Задача 3. Итеративное умножение больших матриц. Линейный размер квадратной матрицы больше числа доступных программе процессоров

Исследование провёл студент группы 22207 Гордеев Никита

Дата выполнения работы 25.12.2022 (Вариант 3)

Проблема

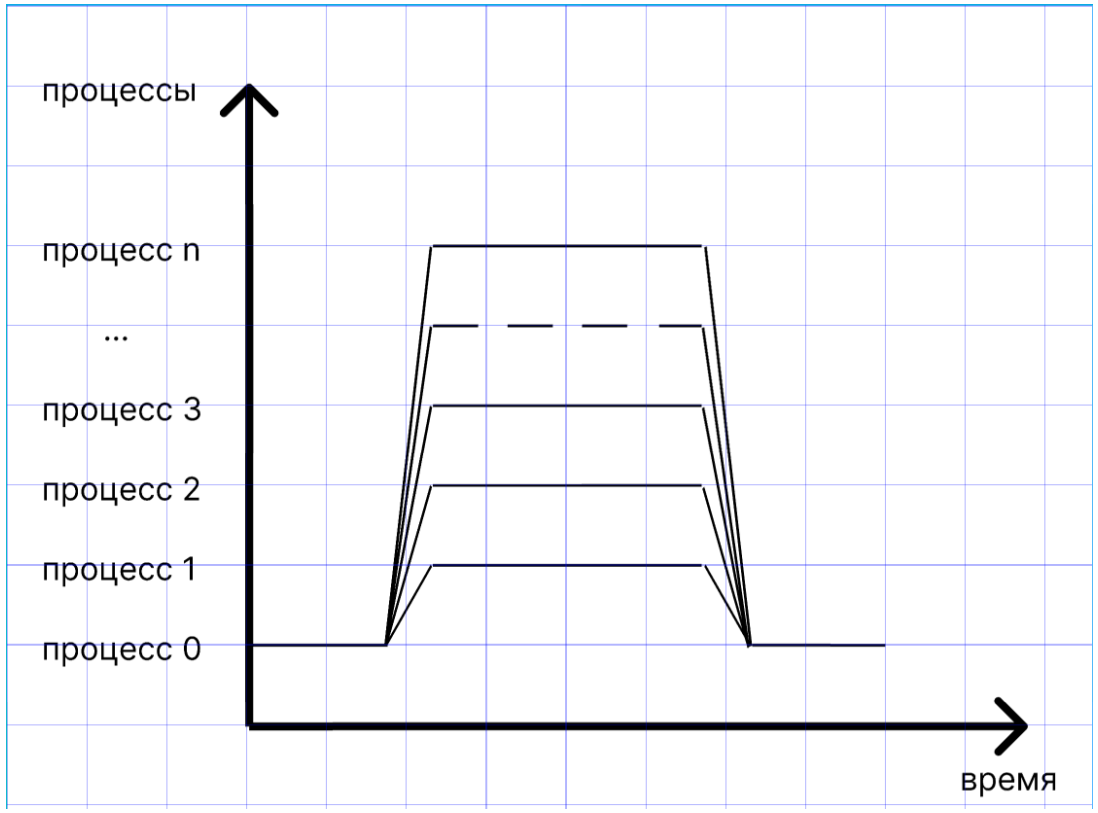
При числе процессоров *p* << размерности матрицы *n* невозможно выделить на каждый процессор только один процесс.

Постановка задач:

1) **Рассмотреть случай, когда *p*<<*n*,**
   1) ***p* – число доступных программе процессоров.**
   2) ***n* – линейный размер квадратной матрицы,**
2) **Предложить параллельную программу итеративного умножения матриц, когда число процессов не превосходит *p*.**

Рассмотрение случая, когда n = p:

```
double a[n,n], b[n,n];
co [i = 0 to (n - 1)] { # по строкам A
    for [j = 0 to (n - 1)] { # по столбцам B
        # выч-ем скалярн. произв. стр. i на ст. j
        c[i, j] = 0, 0; # нач. знач. суммы
        for [k = 0 to (n - 1)] {
            # доб-ем очередн. слаг-е в скаляр. произв.
            c[i, j] += a[i, k] * b[k, j];
        }
    }
}
```



Предложение:

