ОРГАНИЗАЦИЯ ПАРАЛЛЕЛЬНЫХ ЗЕРНИСТЫХ ВЫЧИСЛИТЕЛЬНЫХ ПРОЦЕССОВ

(Получение параллельных последовательностей зернистых вычислений)

(без распределения массивов между процессами и указания обменных операций)

Получение параллельных зернистых вычислительных процессов (примеры). Формулировка теоремы о параллельности всех зернистых вычислительных процессов. Формулировка теоремы о допустимости тайлинга, применяемого для получения параллельных зернистых последовательностей вычислений.

Приведем примеры получения параллельных алгоритмов, множества операций которых разбиты на тайлы. Такие параллельные алгоритмы будем называть зернистыми. Покажем, как можно организовать параллельные зернистые вычисления.

Пример 1. Рассмотрим основную часть численного решения двумерной задачи Дирихле для уравнения Пуассона методом верхней релаксации:

```
do  m = 1, M
    do  i = 1, N_x – 1
        do  j = 1, N_y – 1
            y(i,j)=F(y(i–1,j),y(i,j–1),y(i,j),y(i,j+1),y(i+1,j))        (1)
        enddo
    enddo
enddo
```

Все зависимости алгоритма, задаваемого гнездом циклов (1), являются однородными и выражаются векторами зависимостей (0,1,0), (0,0,1), (1,0,0), (1,0,–1), (1,–1,0) (рис. 1).

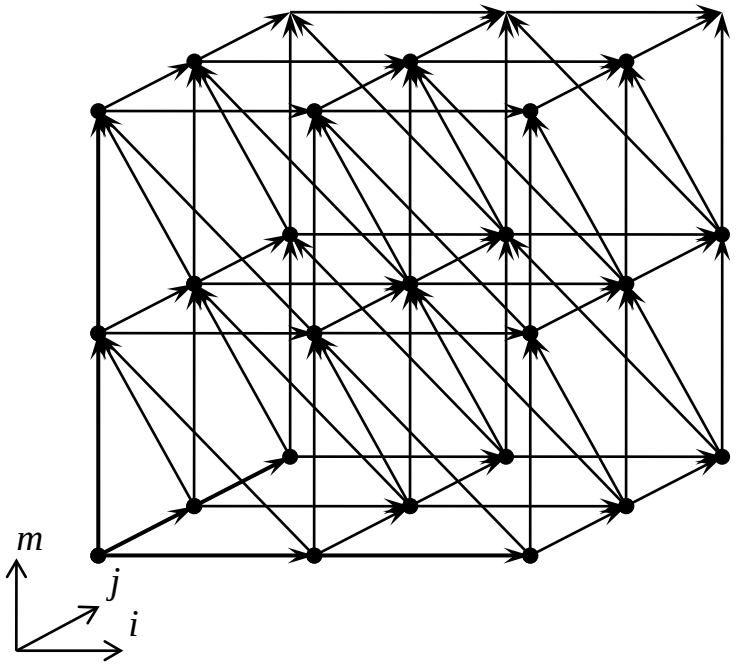


Рисунок 1. Схематичное изображение зависимостей алгоритма (1).

Произведем тайлинг. Будем считать цикл с параметром m глобальным не разбиваемым, а цикл с параметром j локальным не разбиваемым. Разобьем цикл уровня вложенности 2: заменим цикл с параметром i двумерной циклической конструкцией. Обозначим через Q_2 число итераций в первом новом цикле, а через r_2 (наибольшее) число значений параметра i во втором цикле; параметры Q_2 и r_2 связаны соотношениями $Q_2 = \left\lceil \frac{N_x - 1}{r_2} \right\rceil$, $r_2 = \left\lceil \frac{N_x - 1}{Q_2} \right\rceil$. Получим

```
do m = 1, M
   do i^gl = 0, Q_2 –1
   do i = 1 + i^gl r_2, min((i^gl+1)r_2, N_x – 1)
      do j = 1, N_y – 1
         y(i,j)=F(y(i–1,j),y(i,j–1),y(i,j),y(i,j+1),y(i+1,j))
      enddo
   enddo
   enddo
enddo
```

Такое преобразование всегда допустимо (порядок выполнения операций не меняется).

Пусть Q_2 – число процессов (процессоров, ядер), предназначенных для реализации трехмерного цикла (1). Единый для каждого из Q_2 процессов псевдокод параллельного алгоритма можно записать следующим образом (i^{gl} – номер процесса):

```
Для каждого процесса Pr_{i^gl},  0 ≤ i^gl ≤ Q_2–1:
do m = 1, M
   Tile(m,i^gl,0)
   обмен данными (синхронизация)
enddo
```

где вычисления тайла Tile(m,i^{gl},0) задаются двумерным циклом

```
   do i = 1 + i^gl r_2, min((i^gl+1)r_2, N_x – 1)
      do j = 1, N_y – 1
         y(i,j) = F(y(i–1,j),y(i,j–1),y(i,j),y(i,j+1),y(i+1,j))
      enddo
   enddo
```

Каждый процесс i^{gl} на каждой итерации m выполняет операции одного тайла (одного зерна вычислений). Параметры r_2 и $(N_y - 1)$ задают размер зерна вычислений: число итераций, принадлежащих одному полному тайлу равно $r_2 \times (N_y - 1)$. На рис. 2 показано распределение тайлов между процессами. (С

помощью рисунка можно лучше пояснить параллельный алгоритм. Следует обратить внимание, что одновременно могут выполнять операции только половина процессов.)

