ТАЙЛИНГ

Разбиение множества операций алгоритма на макрооперации-тайлы; перестановка циклов, распределение циклов. Примеры использования тайлинга. Допустимость тайлинга.

Тайлинг (tiling) – это преобразование алгоритма для получения макроопераций. Получаемые в результате преобразования макрооперации называются зерном вычислений или тайлами. Множество операций алгоритма, составляющих зерно вычислений, выполняется атомарно, как одна единица вычислений; при параллельных вычислениях выполнение операций одного зерна не может прерываться синхронизацией или обменом данными. Разбиение множества операций алгоритма на тайлы позволяет при подходящем задании тайлов организовать параллельные вычислительные процессы и уменьшить накладные расходы на обмен данными при параллельных вычислениях, уменьшить накладные расходы на использование памяти при последовательных вычислениях.

Использование тайлинга приводит, в частности, к известным блочным алгоритмам. Но идея тайлинга не в разбиении на блоки массивов данных, а в разбиении множества операций алгоритма. Вариантов разбиения множества вычислений, вообще говоря, гораздо больше, чем вариантов разбиения массивов данных. Это связано с тем, что размерности итерационных пространств алгоритмов обычно превосходят размерности массивов данных.

При тайлинге каждый цикл разбивается на два цикла: глобальный, параметр которого определяет на данном уровне вложенности порядок вычисления тайлов, и локальный, в котором параметр исходного цикла изменяется в границах одного тайла. Допускается вырожденное разбиение цикла, при котором все итерации относятся к глобальному циклу или все итерации относятся к локальному циклу. Такие циклы будем называть соответственно глобальными не разбиваемыми и локальными не разбиваемыми.

По смыслу тайлинга, для каждого набора операторов локальные циклы должны быть самыми внутренними. Поэтому общие для операторов локальные циклы распределяются между циклами операторов, если только операторы не окружены одинаковым наборов циклов; локальные циклы переставляются с глобальными и становятся самыми внутренними.

Приведем примеры, иллюстрирующие технику тайлинга, и примеры применения тайлинга.

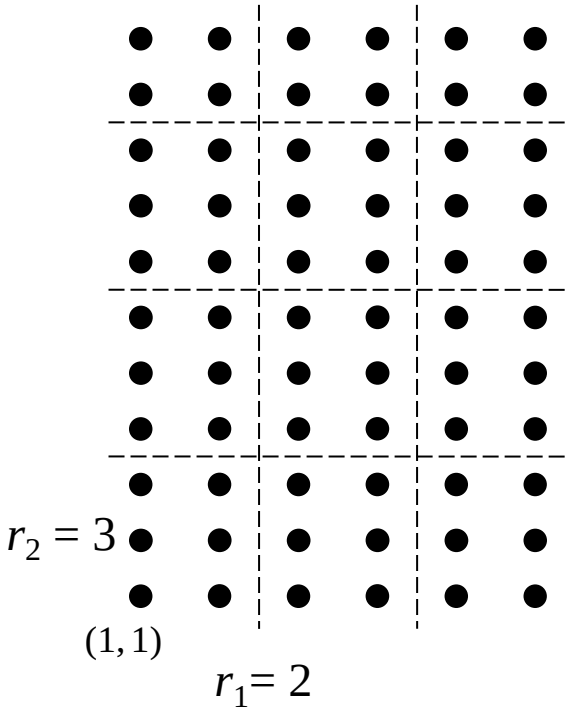
Пример 1 (разбиение всех циклов; два вида записи алгоритма после тайлинга; перестановка и распределение циклов при тайлинге). Рассмотрим два вложенных цикла:

do i = 1, N_1
  do j = 1, N_2
    $S(i, j)$
  enddo
enddo

Пусть параметры r_1 и r_2 характеризуют размеры тайлов, числа Q_1 и Q_2 характеризуют число тайлов; $Q_1 = \left\lceil \frac{N_1}{r_1} \right\rceil$, $Q_2 = \left\lceil \frac{N_2}{r_2} \right\rceil$ ($\lceil \cdot \rceil$ обозначает ближайшее «сверху» целое число). Разобьем каждый цикл на глобальный и локальный циклы (пояснения для случая N_1=6, N_2=11 приведены на рисунке, каждой итерации алгоритма поставлена в соответствие точка (i,j)):

do i^{gl} = 0, Q_1–1
do i = 1 + $i^{gl} r_1$, min((i^{gl}+1) r_1, N_1)
  do j^{gl} = 0, Q_2 –1
  do j = 1 + $j^{gl} r_2$, min((j^{gl}+1)r_2, N_2)
    $S(i, j)$
  enddo
  enddo
enddo
enddo

Отметим, что порядок вычисления операций после разбиения циклов не изменился.



После перестановки циклов (будем считать это преобразование алгоритма корректным) с параметрами i и j^{gl} (локальные циклы должны быть самыми внутренними) получим

do $i^{gl} = 0, Q_1 - 1$
  do $j^{gl} = 0, Q_2 - 1$
    do $i = 1 + i^{gl} r_1, \min((i^{gl}+1)\, r_1, N_1)$
      do $j = 1 + j^{gl} r_2, \min((j^{gl}+1) r_2, N_2)$
        $S(i, j)$
      enddo
    enddo
  enddo
enddo

Глобальные циклы определяют порядок вычисления тайлов, локальные циклы определяют порядок вычисления итераций исходного алгоритма в границах одного тайла:

do $i^{gl} = 0, Q_1 - 1$
  do $j^{gl} = 0, Q_2 - 1$
    Tile(i^{gl}, j^{gl}) \\ макрооперация-тайл включает выполнение $r_1 \times r_2$ итераций (при $i^{gl} = Q_1 - 1$ или $j^{gl} = Q_2 - 1$ тайл может включать меньше итераций
  enddo
enddo

Вычисления одного тайла Tile(i^{gl}, j^{gl}) задаются двумерным циклом

do $i = 1 + i^{gl} r_1, \min((i^{gl}+1)\, r_1, N_1)$
  do $j = 1 + j^{gl} r_2, \min((j^{gl}+1) r_2, N_2)$
    $S(i, j)$
  enddo
enddo

Заметим, что в литературе используется также запись, в которой параметры глобальных циклов изменяются с шагом, определяемым размерами тайла:

