ПРИМЕР ОРГАНИЗАЦИИ ПАРАЛЛЕЛЬНЫХ ЗЕРНИСТЫХ ВЫЧИСЛИТЕЛЬНЫХ ПРОЦЕССОВ И ОБМЕНА ДАННЫМИ

(вариант 21 ЛабМММ)

Дан *ijk*-алгоритм перемножения двух квадратных матриц порядка N:

```
do  i = 1, N
    do  j = 1, N
S1(i,j): c(i,j) =0
        do  k = 1, N
S2(i,j,k):    c(i,j) = c(i,j) + a(i,k) b(k,j)
        enddo
    enddo
enddo
```

Требуется разработать параллельный алгоритм согласно варианту 21.

Тайлинг: r_1 – параметр,

r_2 – параметр,

Q_3 – параметр, $r_3 = \left\lceil \frac{N}{Q_3} \right\rceil$;

s-координата: k;

коммуникации: трансляция части (согласованной с тайлом) C.

Информационная структура алгоритма. Зависимости алгоритма задаются функциями

$$\overline{\Phi}_{1,2}(i,j,1) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} i \\ j \\ 1 \end{pmatrix}, \quad V_{1,2} = \left\{ (i,j,k) \in Z^3 \,\middle|\, 1 \le i \le N,\ 1 \le j \le N,\ k = 1 \right\},$$

$$\overline{\Phi}_{2,2}(i,j,k) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} i \\ j \\ k \end{pmatrix} - \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}, \quad V_{2,2} = \left\{ (i,j,k) \in Z^3 \,\middle|\, 1 \le i \le N,\ 1 \le j \le N,\ 2 \le k \le N \right\}.$$

Тайлинг. Разобьем все три цикла (циклы с параметрами i, j, k); $Q_1 = \left\lceil \frac{N_1}{r_1} \right\rceil$, $Q_2 = \left\lceil \frac{N_2}{r_2} \right\rceil$. Получим

```
do  i^gl = 0, Q1–1
do  i = 1 + i^gl r1, min((i^gl +1)r1, N)
    do  j^gl = 0, Q2–1
    do  j = 1 + j^gl r2, min(( j^gl +1)r2, N)
S1(i,j): c(i,j) =0
        do  k^gl = 0, Q3–1
        do k = 1+ k^gl r3, min((k^gl+1)r3, N)
S2(i,j,k):    c(i,j) = c(i,j) + a(i,k) b(k,j)
```

```
        enddo
        enddo
    enddo
    enddo
enddo
enddo
```

Цикл с параметром k^{gl} не может быть внутренним по отношению к локальным циклам. Кроме того, его не должно быть в окружении оператора $c(i,j)=0$, так как в окружении этого оператора первоначально не было цикла с параметром k. Поэтому распределим циклы с параметрами i и j между выполняемыми операторами и осуществим необходимую перестановку локальных и глобальных циклов:

```
do  i^gl = 0, Q_1–1
    do  j^gl = 0, Q_2 –1
        do  i = 1 + i^gl r_1, min((i^gl +1)r_1, N)
            do  j = 1 + j^gl r_2, min(( j^gl +1)r_2, N)
S_1(i,j):          c(i,j) =0
            enddo
        enddo
        do  k^gl = 0, Q_3–1
            do  i = 1 + i^gl r_1, min((i^gl +1)r_1, N)
                do  j = 1 + j^gl r_2, min(( j^gl +1)r_2, N)
                    do k = 1+ k^gl r_3, min((k^gl+1)r_3, N)
S_2(i,j,k):                c(i,j) = c(i,j) + a(i,k) b(k,j)
                    enddo
                enddo
            enddo
        enddo(k^gl)
    enddo(j^gl)
enddo(i^gl)
```

Операторы S_1 и S_2 окружены разными наборами глобальных циклов, поэтому требуются тайлы двух типов:

```
do  i^gl = 0, Q_1–1
    do  j^gl = 0, Q_2 –1
        Tile1(i^gl,j^gl)
        do  k^gl = 0, Q_3–1
            Tile2(i^gl,j^gl,k^gl)
        enddo(k^gl)
    enddo(j^gl)
enddo(i^gl)
```

Вычисления тайла первого типа Tile1(i^{gl},j^{gl}):

```
        do  i = 1 + i^gl r_1, min((i^gl +1)r_1, N)
            do  j = 1 + j^gl r_2, min(( j^gl +1)r_2, N)
S_1(i,j):           c(i,j)=0
            enddo
        enddo
```

Вычисления тайла второго типа Tile2(i^{gl},j^{gl},k^{gl}):

```
            do  i = 1 + i^gl r_1, min((i^gl +1)r_1, N)
                do  j = 1 + j^gl r_2, min((j^gl+1)r_2, N)
                    do k = 1+ k^gl r_3, min((k^gl+1)r_3, N)
S_2(i,j,k):                 c(i,j) = c(i,j) + a(i,k) b(k,j)
                    enddo
                enddo
            enddo
```

Обоснуем корректность тайлинга (для любого варианта). Имеются зависимости $S_1(i,j)\to S_2(i,j,1)$, $S_2(i,j,k-1)\to S_2(i,j,k)$. Достаточные условия допустимости тайлинга выполняются: для любой зависимости $S_\alpha(I)\to S_\beta(J)$ имеет место $\beta\geq\alpha$ и, если у I и J есть координата с одинаковым номером, её значение в J не меньше, чем в I.