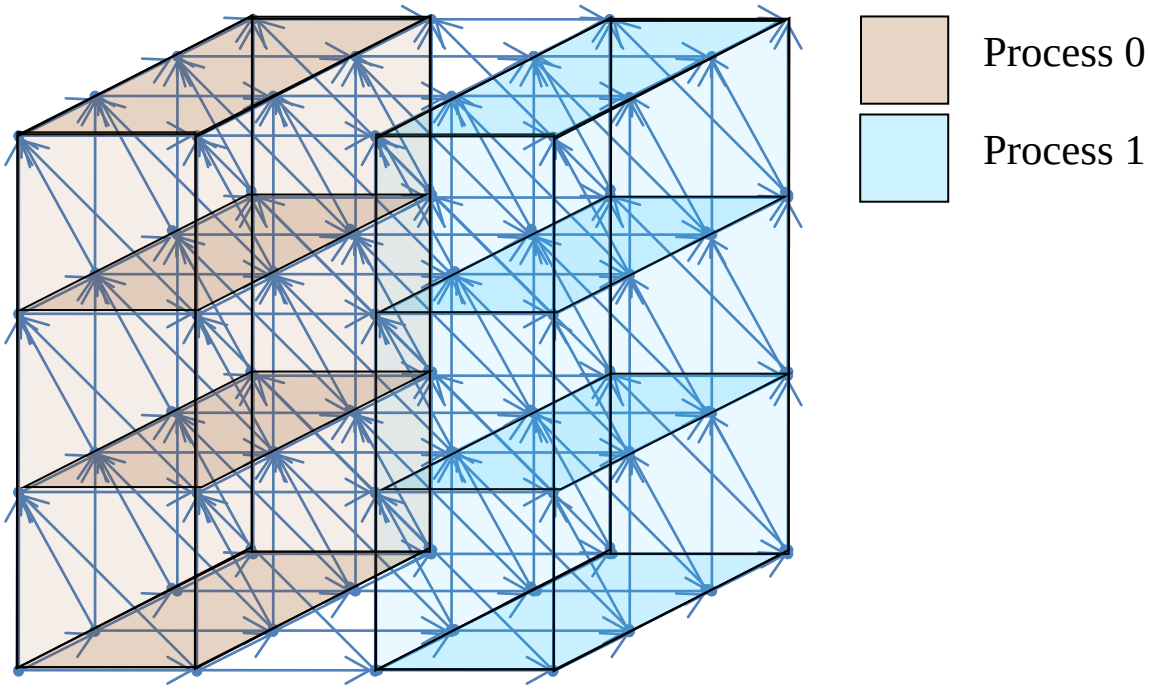


Рисунок 2. Распределение вычислений между процессами, $N_x = N_y = 5$, $M = 4$. Тайлы (выделены более ярким цветом) имеют размер 2×4.

Пример 2. Рассмотрим снова алгоритм (1) и заменим циклы с параметрами i и j (это циклы уровня вложенности 2 и 3) двумерными циклическими конструкциями ($Q_3 = \left\lceil \frac{N_y - 1}{r_3} \right\rceil$):

do $m = 1, M$
  do $i^{gl} = 0, Q_2 - 1$
  do $i = 1 + i^{gl} r_2, \min((i^{gl}+1)r_2, N_x - 1)$
    do $j^{gl} = 0, Q_3 - 1$
    do $j = 1 + j^{gl} r_3, \min((j^{gl}+1)r_3, N_y - 1)$
      $y(i,j)=F(y(i-1,j),y(i,j-1),y(i,j),y(i,j+1),y(i+1,j))$
    enddo
    enddo
  enddo
  enddo
enddo

Такое преобразование всегда допустимо.

Пусть Q_2 – число процессов, предназначенных для реализации трехмерного цикла (1). Переставим циклы с параметрами i и j^{gl} (перестановка циклов не всегда допустима, но в данном случае порядок выполнения

зависимых операций алгоритма не изменится). Единый для каждого из Q_2 процессов псевдокод параллельного алгоритма можно записать следующим образом (i^{gl} – номер процесса):

Для каждого процесса $\mathrm{Pr}_{i^{gl}}$, $0 \le i^{gl} \le Q_2 - 1$:

do $m = 1, M$
  do $j^{gl} = 0, Q_3 - 1$
    do $i = 1 + i^{gl} r_2, \min((i^{gl}+1) r_2, N_x - 1)$ // Начало тайла Tile(m, i^{gl}, j^{gl})
      do $j = 1 + j^{gl} r_3, \min((j^{gl}+1) r_3, N_y - 1)$
        $y(i,j) = F(y(i-1,j), y(i,j-1), y(i,j), y(i,j+1), y(i+1,j))$ (2)
      enddo
    enddo // Конец тайла Tile(m, i^{gl}, j^{gl})
    обмен данными (синхронизация)
  enddo
enddo

Каждый процесс i^{gl} на каждой итерации m выполняет операции Q_3 тайлов (зерен вычислений); одному тайлу соответствуют два самых внутренних цикла. Параметры r_2 и r_3 задают размер зерна: число итераций, принадлежащих одному полному тайлу, равно $r_2 \times r_3$. Можно показать, что объём данных, который передается при выполнении межпроцессорной коммуникационной операции, характеризует параметр r_3. Поэтому далее мы размер зерна вычислений будем задавать целым положительным числом r_3. На рис. 3 показано распределение тайлов между процессами. (С помощью рисунка пояснить параллельный алгоритм.)