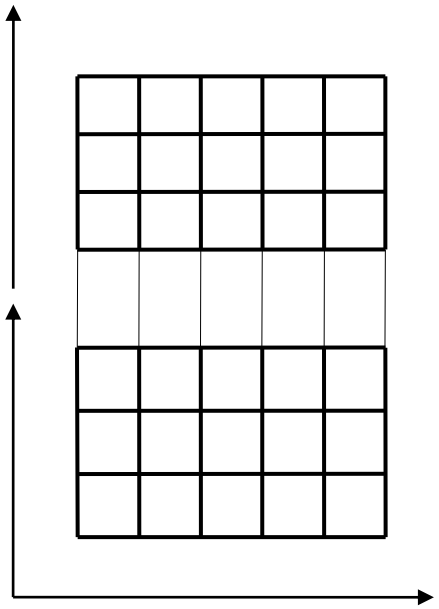
do $i^{gl} = 1, N_1, r_1$
  do $j^{gl} = 1, N_2, r_2$
    do $i = i^{gl}, \min(i^{gl} + r_1 - 1, N_1)$
      do $j = j^{gl}, \min(j^{gl} + r_2 - 1, N_2)$
        $S(i, j)$

```
            enddo
        enddo
    enddo
enddo
```

Рассмотрим случай, когда два оператора окружены не совпадающим набором циклов:

```
do i = 1, N
    do j = 1, N
        S1(i, j)
    enddo
    do j = 1, N
        S2(i, j)
    enddo
enddo
```

На рисунке схематично изображены тайлы итераций для первого (ниже) и второго (выше) операторов. Для параметров i ось общая, для параметров j оси разные.



Разобьем каждый цикл на глобальный и локальный циклы:

```
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)
```

```
    enddo
    enddo
    do j^gl = 0, Q2 –1
    do j = 1 + j^gl r2, min((j^gl+1)r2, N)
        S2(i, j)
    enddo
    enddo
enddo
enddo
```

Для того чтобы переставить циклы с параметрами i и j^{gl} (локальные циклы должны быть самыми внутренними) требуется сначала распределить цикл с параметром i между разными наборами циклов выполняемых операторов (как и ранее, будем считать все преобразования алгоритма допустимыми):

```
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)
        enddo
        enddo
    enddo(i)
    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)
            S2(i, j)
        enddo
        enddo
    enddo(i)
enddo
enddo
```

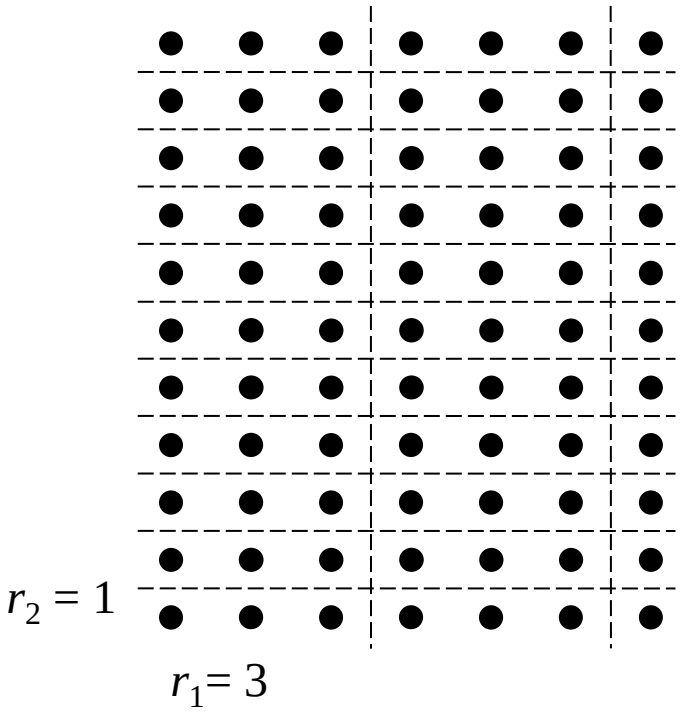
После перестановки циклов с параметрами i и j^{gl} окончательно получим

```
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)
                S1(i, j)
```

```
            enddo
        enddo(i)
    enddo(j^gl)
    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_2(i, j)
            enddo
        enddo(i)
    enddo(j^gl)
enddo
```

Отметим преобразования циклов, применяемые при тайлинге: каждый цикл заменяется на глобальный и локальный циклы, общие локальные циклы распределяются между операторами (если циклы не являются тесно вложенными), локальные циклы переставляются с глобальными и становятся по отношению к ним внутренними.

Пример 2 (не разбиваемые глобальные циклы, не разбиваемые локальные циклы). Пусть в рассмотренном первом примере r_2=1; тогда Q_2=N_2 (пояснения для случая N_1=7, N_2=11 приведены на рисунке).




```
do i^gl = 0, Q_1 –1
do i = 1 + i^gl r_1, min((i^gl+1) r_1, N_1)
    do j^gl = 0, N_2–1
    do j = 1 + j^gl 1, min((j^gl+1)1, N_2)
```

$S(i, j)$
enddo
enddo
enddo
enddo

После перестановки циклов (если это преобразование алгоритма корректно) получим

do $i^{gl} = 0, Q_1 - 1$
do $j^{gl} = 0, N_2 - 1$
do $i = 1 + i^{gl} r_1, \min((i^{gl}+1)\, r_1, N_1)$
$j = 1 + j^{gl}$
$S(i, j)$
enddo
enddo
enddo

Можно поступить проще: считать, что цикл с параметром j не разбивается и является глобальным (тогда границы изменения цикла тайлинг оставляет прежними):