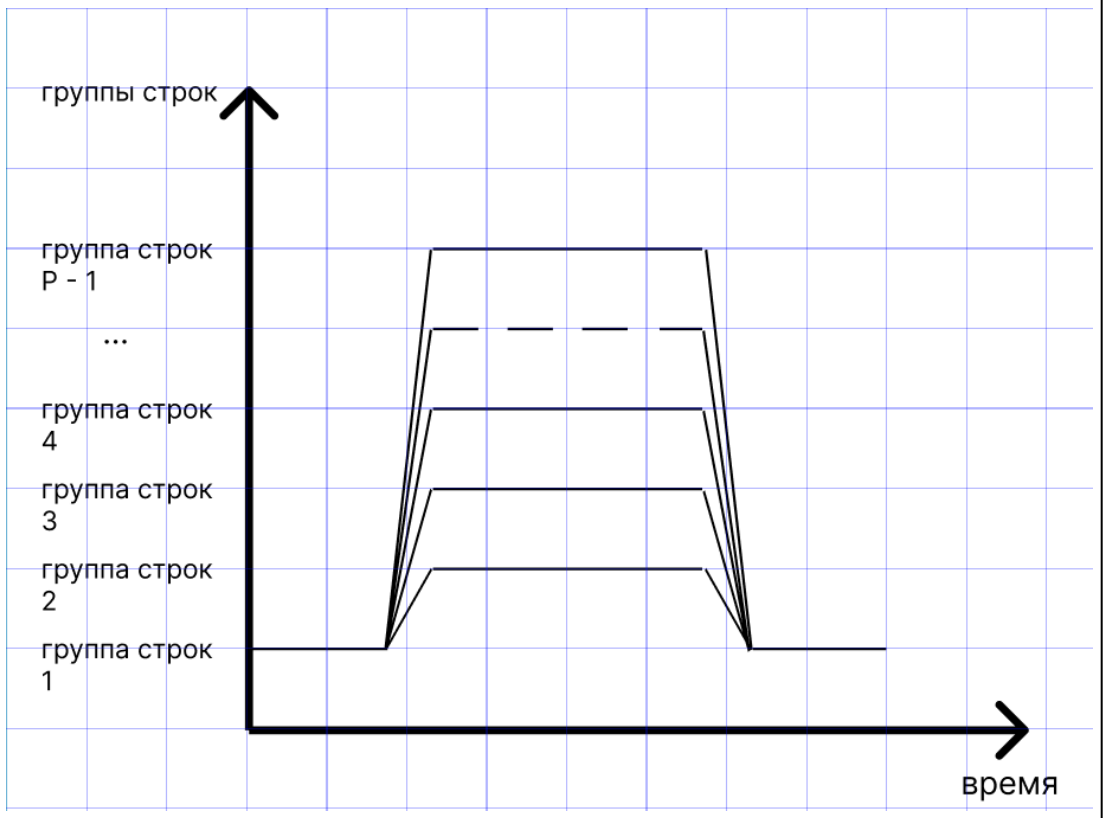
-] ***n*** **– размер матрицы.** ***p*** **– количество процессов.**
-] ***ceil()*** **- округляет аргумент до ближайшего большего целого.**
- **Введём ограничитель** ***cnt = ceil(n/p)*** **— количество строк, работа с которыми приходится на** ***1*** **процесс;**
- **Переберем** ***cnt*** **– количество строк, и запустим вычисление строк этой группы на отдельном процессоре;**
- **Увеличиваем** ***cnt*** **на** ***1*****, если** ***n*** **не делится нацело на** ***p*****;**

Ожидаемые результаты:

1) **При распараллеливании данной программы образуются p процессов, каждый из которых обрабатывает до *cnt* строк.**
2) **За счёт распараллеливания уменьшается время работы программы.**
3) **Так как при каждом процессе меняются только его переменные, не возникнет коллизии, так как никакие два процесса одновременно не будут изменять одну переменную.**

Рассмотрение случая, когда n<<p:

```
// выделение синим – доработки исходной программы
double a[n,n], b[n,n], c[n,n];
cnt = ceil(n/p); // количество строк в группе
if (n % p != 0)
    cnt++;
// по группам параллельно
co [pr = 0 to p-1] {
    first = pr * cnt;
    last = min((pr + 1) * cnt - 1, n - 1);
    for [i = first to last] {
    //выч-ем скалярн. произв. стр. i на ст. j
        c[i, j] = 0, 0;
        for [k = 0 to (n - 1)] {
            // доб-ем очередн. слаг-е в скаляр. произв.
                c[i, j] += a[i, k] * b[k, j];
        }
    }
}
```



Материалы:

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

Изменения

- **Версия 2**
  - **Исправил формулировку задачи на правильную**
  - **Перерисовал графики**
  - **Внёс полное название задачи на титульный слайд**
- **Версия 3**
  - **Комментарии в код**
  - **Выделил переменные в тексте курсивом**