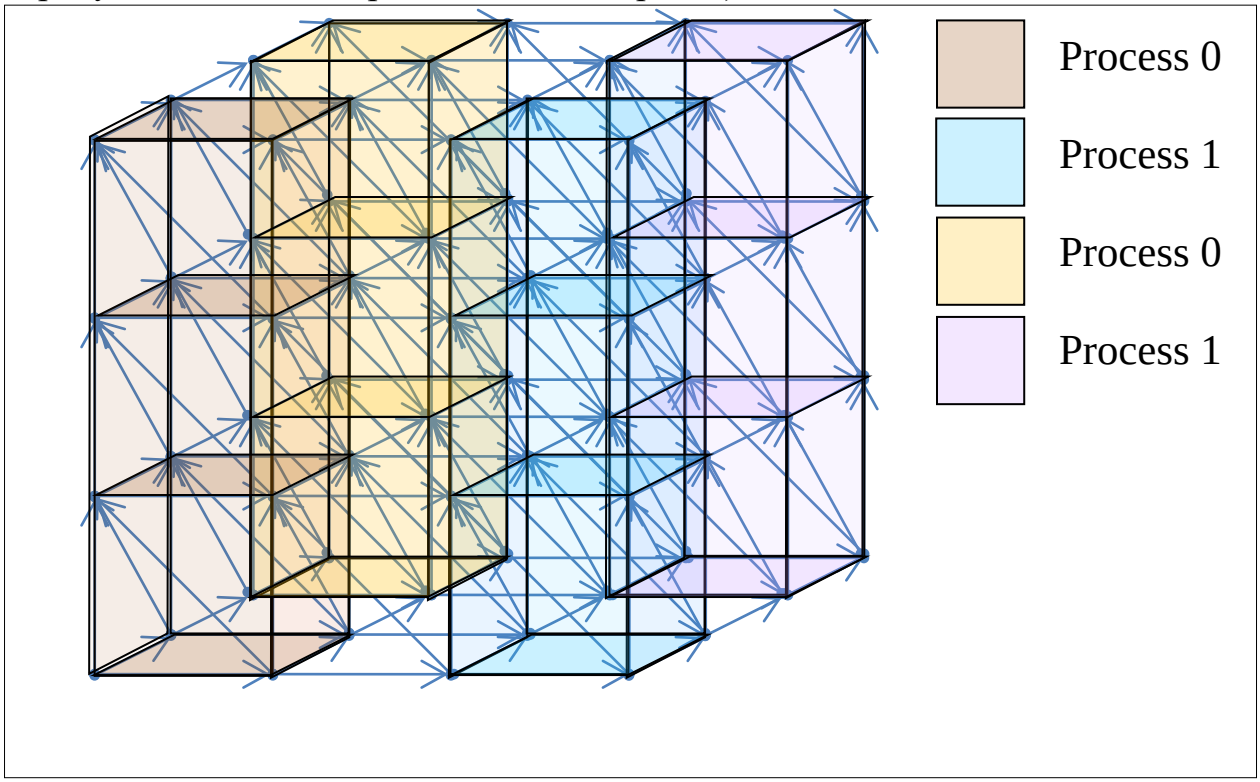


Рисунок 3. Распределение вычислений алгоритма (2) между процессами, $N_x = N_y = 5$, $M = 4$. Тайлы (выделены более ярким цветом) имеют размер 2×2.

Наличие векторов зависимостей (0,1,0) и (1,–1,0) приводит к обмену данными по второй координате, отвечающей за номер процессора, как в прямом, так и в обратном направлении. Обмен данными между процессорами можно вдвое уменьшить, если к алгоритму (1) применить аффинное преобразование $(m,i,j)\to(m,m+i,j)$. Такое преобразование называется скашиванием цикла и приводит к гнезду циклов

do $m = 1, M$
  do $i' = m+1, N_x + m - 1$
    do $j = 1, N_y - 1$
      $y(i'-m,j)=F(y(i'-m-1,j),y(i'-m,j-1),u(i'-m,j),y(i'-m,j+1),y(i'-m+1,j))$
    enddo
  enddo
enddo

Теперь зависимости задаются векторами (0,1,0), (0,0,1), (1,1,0), (1,1,–1), (1,0,0), поэтому по второй координате, отвечающей за номер процесса, обмен данными в обратном направлении отсутствует. Для этого алгоритма, по аналогии с алгоритмом (2), можно записать псевдокод, который далее мы будем называть преобразованным алгоритмом; алгоритм (2) будем называть исходным алгоритмом.

Сравним время выполнения на суперкомпьютере СКИФ К-1000 исходного и преобразованного (теоретически лучшего для реализации на параллельных компьютерах с распределенной памятью) алгоритмов, исследуем влияние размера зерна вычислений на скорость выполнения алгоритмов. Суперкомпьютер СКИФ К-1000 был установлен в ОИПИ НАН Беларуси; в списке Top 500 самых мощных суперкомпьютеров мира занимал 99-ю позицию (ноябрь 2004 г.) с характеристиками Rmax=2 Тфлопс, Rpeak=2,5 Тфлопс (576 ядер, процессоры суперкомпьютера одноядерные).

