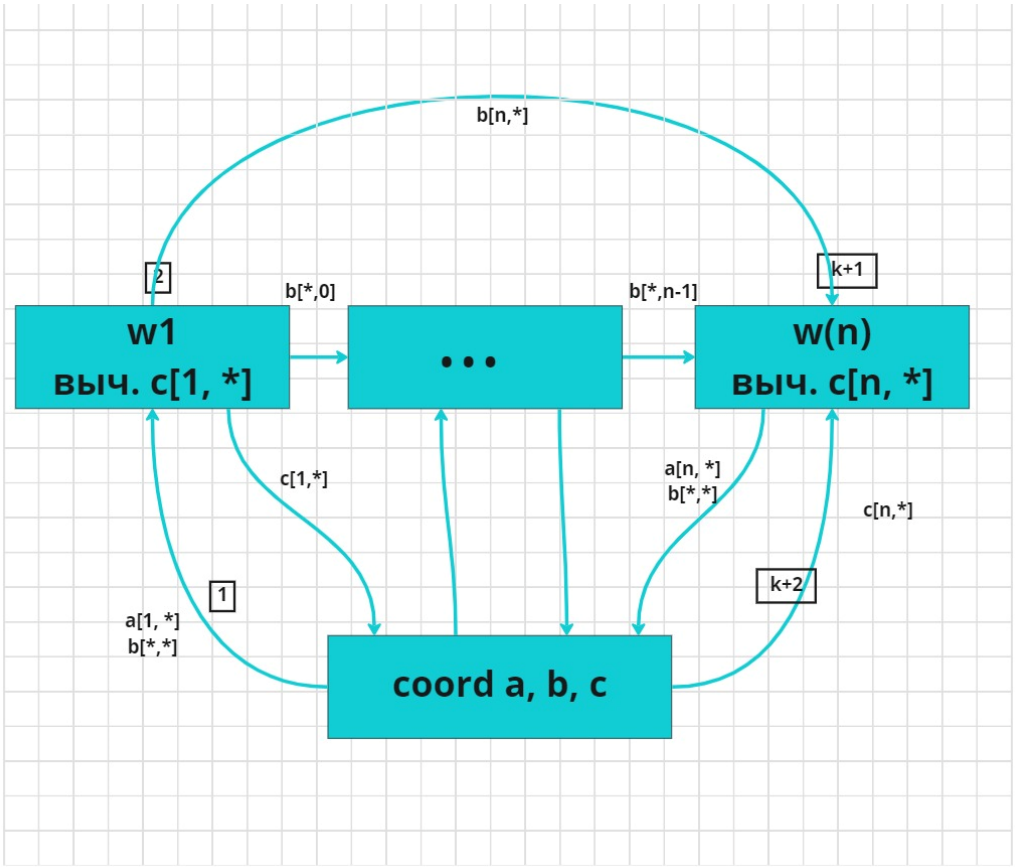
Постановка задач

На основе рассмотренной на лекции программы распределенного умножения матриц необходимо предложить вариант умножения матриц, в котором:

1) **Сеть передачи данных используется эффективнее;**
2) **Участвующие параллельные процессы не тратят много времени на ожидание поступления данных для вычислений.**

Идея изменённой программы:

1) **Круговой конвейер**
2) **Все процессы – взаимодействующие равные**
3) **Каждый процесс получает i-ю строку матрицы A и i-й столбец матрицы B. Теперь он может вычислить значение C[i,i]. Каждый процесс будет отсылать текущий столбец следующему и получать новый от предыдущего.**