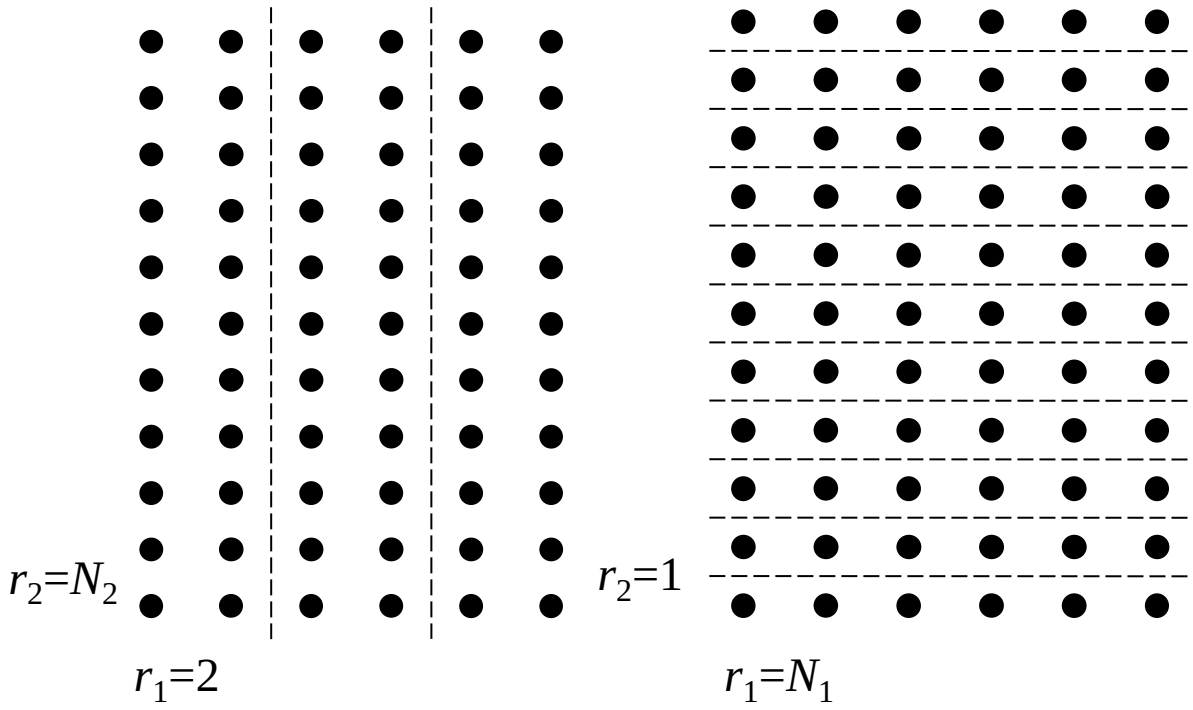
do $i^{gl} = 0, Q_1 - 1$
do $j^{gl} = 1, N_2$
Tile(i^{gl}, j^{gl})
enddo
enddo

Вычисления тайла Tile(i^{gl}, j^{gl}) задаются циклом

do $i = 1 + i^{gl} r_1, \min((i^{gl}+1) r_1, N_1)$
$j = j^{gl}$
$S(i, j)$
enddo

Пусть теперь $r_2 = N_2$ (рисунок ниже). Тогда (напомним, $Q_2 = \lceil N_2 / r_2 \rceil$)

do $i^{gl} = 0, Q_1 - 1$
do $i = 1 + i^{gl} r_1, \min((i^{gl}+1)\, r_1, N_1)$
do $j^{gl} = 0, 0$
do $j = 1 + j^{gl} N_2, \min((j^{gl}+1) N_2, N_2)$
$S(i, j)$
enddo
enddo
enddo
enddo



После упрощения (тайлинг не изменил порядок выполнения операций) имеем

```
do i^gl = 0, Q_1 –1
   do i = 1 + i^gl r_1, min((i^gl+1) r_1, N_1)
      do j = 1, N_2
         S(i, j)
      enddo
   enddo
enddo
```

Получили, что цикл с параметром j не разбивается и является локальным:

```
do i^gl = 0, Q_1 –1
   Tile(i^gl,0)
enddo
```

где вычисления тайла Tile(i^{gl},0) есть

```
do i = 1 + i^gl r_1, min((i^gl+1) r_1, N_1)
   do j = 1, N_2
      S(i,j)
   enddo
enddo
```

Таким образом, если по какой-либо координате размер тайла равен

единице, то соответствующий цикл можно не разбивать и считать глобальным, а если по какой-либо координате размер тайла равен числу значений параметра цикла, то соответствующий цикл не разбивается и является локальным.

Заметим, что в случае $r_1=N_1$, $r_2=1$ тайлинг фактически означает перестановку циклов исходного алгоритма:

do j^{gl} = 1, N_2
  Tile(0,j^{gl})
enddo

Вычисления одного тайла Tile(0,j^{gl}) есть

do i = 1, N_1
  $S(i, j^{gl})$
enddo

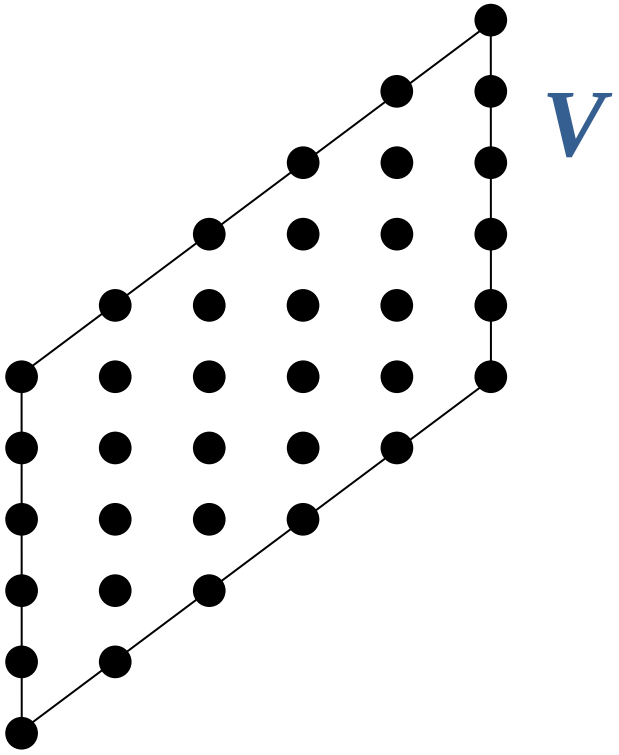
Пример 3 ([1], метод окаймления). Рассмотрим двумерный цикл, у которого итерационная область не является прямоугольным параллелепипедом:

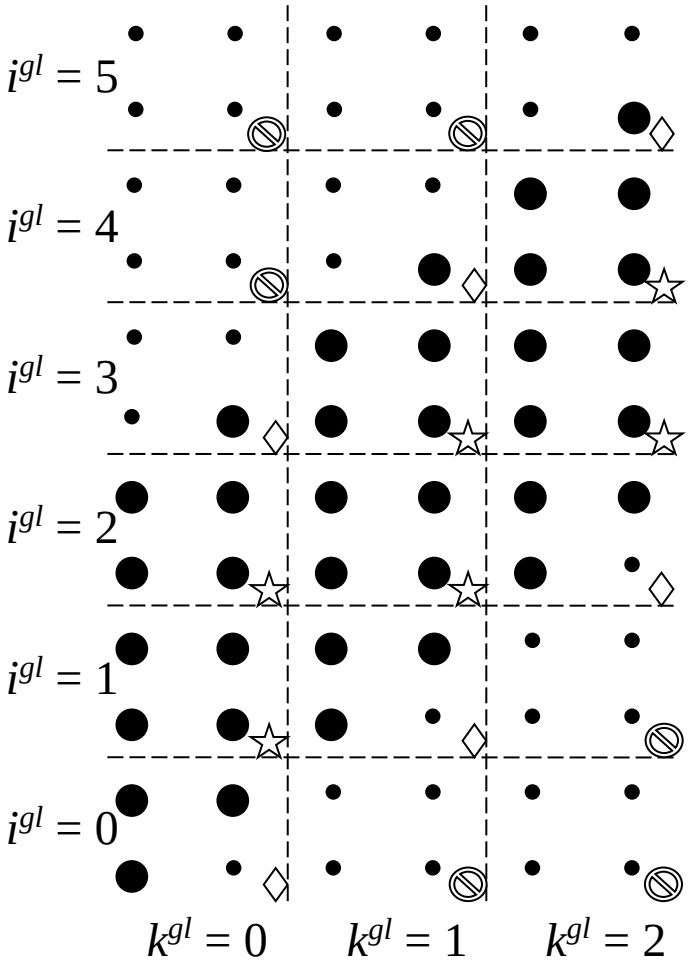
do k = 1, N_1
  do i = k+1, k+ N_2
    $S(k, i)$
  enddo
enddo

После применения тайлинга (пояснения для случая $N_1=N_2=6$, $r_1=r_2=2$ приведены на рисунке, $Q_1=\left\lceil \frac{N_1}{r_1} \right\rceil$, $Q_2=\left\lceil \frac{N_1+N_2-1}{r_2} \right\rceil$) получим

do k^{gl} = 0, Q_1–1
  do i^{gl} = 0, Q_2–1
    do k = 1 + $k^{gl}r_1$, min((k^{gl}+1)r_1, N_1)
      do i = max(1 + $i^{gl}r_2$, k+1), min((i^{gl}+1)r_2, k+ N_2)
        $S(k, i)$
      enddo
    enddo
  enddo
enddo

На рисунке видно, что часть тайлов являются пустыми или неполными; представленный подход, при котором допускаются избыточные области изменения параметров глобальных циклов, называется методом окаймления.





Виды тайлов: ☆ – полный ◇ – неполный ⊘ – пустой

Пример 4 (задание зерна вычислений при организации зернистых параллельных вычислительных процессов). Рассмотрим основную часть алгоритма решения двумерной задачи Дирихле для уравнения Пуассона методом верхней релаксации:

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

Будем считать цикл с параметром m глобальным не разбиваемым, а цикл с параметром j локальным не разбиваемым. Произведем разбиение цикла уровня вложенности 2: заменим цикл по 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 $j = 1, N_y - 1$
      $u(i,j)=F(u(i-1,j),u(i,j-1),u(i,j),u(i,j+1),u(i+1,j))$
    enddo
  enddo
  enddo
enddo

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 ($\mathrm{Pr}_{i^{gl}}$ – процесс с номером i^{gl}):

Для каждого процесса $\mathrm{Pr}_{i^{gl}}$, $0 \le i^{gl} \le Q_2-1$:
do $m = 1, M$
  Tile(m,i^{gl},0)
  обмен данными (синхронизация)
enddo

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
        u(i,j) = F(u(i–1,j), u(i,j–1), u(i,j), u(i,j+1), u(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 тайлу (кроме, возможно, граничных тайлов), равно $r_2 \times (N_y - 1)$.

Пример 5 (улучшение локальности при последовательных вычислениях). Рассмотрим один из вариантов алгоритма Гаусса-Жордана решения систем N линейных алгебраических уравнений с N неизвестными и P правыми частями:

```
do  k=1,N
    l(1)=1/a(1,k)
    do  p=2,N
        l(p)= –a(p,k)
    enddo
    do  j=k+1,N+P
        u(j)=a(1,j)l(1)
        do  i=2,N
            a(i–1,j)=a(i,j)+l(i)u(j)
        enddo
        a(N,j)=u(j)
    enddo
enddo
```

Если массивы хранятся в памяти строками, то при изменении параметра цикла i данные $a(i,j)$ извлекаются с большим шагом (при достаточно большом N). Тайлинг позволит использовать данные $a(i,j)$ одного блока кэша, что уменьшит время обращения к памяти.

Заменим цикл по j двумерной циклической конструкцией:

```
do  k=1,N
    l(1)=1/a(1,k)
    do  p = 2,N
        l(p)= –a(p,k)
    enddo
    do  j^gl = 0, Q_2 –1
```

```
      do  j = max(2 + j^gl r_2 , k +1), min(1+( j^gl + 1)r_2, N + P)
         u(j)=a(1,j)l(1)
         do  i = 2,N
            a(i–1,j)=a(i,j)+l(i)u(j)
         enddo
         a(N,j)=u(j)
      enddo(j)
      enddo( j^gl )
   enddo(k)
```

Здесь r_2 характеризует число значений параметров второго цикла двумерной конструкции, $Q_2=\lceil (N+P-1)/r_2 \rceil$. Распределим теперь цикл с параметром j между выполняемыми операторами и переставим его с циклом по i:

```
   do  k=1,N
      l(1)=1/a(1,k)
      do  p = 2,N
         l(p)= –a(p,k)
      enddo
      do  j^gl = 0, Q_2 –1
         do  j = max(2 + j^gl r_2 , k +1), min(1+( j^gl + 1)r_2, N + P)
            u(j)=a(1,j)l(1)
         enddo
         do  i = 2, N
            do  j = max(2+ j^gl r_2 , k+1), min(1+( j^gl +1)r_2, N + P)
               a(i–1,j)=a(i,j)+l(i)u(j)
            enddo
         enddo
         do  j = max(2 + j^gl r_2 , k +1), min(1+( j^gl + 1)r_2, N + P)
            a(N,j)=u(j)
         enddo
      enddo( j^gl )
   enddo(k)
```

Теперь при небольших r_2 параметры j, i каждого из локальных циклов (i – параметр не разбиваемого локального цикла) изменяются через небольшое число итераций алгоритма, что позволяет эффективно использовать кэш при хранении элементов массива как строками, так и столбцами.

Пример 6 (ускорение вычислений на многоядерном процессоре). Основная часть одного из вариантов метода Якоби решения разностной задачи Дирихле (одномерное уравнение) можно задать следующим гнездом циклов:

```
do m = 1, M
   do i = 1, N_x–1
      u(i)=(l(i–1)+l(i)+l(i+1))/3
   enddo
   do i = 1, N_x–1
      l(i)=u(i)
   enddo
enddo
```

Время выполнения параллельного OpenMP-алгоритма на восьмиядерном процессоре с использованием разбиения на тайлы (при этом требуется предварительное аффинное преобразование циклов) и без использования разбиения отличается почти в 10 раз [2].