Все вычисления, результаты которых представлены в этом разделе, проводились при следующих значениях параметров алгоритмов: $N_x=N_y=$ 10 000, M=100. Результаты вычислительных экспериментов при размере зерна, равном 2000, представлены на рис. 4 и рис. 5. Из рисунков видно, что применение к алгоритму преобразования привело к значительному улучшению характеристик.

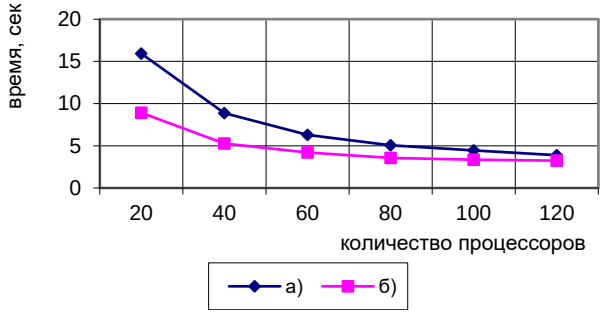


Рисунок 4 – График зависимости времени решения задачи от количества процессоров (зерно вычислений равно 2000)
а) исходный алгоритм б) преобразованный алгоритм

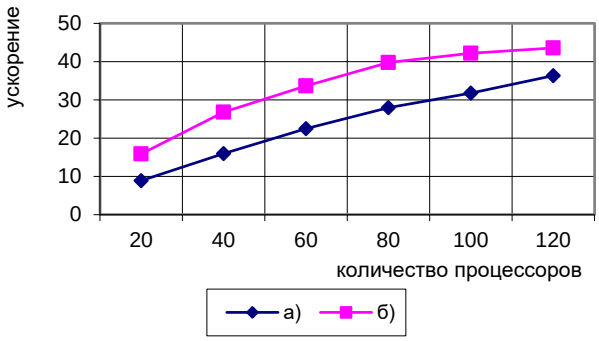


Рисунок 5 – График зависимости ускорения вычислений от количества процессоров (зерно вычислений равно 2000)
а) исходный алгоритм б) преобразованный алгоритм

Проведем теперь вычисления с размером зерна, равным 200 (рис. 6 и рис. 7). Видно, что преобразование гнезда циклов в этом случае не дало ощутимого преимущества. При этом характеристики улучшились, по сравнению с использованием зерна размера 2000, как для исходного алгоритма (причем значительно), так и для преобразованного.

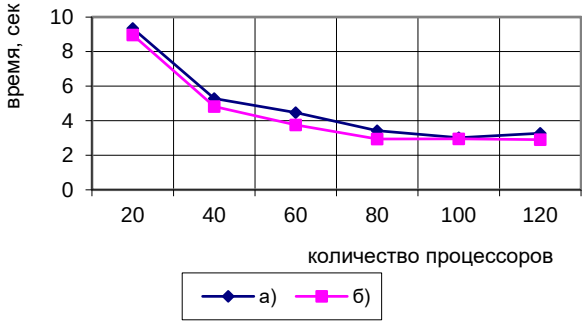


Рисунок 6 – График зависимости времени решения задачи от количества процессоров (зерно вычислений равно 200)
а) исходный алгоритм б) преобразованный алгоритм

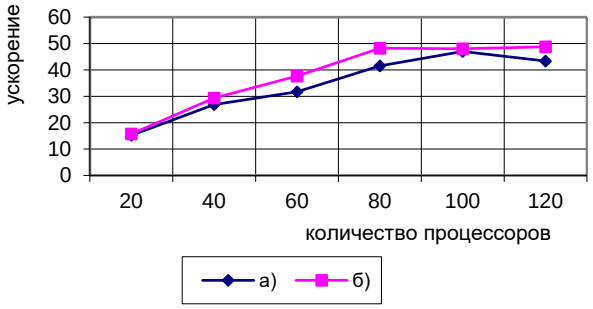


Рисунок 7 – График зависимости ускорения вычислений от количества процессоров (зерно вычислений равно 200)
а) исходный алгоритм б) преобразованный алгоритм

Таким образом, размер зерна вычислений может сильно влиять на время вычислений. Было проведено экспериментальное нахождение оптимального размера зерна вычислений при фиксированном (двадцать и сорок) количестве процессоров. Результаты представлены на рис. 8 и рис. 9.

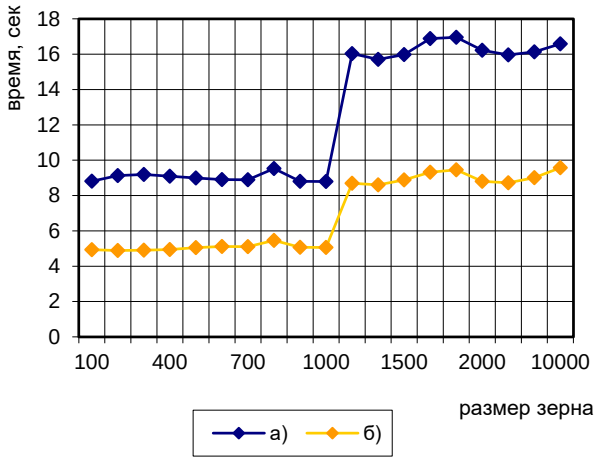


Рисунок 8 – График зависимости времени реализации исходного алгоритма от размера зерна вычислений
а) на 20 процессорах б) на 40 процессорах