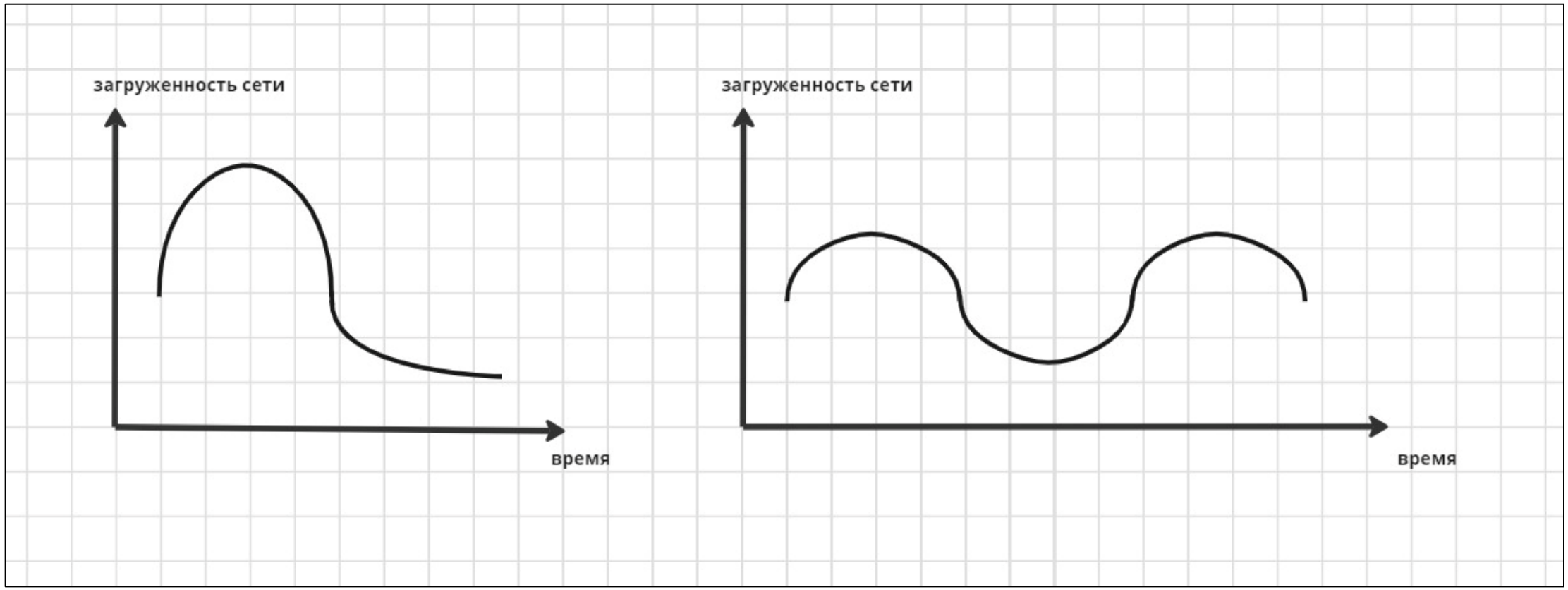


Изменённая программа умножения матриц

```
process worker [i = 0 to n - 1] {
    double a[n]; // строка i матрицы a
    double b[n]; // столбец матрица b
    double c[n]; // строка i матрицы c
    double sum = 0.0; // для промежуточных произведений
    int nextCol = i; // следующий столбец результатов
    recive строку i матрицы a и столбец i матрицы b;
    //вычислить c[i, i] = a[i, *] * b[*, i]
    for [k = 0 to n - 1]
        sum = sum + a[k] * b[k];
    c[nextCol] = sum;
    // круговой конвейер
    for [j = 1 to n - 1] {
        send текущий столбец b следующему процессу;
        recieve новый столбец матрицы b от предыдущего;
        sum = 0.0;
        for [k = 0 to n - 1]
            sum = sum + a[k] * b[k];
        if (nextCol == 0)
            nextCol = n - 1;
        else
            nextCol--;
        c[nextCol] = sum;

    send вектор-результат c управляющему процессу;
}
```

Сравнение программ — визуализация использования сети



Сравнение программ

Исходная программа

1) **Дублирование матрицы B данные принимаются один раз**

Изменённая программа

1) **Сеть передачи используется гораздо эффективней т.к. матрицы A и B распределены по всем процессам, каждый процесс в один момент времени хранит одну строку и один столбец**
2) **N процессов N раз отправляют и принимают данные, что плохо сказывается на времени работы, но процессы не ожидают долго поступления данных, т.к. все равнозначны и работают приблизительно одинаковое время**

Вывод:

- **Второй вариант гораздо более эффективен**

Материалы:

- **Эндрюс Г.Р. Основы многопоточного, параллельного и распределенного программирования / Г.Р.Эндрюс. - Москва : Вильямс, 2003. - 512 с**

Изменения:

- **Версия 2:**
  - **Исправлены открывающие и закрывающие скобки**
  - **Добавлены комментарии в программе**
  - **Проведен более подробный анализ**