Пример 7 (двухуровневый тайлинг для распараллеливания алгоритмов при программирования общего назначения на графических процессорах [3–5]). Для реализации алгоритма на графическом процессоре множество операций алгоритма должно быть разбито на блоки, а блоки – на потоки (нити) вычислений.

Рассмотрим основную часть алгоритма перемножения матрицы A размера $N_1 \times N_3$ и матрицы B размера $N_3 \times N_2$:

```
do i = 1, N_1
   do j = 1, N_2
      do k = 1, N_3
         c(i,j) = c(i,j) + a(i,k) b(k,j)
      enddo
   enddo
enddo
```

После разбиения всех трех циклов (r_1, r_2, r_3 – параметры, $Q_1 = \left\lceil \frac{N_1}{r_1} \right\rceil$, $Q_2 = \left\lceil \frac{N_2}{r_2} \right\rceil$, $Q_3 = \left\lceil \frac{N_3}{r_3} \right\rceil$) и перестановки циклов получим:

```
do i^gl = 0, Q_1 –1
   do j^gl = 0, Q_2–1
      do k^gl = 0, Q_3–1
         do i = 1+ i^gl r_1, min((i^gl+1)r_1, N_1)
```

```
                do j = 1+ j^gl r_2, min((j^gl+1)r_2, N_2)
                    do k = 1+ k^gl r_3, min((k^gl+1)r_3, N_3)
                        c(i,j) = c(i,j) + a(i,k) b(k,j)
                    enddo
                enddo
            enddo
        enddo
    enddo
```

Положим, что один блок вычислений Bl(i^{gl},j^{gl},k^{gl}) составляют вычисления

```
do i = 1+ i^gl r_1, min((i^gl+1)r_1, N_1)
    do j = 1+ j^gl r_2, min((j^gl+1)r_2, N_2)
        do k = 1+ k^gl r_3, min((k^gl+1)r_3, N_3)
            c(i,j) = c(i,j) + a(i,k) b(k,j)
        enddo
    enddo
enddo
```

Для организации потоков произведем разбиение операций блоков вычислений Bl(i^{gl},j^{gl},k^{gl}) (тайлинг уровня 2).

Обозначим через $Q_{1,2}$, $Q_{2,2}$ число итераций в новых глобальных циклах с параметрами i^{gl2}, j^{gl2} соответственно, а через $r_{1,2}$, $r_{2,2}$ число значений параметров i, j в локальных циклах второго уровня ($r_{1,2}=\left\lceil \frac{r_1}{Q_{1,2}} \right\rceil$, $r_{2,2}=\left\lceil \frac{r_2}{Q_{2,2}} \right\rceil$); цикл с параметром k зададим локальным не разбиваемым. Вычисления одного блока после разбиения можно записать в виде

```
do i^gl2= 0, Q_1,2 –1
    do j^gl2= 0, Q_2,2–1
        do i=1+i^gl r_1+i^gl2 r_1,2, min(i^gl r_1+(i^gl2+1)r_1,2, (i^gl+1)r_1, N_1)
            do j=1+j^gl r_2+j^gl2 r_2,2, min(j^gl r_2+(j^gl2+1)r_2,2, (j^gl+1)r_2, N_2)
                do k = 1+ k^gl r_3, min((k^gl+1)r_3, N_3)
                    c(i,j) = c(i,j) + a(i,k) b(k,j)
                enddo
            enddo
        enddo
    enddo
enddo
```

Будем считать, что параметры $Q_{1,2}$, $Q_{2,2}$ задают число потоков ($Q_{1,2}\times Q_{2,2}$) в блоке. Один поток вычислений Thr(i^{gl2},j^{gl2}) блока Bl(i^{gl},j^{gl},k^{gl}) составляют вычисления

do i=1+$i^{gl}r_1$+$i^{gl2}r_{1,2}$, min($i^{gl}r_1$+(i^{gl2}+1)$r_{1,2}$, (i^{gl}+1)r_1, N_1)
  do j=1+$j^{gl}r_2$+$j^{gl2}r_{2,2}$, min($j^{gl}r_2$+(j^{gl2}+1)$r_{2,2}$, (j^{gl}+1)r_2, N_2)
    do k = 1+ $k^{gl}r_3$, min((k^{gl}+1)r_3, N_3)
      $c(i,j) = c(i,j) + a(i,k)\ b(k,j)$
    enddo
  enddo
enddo

Таким образом, в результате двухуровневого тайлинга получили алгоритм, множество операций которого разбито на блоки и потоки (нити) вычислений. Блоки вычислений могут выполняться независимо для каждого k^{gl}:

do k^{gl} = 0, Q_3–1
  dopar j^{gl} = 0, Q_2–1
    dopar i^{gl} = 0, Q_1–1
      Bl(i^{gl},j^{gl},k^{gl})·
    enddo
  enddopar
enddopar

Вычисления одного блока Bl(i^{gl},j^{gl},k^{gl}) есть параллельно выполняемые потоки:

dopar i^{gl2}= 0, $Q_{1,2}$–1
  dopar j^{gl2}= 0, $Q_{2,2}$–1
    Thr(i^{gl2},j^{gl2})
  enddopar
enddopar

Формальная техника тайлинга [6–8]. Пусть в гнезде циклов имеется Θ наборов выполняемых операторов таких, что в окружении каждого набора есть хотя бы один разбиваемый цикл. Под набором операторов будем понимать один или несколько операторов, окруженных одним и тем же множеством циклов. Наборы операторов линейно упорядочены расположением их в записи алгоритма; операторы тоже линейно упорядочены расположением в записи алгоритма. Обозначим: V^θ, $1\le\theta\le\Theta$, – области изменения параметров циклов, окружающих наборы операторов, n^θ – размерность области V^θ, число циклов, окружающих θ-й

набор операторов.

Следующие величины и множества используются для формализации тайлинга.