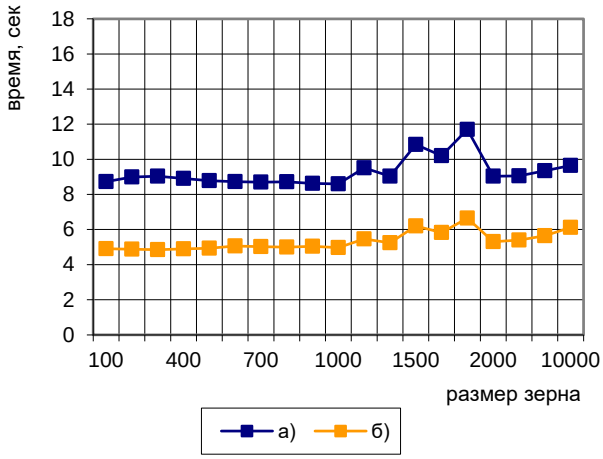


Рисунок 9 – График зависимости времени реализации преобразованным алгоритмом от размера зерна вычислений
а) на 20 процессорах б) на 40 процессорах

Для исходного алгоритма существует порог, после которого увеличение размера зерна приводит к резкому увеличению времени вычислений. Для преобразованного алгоритма размер зерна меньше влияет на время вычислений, но в обоих случаях следует выбирать зерно размером до 1000. При удачном выборе размера зерна вычислений скашивание цикла (уменьшившее число коммуникаций) может не дать ощутимого уменьшения времени решения задачи. Однако при не оптимальном выборе размера зерна вычислений выигрыш от использования преобразования может быть значительным.

Пример 3. Если циклы в представлении алгоритма не являются тесно

вложенными, то тайлинг, а следовательно и организация параллельных зернистых вычислительных процессов, могут оказаться технически сложнее. Рассмотрим *ijk*-алгоритм перемножения двух квадратных матриц порядка N:

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

Разобьем циклы с параметрами i и j; цикл с параметром k будем считать глобальным не разбиваемым:

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

Здесь параметры r_1 и r_2 задают размеры тайлов, числа Q_1, и Q_2 задают число тайлов; $Q_1 = \left\lceil \frac{N_1}{r_1} \right\rceil$, $Q_2 = \left\lceil \frac{N_2}{r_2} \right\rceil$.

Цикл с параметром k^{gl} не может быть внутренним по отношению к локальным циклам, но 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 (допустимые в данном случае преобразования, так как порядок выполнения зависимых операций алгоритма не изменится):

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

Пусть число процессов, предназначенных для реализации алгоритма, равно Q_2. Единый для каждого процесса псевдокод параллельного алгоритма можно записать следующим образом:

Для каждого процесса $\mathrm{Pr}_{j^{gl}}$, $0 \le j^{gl} \le Q_2 - 1$:

```
do  i^gl = 0, Q1 –1
        Tile1(i^gl,j^gl)
        обмен данными (синхронизация)
        do  k^gl = 1, N
            Tile2(i^gl,j^gl,k^gl)
            обмен данными (синхронизация)
        enddo
enddo
```

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

```
        do  i = 1 + i^gl r1, min((i^gl +1)r1, N)
            do  j = 1 + j^gl r2, min((j^gl+1)r2, N)
                c(i,j) =0
            enddo
        enddo
```

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

```
        do  i = 1 + i^gl r1, min((i^gl+1)r1, N)
            do  j = 1 + j^gl r2, min((j^gl+1)r2, N)
                k = k^gl
                c(i,j) = c(i,j) + a(i,k) b(k,j)
            enddo
        enddo
```

Теоретическое обоснование. Введем в рассмотрение последовательности зернистых (т.е. разбитых на тайлы) вычислений. Каждая последовательность вычислений отождествляется с некоторым упорядоченным множеством операций алгоритма. Если последовательности вычислений можно выполнять одновременно, то их называют параллельными. Задание параллельных последовательностей зернистых вычислений позволяет организовать параллельные зернистые вычислительные процессы на компьютерах с распределенной памятью.

Будем рассматривать следующий способ получения параллельных последовательностей зернистых вычислений. К одной последовательности отнесем вычисления тайлов с одинаковыми значениями некоторых функций $\Pr^{\vartheta}\left(J^{gl}\right)$, $1 \le \vartheta \le \Theta$, $J^{gl} \in V^{\vartheta,gl}$, $V^{\vartheta,gl} = \left\{ J^{gl}(j_1^{gl},...,j_{n^\vartheta}^{gl}) \mid \quad 0 \le j_\zeta^{gl} \le Q_\zeta^\vartheta - 1, \quad 1 \le \zeta \le n^\vartheta \right\}$, отображающих тайлы на процессы. Для каждого ϑ зафиксируем число ξ, $1 \le \xi \le n^\vartheta$. Будем полагать

$$\Pr^{\vartheta}\left(J^{gl}(j_1^{gl},...,j_{n^\vartheta}^{gl})\right) = j_\xi^{gl} \tag{3}$$

или

$$\Pr^{\vartheta}\left(J^{gl}(j_1^{gl},...,j_{n^\vartheta}^{gl})\right) = Q_\xi^\vartheta - j_\xi^{gl} - 1. \tag{4}$$

Функция $\Pr^{\vartheta}$ вида (3) или (4) задает вычислительный процесс $\Pr^{\vartheta}\left(J^{gl}\right)$ выполнения операций тайлов $V_{J^{gl}}^{\vartheta}$ с одинаковыми значениями ξ-й координаты j_ξ^{gl} векторов J^{gl}. Этот вычислительный процесс будем обозначать $\Pr_{j_\xi^{gl}}^{\vartheta}$. Будем называть ξ-ю координату s-координатой. Операторы, расположенные непосредственно перед циклом с параметром j_ξ^{gl}, выполняются процессом $\Pr_0^{\vartheta}$, если используется функция вида (3), и процессом $\Pr_{Q_\xi^{gl}-1}^{\vartheta}$, если используется функция вида (4). Операторы, расположенные сразу после цикла с параметром j_ξ^{gl}, выполняются процессом $\Pr_{Q_\xi^{gl}-1}^{\vartheta}$, если используется функция вида (3), и процессом $\Pr_0^{\vartheta}$, если используется функция вида (4).