Запись параллельных зернистых вычислительных процессов. Из условия следует, что Q_3 – число процессов, предназначенных для реализации алгоритма. Единый для каждого из Q_3 процессов псевдокод параллельного алгоритма (без учета операций обмена данными) можно записать следующим образом ($p=k^{gl}$ – номер процесса):

Для каждого процесса Pr_p, $0\leq p\leq Q_3-1$:

```
do  i^gl = 0, Q_1–1
    do  j^gl = 0, Q_2–1
        if  p=0  Tile1(i^gl,j^gl)
        Tile2(i^gl,j^gl,p )
    enddo(j^gl)
enddo(i^gl)
```

В нулевом процессе Pr_0 осуществляются все операции $S_1(i,j)$ и все вычисления алгоритма, для которых $1\leq k\leq r_3$; в процессе Pr_1 осуществляются вычисления, для которых $r_3+1\leq k\leq 2r_3$. Далее в процессе Pr_p, кроме, возможно, (Q_3–1)-го процесса, осуществляются все вычисления алгоритма, для которых $1+p\,r_3\leq k\leq (p+1)r_3$; в процессе с номером Q_3–1 осуществляются вычисления алгоритма, для которых $1+(Q_3-1)r_3\leq k\leq \min(Q_3 r_3, N)$.

Распределение входных и выходных данных. Соответственно распределению вычислений происходит распределение между процессами элементов матриц A и B; согласно заданию варианта 21 элементы матрицы C назначаются процессам динамически. Произвольному процессу Pr_p распределяются столбцы матрицы A и строки матрицы B с номерами с $1+p\,r_3$ по $\min((p+1)r_3, N)$. Нулевой процесс Pr_0 отправляет процессам Pr_p, $p>0$, «свои» части, обозначим их A_p и B_p, матриц A и B. Результаты вычислений – вся матрицы C – окончательно подсчитывается в $\mathrm{Pr}_{Q_3\text{-}1}$.

Общее представление о работе параллельного алгоритма и об обмене данными. Эти рассуждения здесь не приводим (в контрольной работе этот пункт тоже можно опустить).

Выделение массивов. Приватизация массивов. A_p – матрицs размера $N\times r_3$ и B_p – матрицы размера $r_3\times N$ – приватизируются процессом Pr_p, $0 \le p \le Q_2–1$. CP – матрица размера $r_1\times r_2$ используется для трансляции. Результирующая матрица C формируется в процессе $\mathrm{Pr}_{Q_3\text{-}1}$.

Запись тайла с выделенными массивами. Напомним вид тайла Tile1(i^{gl},j^{gl}):

```
            do  i = 1 + i^gl r_1, min((i^gl +1)r_1, N)
                do  j = 1 + j^gl r_2, min(( j^gl +1)r_2, N)
S_1(i,j):               c(i,j)=0
                enddo
            enddo
```

Tile1(i^{gl},j^{gl}) с выделенными массивами:

```
            do  i = 1 + i^gl r_1, min((i^gl +1)r_1, N)
                do  j = 1 + j^gl r_2, min(( j^gl +1)r_2, N)
                    ip=i–i^gl r_1
                    jp= j–j^gl r_2
S_1(i,j):           cp(ip,jp)=0
                enddo
            enddo
```

Напомним вид тайла Tile2(i^{gl},j^{gl},p):

```
                do  i = 1 + i^gl r_1, min((i^gl +1)r_1, N)
                    do  j = 1 + j^gl r_2, min((j^gl+1)r_2, N)
                        do  k = 1+ p r_3, min((p+1)r_3, N)
S_2(i,j,k):                 c(i,j) = c(i,j) + a(i,k) b(k,j)
                        enddo
                    enddo
                enddo
```

Tile2(i^{gl},j^{gl},p) с выделенными массивами:

```
                do  i = 1 + i^gl r_1, min((i^gl +1)r_1, N)
                    ip=i–i^gl r_1
                    do  j = 1 + j^gl r_2, min((j^gl+1)r_2, N)
                        jp= j–j^gl r_2
                        do  k = 1+ p r_3, min((p+1)r_3, N)
                            kp= k–p r_3
S_2(i,j,k):                 cp(ip,jp)= cp(ip,jp)+a_p(i,kp) b_p(kp,j)
                        enddo
                    enddo
                enddo
```

Оптимизация вычислений в тайлах. Оптимизацию вычислений **(**например, вычисление границ цикла следует выполнять вне цикла**)** рассматривать не будем. В контрольной работе этот пункт можно опустить.

Структурирование коммуникаций. Трансляцию данных опишем непосредственно при записи псевдокода.

Псевдокод параллельного зернистого алгоритма.

Для каждого процесса Pr_p, $0 \le p \le Q_3–1$:

```
{if  p=0  сформировать матрицы A_q, 0 ≤ q ≤ Q_3–1,
              send(Pr_q; A_q; N×r_3), 1 ≤ q ≤ Q_3–1,
          сформировать матрицы B_q, 0 ≤ q ≤ Q_3–1,
              send(Pr_q; B_q; r_3×N), 1 ≤ q ≤ Q_3–1}
if  p>0  receive(Pr_0; A_p; N×r_3),  receive(Pr_0; B_p; r_3×N)
do  i^gl = 0, Q_1–1
    do  j^gl = 0, Q_2–1
        if  p=0  Tile1(i^gl,j^gl)
        if  p>0  receive(Pr_{p-1}; CP; r_1×r_2)
        Tile2(i^gl,j^gl,p )
        if  p<Q_3–1  send(Pr_{p+1}; CP; r_1×r_2)
        if  p=Q_3–1  используя CP сформировать часть матрицы C
    enddo(j^gl)
enddo(i^gl)
```