$m_\zeta^\vartheta = \min\, j_\zeta$, $M_\zeta^\vartheta = \max\, j_\zeta$ – предельные значения изменения параметра цикла уровня вложенности ζ для ϑ-го набора операторов; m_ζ^ϑ и M_ζ^ϑ являются константами, которые могут зависеть от внешних переменных; если два набора операторов имеют общий цикл с параметром j_ζ, то $m_\zeta^{\vartheta_1} = m_\zeta^{\vartheta_2}$, $M_\zeta^{\vartheta_1} = M_\zeta^{\vartheta_2}$;

$r_1^\vartheta, ..., r_{n^\vartheta}^\vartheta$ – заданные натуральные числа, определяющие размеры тайла; r_ζ^ϑ обозначает число значений параметра j_ζ, приходящихся на один тайл θ-го набора операторов; r_ζ^ϑ может принимать фиксированное значение в пределах от 1 до $r_\zeta^{\vartheta,\max}$ включительно, где $r_\zeta^{\vartheta,\max} = M_\zeta^\vartheta - m_\zeta^\vartheta + 1$; если $r_\zeta^\vartheta = 1$, то цикл с параметром j_ζ является глобальным не разбиваемым; если $r_\zeta^\vartheta = r_\zeta^{\vartheta,\max}$, то цикл с параметром j_ζ является локальным не разбиваемым; если два набора операторов имеют общий цикл с параметром j_ζ, то $r_\zeta^{\vartheta_1} = r_\zeta^{\vartheta_2}$;

$Q_\zeta^\vartheta = \left\lceil (M_\zeta^\vartheta - m_\zeta^\vartheta + 1)/r_\zeta^\vartheta \right\rceil$, $1 \le \zeta \le n^\vartheta$, – число частей, на которые при формировании тайлов разбивается область значений параметра j_ζ цикла, окружающего ϑ-й набор операторов;

$V^{\vartheta,gl} = \left\{ J^{gl}(j_1^{gl}, ..., j_{n^\vartheta}^{gl}) \middle|\ 0 \le j_\zeta^{gl} \le Q_\zeta^\vartheta - 1,\ 1 \le \zeta \le n^\vartheta \right\}$ – области изменения параметров глобальных, т.е. уровня тайлов, циклов;

$V_{J^{gl}}^\vartheta = \left\{ J(j_1, ..., j_{n^\vartheta}) \in V^\vartheta \middle|\ m_\zeta^\vartheta + j_\zeta^{gl} r_\zeta^\vartheta \le j_\zeta \le m_\zeta^\vartheta - 1 + (j_\zeta^{gl} + 1) r_\zeta^\vartheta,\ 1 \le \zeta \le n^\vartheta \right\}$,

$J^{gl}(j_1^{gl}, ..., j_{n^\vartheta}^{gl}) \in V^{\vartheta,gl}$, – области изменения параметров локальных (уровня операций тайлов) циклов при фиксированных значениях параметров глобальных циклов. Если цикл является глобальными не разбиваемыми, то условимся, что границы изменения этого цикла тайлинг оставляет прежними.

Если $J(j_1, ..., j_{n^\vartheta}) \in V_{J^{gl}}^\vartheta$, то имеют место равенства

$$j_\zeta^{gl} = \left\lfloor \frac{j_\zeta - m_\zeta^\vartheta}{r_\zeta^\vartheta} \right\rfloor, \tag{2}$$

$$j_\zeta^{gl} = \left\lceil \frac{j_\zeta - m_\zeta^\vartheta + 1}{r_\zeta^\vartheta} \right\rceil - 1, \tag{3}$$

где используются обозначения ближайшего «снизу» целого числа и

ближайшего «сверху» целого числа. Действительно, так как $m_\zeta^\vartheta + j_\zeta^{gl} r_\zeta^\vartheta \le j_\zeta \le m_\zeta^\vartheta - 1 + (j_\zeta^{gl}+1) r_\zeta^\vartheta$, то из системы двух неравенств $j_\zeta - m_\zeta^\vartheta + 1 \le (j_\zeta^{gl}+1) r_\zeta^\vartheta$ и $j_\zeta^{gl} r_\zeta^\vartheta \le j_\zeta - m_\zeta^\vartheta$ получим $\frac{j_\zeta - m_\zeta^\vartheta + 1}{r_\zeta^\vartheta} - 1 \le j_\zeta^{gl} \le \frac{j_\zeta - m_\zeta^\vartheta}{r_\zeta^\vartheta}$. Так как j_ζ^{gl} является целым числом, то $j_\zeta^{gl} = \left\lceil \frac{j_\zeta - m_\zeta^\vartheta + 1}{r_\zeta^\vartheta} \right\rceil - 1 = \left\lfloor \frac{j_\zeta - m_\zeta^\vartheta}{r_\zeta^\vartheta} \right\rfloor$.

Множество операций, выполняемых на итерациях множества $V_{J^{gl}}^\vartheta$, будем также обозначать $V_{J^{gl}}^\vartheta$. Множества $V_{J^{gl}}^\vartheta$ называются тайлами $V_{J^{gl}}^\vartheta$ или тайлами J^{gl}. Тайлы $V_{J^{gl}}^\vartheta$ будем называть тайлами ϑ-го типа.

Опишем структуру многомерного цикла после применения к нему преобразования тайлинга. Параметры циклов j_ζ^{gl} изменяются в соответствии с неравенствами $0 \le j_\zeta^{gl} \le Q_\zeta^\vartheta - 1$, для каждого набора операторов имеется столько локальных циклов, сколько $r_1^\vartheta, ..., r_{n^\vartheta}^\vartheta$ превосходят единицу. Соответственно этому описанию генерируется код.

Отметим, что некоторые множества $V_{J^{gl}}^\vartheta$ могут быть пустыми, так как применяется метод окаймления.

Допустимость тайлинга. Установим необходимые и достаточные условия, при выполнении которых преобразование тайлинга является корректным.

Пусть имеется некоторая зависимость $S_\alpha(I) \to S_\beta(J)$. Обозначим через ϑ^α и ϑ^β номера наборов операторов, которым принадлежат S_α и S_β соответственно, а через $c_{\alpha,\beta}$ множество общих циклов в окружении операторов S_α и S_β. Отметим очевидное: если некоторый цикл является общим для операторов, то общим является и любой более внешний, в том числе и самый внешний, цикл. Если общих циклов не существует, то положим $c_{\alpha,\beta}=0$.