Функции (3) и (4) являются частными случаями функции вида

$$\Pr^{\vartheta}\left(J^{gl}(j_1^{gl},...,j_{n^\vartheta}^{gl})\right) = \chi^\vartheta j_\xi^{gl} + \delta^\vartheta\left(Q_\xi^\vartheta - 1\right) + \gamma^\vartheta, \tag{5}$$

$$\chi^\vartheta \in \{-1;1\},\ \delta^\vartheta \in \{0;1\},\ \gamma^\vartheta \in \mathbf{Z}.$$

Примем, что множество операций любого тайла выполняется за одну единицу времени, и все результаты и аргументы вычислений одного тайла в следующий момент времени могут участвовать в вычислениях любого другого тайла. Определим продолжительность разгона вычислений – число единиц времени, когда еще не все процессы начали выполняться. Продолжительность торможения вычислений – число единиц времени, когда еще не все процессы закончили выполняться.

Сформулируем достаточные условия, при выполнении которых все зернистые вычислительные процессы являются параллельными

(доказательство не приводится).

Теорема 1. *Пусть тайлинг является допустимым и для каждого набора операторов выполнено хотя бы одно из условий:*

- *по крайней мере для одного цикла, допускающего разбиение, имеет место* $Q_i^{\vartheta} > 1$, $i \neq \xi$, *причем* $Q_i^{\vartheta} >> Q_{\xi}^{\vartheta}$;
- *цикл с параметром* j_{ξ}^{gl} *является параллельным.*

Тогда, без учета разгона и торможения вычислений, все зернистые вычислительные процессы в каждый момент времени загружены работой; продолжительности разгона и торможения вычислений намного меньше общей продолжительности вычислений.

В рассмотренном примере 2 имеем (индекс ϑ, если это не вызывает путаницы, указывать не будем): $\xi = 2$, $\Pr\left(J^{gl}(m, i^{gl}, j^{gl})\right) = i^{gl}$, $Q_2 > 1$, $Q_3 > 1$. Достаточное условие теоремы 1 параллельности всех вычислительных процессов выполняется, если выбрать $Q_3 >> Q_2$.

В примере 3 $\xi = 2$, $\Pr^1\left(J^{gl}(i^{gl}, j^{gl})\right) = j^{gl}$, $\Pr^2\left(J^{gl}(i^{gl}, j^{gl}, k^{gl})\right) = j^{gl}$. Все вычислительные процессы являются параллельными, так как цикл с параметром j^{gl} является параллельным (нет зависимостей по циклу с параметром j).

Тайлинг является допустимым, если выполняются условия теоремы 1 или теоремы 2 лекции «Тайлинг».

Обозначим через $c_{\vartheta^{\alpha,\beta}}$ количество общих для операторов S_α и S_β разбиваемых циклов, j_{k_ζ}, $1 \le \zeta \le c_{\vartheta^{\alpha,\beta}}$, – параметры этих циклов. Следующее утверждение устанавливает более слабые, чем в теореме 2 лекции «Тайлинг» достаточные условия допустимости тайлинга, применяемого для получения параллельных зернистых последовательностей вычислений.

Теорема 2. *Пусть не разбиваемый цикл является глобальным, если не существует внешнего по отношению к нему разбиваемого цикла, и является локальным, если существует внешний по отношению к нему разбиваемый цикл. Пусть существует хотя бы одна такая зависимость* $S_\alpha(I(i_1,...,i_{n_\alpha})) \to S_\beta(J(j_1,...,j_{n_\beta}))$, *что в окружении операторов* S_α *и* S_β *имеется хотя бы один общий разбиваемый цикл и равны параметры всех внешних не разбиваемых циклов:* $(i_1,...,i_{k_1-1}) = (j_1,...,j_{k_1-1})$, $k_1 \neq 1$. *Тайлинг является допустимым, если для любой такой зависимости выполняются следующие условия:*

$$j_{k_\zeta} \ge i_{k_\zeta},\ 2 \le \zeta \le c_{\vartheta^{\alpha,\beta}}, \tag{6}$$

$$\beta \ge \alpha. \tag{7}$$

При отсутствии указанных зависимостей тайлинг допустим.

Доказательство. При тайлинге (в предположениях теоремы) может меняться относительно друг друга порядок выполнения операций, зависящих от параметров одних и тех же разбиваемых циклов. Поэтому порядок выполнения зависимых операций может нарушиться только для таких

зависимостей $S_\alpha(I) \to S_\beta(J)$, что операторы S_α и S_β окружены одним или несколькими общими разбиваемыми циклами и совпадают значения координат векторов I и J, относящиеся к внешним, по отношению к разбиваемым, циклам. Таким образом, следует учитывать только зависимости, указанные в предположениях утверждения.