Теорема 1. *Пусть существует хотя бы одна такая зависимость* $S_\alpha(I) \to S_\beta(J)$, $I \in V_{I^{gl}}^{\vartheta^\alpha}$, $J \in V_{J^{gl}}^{\vartheta^\beta}$, *что* $c_{\alpha,\beta} \neq 0$. *Тайлинг является допустимым тогда и только тогда, когда для любой такой зависимости выполняются следующие условия:*

$$(j_1^{gl}, ..., j_{c_{\alpha\beta}}^{gl}) \ge_{lex} (i_1^{gl}, ..., i_{c_{\alpha\beta}}^{gl}), \tag{4}$$

$$if \ \ (j_1^{gl}, ..., j_{c_{\alpha,\beta}}^{gl}) = (i_1^{gl}, ..., i_{c_{\alpha,\beta}}^{gl}) \ \ then \ \ \vartheta^\beta \ge \vartheta^\alpha. \tag{5}$$

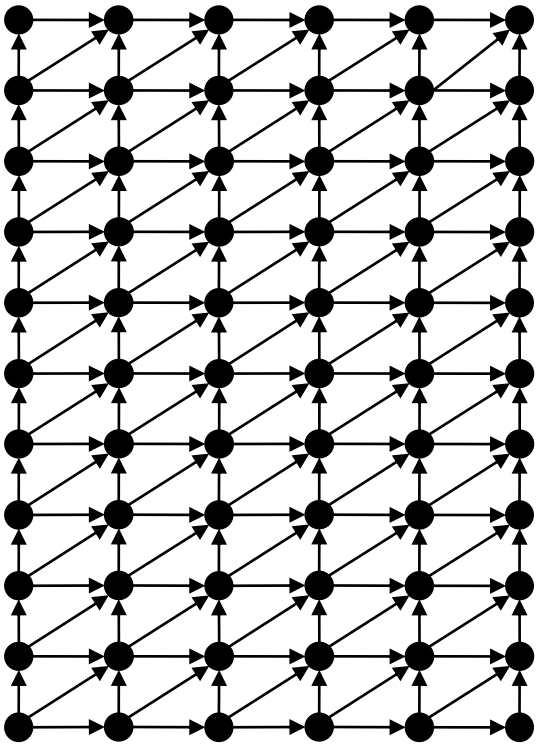
При отсутствии указанных зависимостей тайлинг допустим.

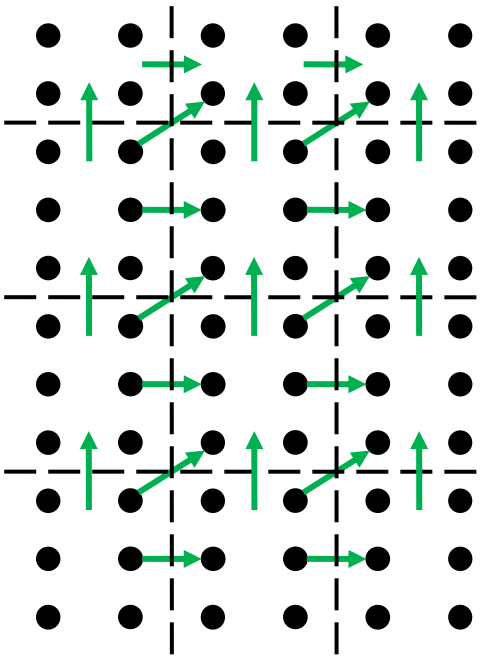
Действительно, при тайлинге может меняться порядок выполнения операций, зависящих от параметров общих циклов. Рассматриваемый

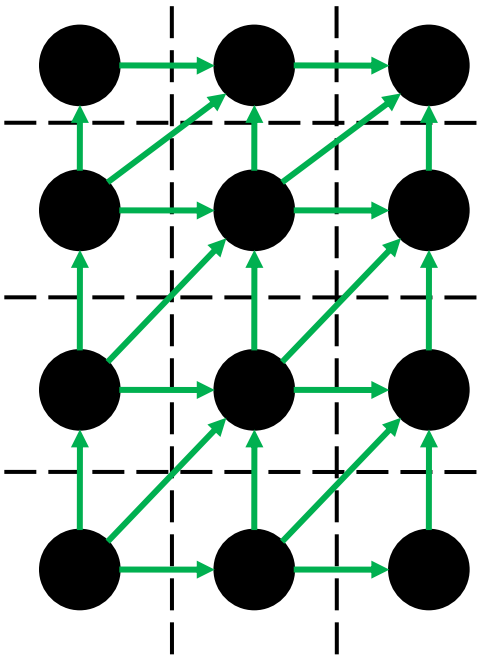
тайлинг не изменяет порядок выполнения операций в пределах любого фиксированного тайла. Поэтому необходимо и достаточно установить корректность тайлинга на глобальном уровне (уровне тайлов). Условие (4) устанавливает сохранение зависимостей между итерациями общих глобальных циклов. Условие (5) является необходимым и достаточным для сохранения зависимостей в случае совпадения параметров общих глобальных циклов. В случае $(j_1^{gl},...,j_{c_{\alpha\beta}}^{gl}) = (i_1^{gl},...,i_{c_{\alpha\beta}}^{gl})$, $\vartheta^{\beta} = \vartheta^{\alpha}$ зависимые операции принадлежат одному тайлу, поэтому тайлинг не нарушает зависимостей.

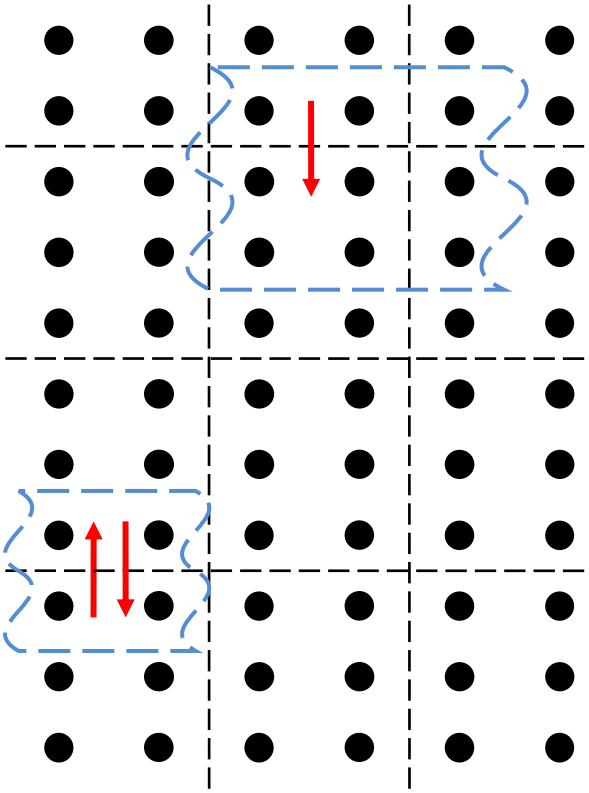
Заметим, что условие (4) может выполняться при одних размерах тайлов, но не выполняться при других размерах.