Покажем, что после применения к исходному многомерному циклу преобразования тайлинга порядок выполнения зависимых операций не изменится.

Напомним, что при тайлинге, с точки зрения преобразования циклов, каждый разбиваемый цикл заменяется на глобальный и локальный циклы, общие локальные циклы распределяются между циклами каждого набора операторов, локальные циклы переставляются с глобальными и становятся по отношению к ним внутренними. Условия (6) являются достаточными для перестановки указанных циклов, а условия (7) достаточны для распределения циклов. □

Отметим, что для тайлинга исходного алгоритма примера 2 достаточные условия допустимости тайлинга теоремы 2 лекции «Тайлинг» не выполняются, а достаточные условия доказанной теоремы выполняются.

Рассмотрим следующий из теоремы 2 способ проверки корректности тайлинга.

Для каждой пары итераций *I* и *J*, порождающих зависимость между операциями $S_\alpha(I) \to S_\beta(J)$:

- определить общие разбиваемые циклы;
- если $c_{\vartheta^{\alpha,\beta}} \le 1$, то зависимость не нарушает корректности тайлинга;
- если $c_{\vartheta^{\alpha,\beta}} \ge 2$, то определить все k_ζ, $1 \le \zeta \le c_{\vartheta^{\alpha,\beta}}$;
- если $(i_1,...,i_{k_1-1}) \ne (j_1,...,j_{k_1-1})$, то зависимость не нарушает корректности тайлинга;
- если $(i_1,...,i_{k_1-1}) = (j_1,...,j_{k_1-1})$, то проверить справедливость условий $j_{k_\zeta} \ge i_{k_\zeta}$, $2 \le \zeta \le c_{\vartheta^{\alpha,\beta}}$, и $\beta \ge \alpha$ (или $\vartheta^\beta \ge \vartheta^\alpha$).

О выборе s-координаты – файл «Локальность паралл».

Итерационные процессы для решения разностной задачи Дирихле

Рассмотрим двумерную задачу Дирихле для уравнения Пуассона в прямоугольной области $G=[0<x_1<l_1]\times[0<x_2<l_2]$ с границей Г:

$$\frac{\partial^2 u}{\partial x_1^2} + \frac{\partial^2 u}{\partial x_2^2} = f(x_1,x_2),\ (x_1,x_2) \in G,$$
$$u(x_1,x_2)|_\Gamma = \mu(x_1,x_2).$$

Величины l_1, l_2 и функции $f(x, x_2)$, $\mu(x_1,x_2)$ считаются известными, функция $u(x_1,x_2)$ – искомая.

Эта задача может рассматриваться как задача на отыскание деформации упругой мембраны, находящейся под внешним воздействием, края которой закреплены по заданному закону, либо как задача об отыскании стационарного распределения температур на прямоугольной пластине с внешними источниками тепла, на краях которой поддерживается заданный температурный режим.

Для численного решения задачи введем в области $G+\Gamma$ сетку узлов

$$\{ih_1,\ i=0,1,\ldots,N_1,\ N_1h_1=l_1,\ jh_2,\ j=0,1,\ldots,N_2,\ N_2h_2=l_2\}.$$

Наряду с обозначениями N_1, N_2 далее будем использовать также обозначения N_x, N_y.

Используя пятиточечный шаблон «крест» для вычисления приближенных значений производных и сеточное представление функций, запишем разностную задачу Дирихле:

$$\frac{y_{i+1,j}-2y_{i,j}+y_{i-1,j}}{h_1^2}+\frac{y_{i,j+1}-2y_{i,j}+y_{i,j-1}}{h_2^2}=f_{i,j},\quad i=1,\ldots,N_1-1,\ j=1,\ldots,N_2-1,$$

$$y|_{\gamma_h}=\mu(x_1,x_2)|_{\gamma_h},$$

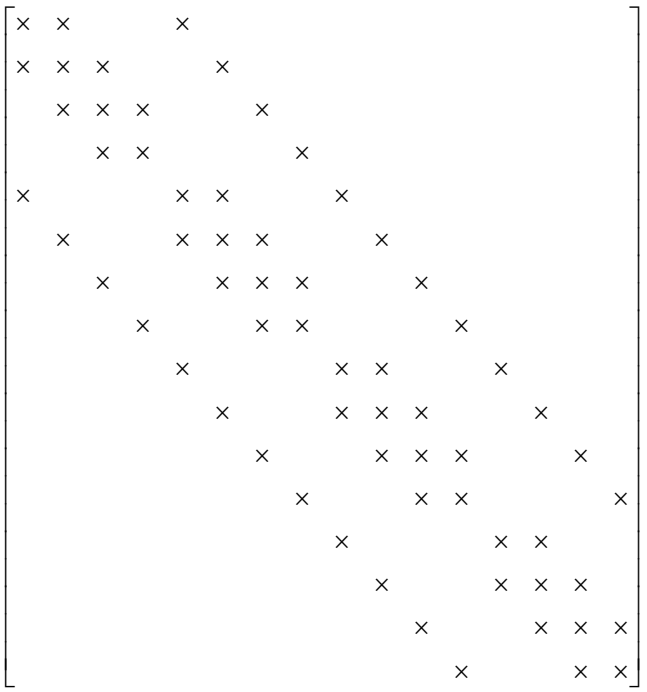
где γ_h – граничные узлы (кроме угловых узлов, они не используются).

Эта разностная задача порождает СЛАУ относительно вектора неизвестных

$$\left(y_{1,1},\ldots,y_{1,N_2-1},y_{2,1},\ldots,y_{2,N_2-1},\ \ldots\ ,y_{N_1-1,1},\ldots,y_{N_1-1,N_2-1}\right).$$

Матрица системы имеет порядок $(N_1-1)(N_2-1)$, т.е. имеет размеры $((N_1-1)(N_2-1))\times((N_1-1)(N_2-1))$.

На рисунке изображён портрет матрицы (отмечены позиции ненулевых элементов) для случая $N_1-1=4$, $N_2-1=4$:



Матрица имеет блчно-трёхдиагональную структуру. На главной диагонали – N_1–1 трёхдиагональных матриц-блоков размеров (N_2–1)×(N_2–1), на поддиагонали и на наддиагонали – N_1–2 диагональных матриц-блоков размеров (N_2–1)×(N_2–1).

Если полученную систему линейных алгебраических уравнений решать каким-либо прямым методом, то в процессе матричных преобразований общее число ненулевых элементов возрастает. Резко возрастают и вычислительные затраты, требуемый объём памяти. Поэтому для достаточно подробных сеток полученную систему обычно решают итерационными методами. Перед началом вычислений нужно задать начальное (например, нулевое) приближение $y_{i,j}^0$ во внутренних точках сетки и заполнить значения $y_{i,j}^0$ в граничных узлах точными значениями $\mu_{i,j}$. В силу специфики матрицы ее можно не хранить целиком, а вычислять ее коэффициенты в процессе численного решения. Это позволяет применять итерационные методы, запоминая лишь значения сеточной функции на очередной итерации.

Итерационный процесс Якоби имеет следующий вид:

$$y_{i,j}^l = \left(\frac{2}{h_1^2}+\frac{2}{h_2^2}\right)^{-1}\left(\frac{y_{i+1,j}^{l-1}+y_{i-1,j}^{l-1}}{h_1^2}+\frac{y_{i,j+1}^{l-1}+y_{i,j-1}^{l-1}}{h_2^2}-f_{i,j}\right),$$

$$i=1,\ldots,N_1-1,\ j=1,\ldots,N_2-1,\ \ l=1,2,\ldots$$

Основная часть псевдокода процесса Якоби ($h_1=h_2=h$):

```
do l = 1, r_it // r_it – некоторое фиксированное число итераций
    do i = 1, N_1–1 // N_1–1 – число строк матрицы, задающей
                            внутренние узлы сетки
        do j = 1, N_2–1 // N_2–1 – число столбцов матрицы, задающей
                                внутренние узлы сетки
            y(i,j)=0,25(low(i–1,j)+low(i,j–1)+low(i,j+1)+low(i+1,j)–h^2 f(i,j))
        enddo
    enddo
    do i = 1, N_1–1
        do j = 1, N_y–1
            low(i,j)=y(i,j)
        enddo
    enddo
enddo
```

Итерационный процесс Гаусса-Зейделя имеет следующий вид:

$$y_{i,j}^l = \left(\frac{2}{h_1^2}+\frac{2}{h_2^2}\right)^{-1}\left(\frac{y_{i+1,j}^{l-1}+y_{i-1,j}^{l}}{h_1^2}+\frac{y_{i,j+1}^{l-1}+y_{i,j-1}^{l}}{h_2^2}-f_{i,j}\right),$$

Основная часть псевдокода процесса Гаусса-Зейделя ($h_1=h_2=h$):

do l = 1, r_{it}
  do i = 1, N_1–1
    do j = 1, N_2–1
      $y(i,j)=0{,}25(y(i-1,j)+y(i,j-1)+y(i,j+1)+y(i+1,j)-h^2f(i,j))$
    enddo
  enddo
enddo

Метод Гаусса-Зейделя является частным случаем более употребительного на практике метода верхней релаксации (SOR):

$$y_{i,j}^{l}=\omega\left(\frac{2}{h_1^2}+\frac{2}{h_2^2}\right)^{-1}\left(\frac{y_{i+1,j}^{l-1}+y_{i-1,j}^{l}}{h_1^2}+\frac{y_{i,j+1}^{l-1}+y_{i,j-1}^{l}}{h_2^2}-f_{i,j}\right)+(1-\omega)\,y_{i,j}^{l-1},$$

Обозначим

$$F(y_{i-1,j},y_{i,j-1},y_{i,j},y_{i,j+1},y_{i+1,j})=$$
$$\omega\left(\frac{2}{h_1^2}+\frac{2}{h_2^2}\right)^{-1}\left(\frac{y_{i+1,j}+y_{i-1,j}}{h_1^2}+\frac{y_{i,j+1}+y_{i,j-1}}{h_2^2}-f_{i,j}\right)+(1-\omega)\,y_{i,j}.$$

Тогда основную часть численного решения двумерной задачи Дирихле для уравнения Пуассона методом верхней релаксации можно представить в следующем виде:

do l = 1, r_{it} // r_{it} – некоторое фиксированное число итераций
  do i = 1, N_1–1 // N_1–1 – число строк матрицы, задающей внутренние узлы сетки
    do j = 1, N_2–1 // N_2–1 – число столбцов матрицы, задающей внутренние узлы сетки
      $y(i,j)=F(y(i-1,j),y(i,j-1),y(i,j),y(i,j+1),y(i+1,j))$
    enddo
  enddo
enddo