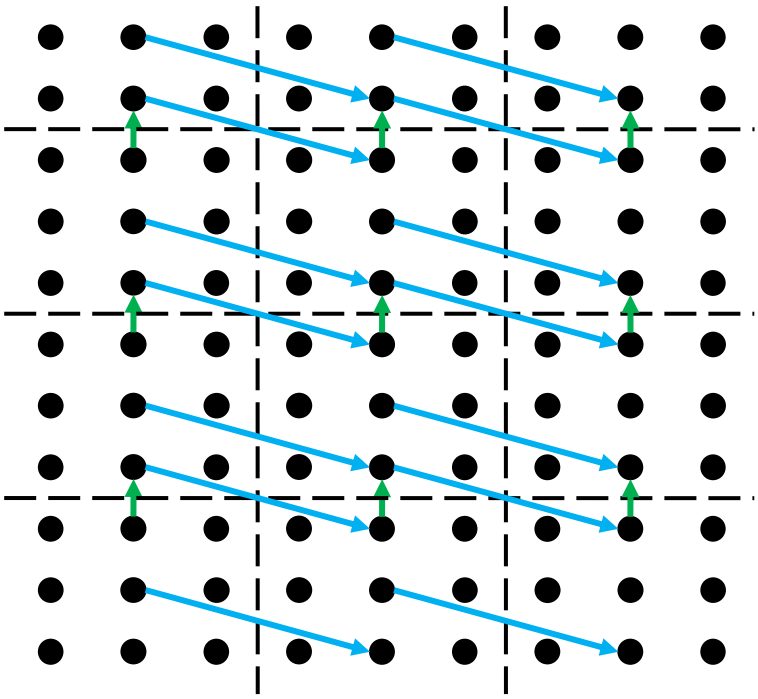
Ниже приведены некоторые иллюстрации, демонстрирующие необходимость и достаточность условий (4) и (5). Отметим, что один вектор зависимостей на уровне операций алгоритма может порождать несколько глобальных векторов зависимостей на уровне тайлов.

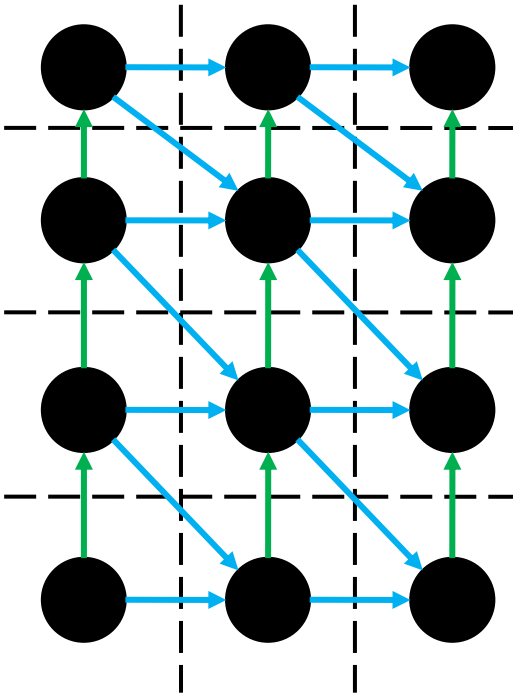




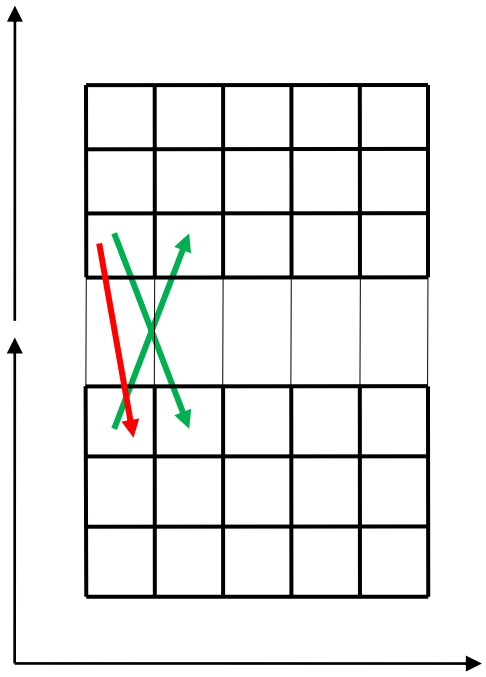








```
do i = 1, N
    do j = 1, N
        S1(i, j)
    enddo
    do j = 1, N
        S2(i, j)
    enddo
enddo
```



Рассмотрим непосредственно следующий из теоремы 1 способ проверки корректности тайлинга.

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

- определить общие циклы;
- если $c_{\alpha,\beta}=0$, то зависимость не нарушает корректности тайлинга (ввиду $\vartheta^\beta \geq \vartheta^\alpha$);
- если $c_{\alpha,\beta}\neq 0$, то определить первые $c_{\alpha,\beta}$ глобальные координаты итераций I и J ($I \in V_{I^{gl}}^{\vartheta^\alpha,loc}$, $J \in V_{J^{gl}}^{\vartheta^\beta,loc}$);
- если $c_{\alpha,\beta}\neq 0$, то проверить справедливость условия $(j_1^{gl},...,j_{c_{\alpha\beta}}^{gl}) \geq_{lex} (i_1^{gl},...,i_{c_{\alpha\beta}}^{gl})$;
- если $c_{\alpha,\beta}\neq 0$ и $(j_1^{gl},...,j_{c_{\alpha,\beta}}^{gl}) = (i_1^{gl},...,i_{c_{\alpha,\beta}}^{gl})$, то проверить справедливость условия $\beta \geq \alpha$ (или, что равносильно, проверить справедливость $\vartheta^\beta \geq \vartheta^\alpha$).

[При определении глобальных координат: если пользоваться имеющейся на данный момент программой, то считать, что 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.]

Приведем известные удобные для практического использования достаточные условия допустимости тайлинга. Выполнение этих условий гарантирует корректность тайлинга для тайлов любого размера.

Теорема 2. *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* $c_{\alpha,\beta}\neq 0$. *Тайлинг является допустимым, если для любой такой зависимости выполняются следующие условия:*

$$j_\xi \geq i_\xi,\ 2 \leq \zeta \leq c_{\alpha,\beta}, \tag{6}$$

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

При отсутствии указанных зависимостей тайлинг допустим.

Доказательство. Покажем, что справедливость условий теоремы 2 гарантирует справедливость условий теоремы 1.

Отметим сразу, что для $\xi=1$ условие (6) $j_\xi \geq i_\xi$ выполняется всегда (следует из определения зависимости операций).

Если выполняются условия (6), то из равенств (2) (или (3)) следует справедливость условия (4) $j_\xi^{gl} \geq i_\xi^{gl}$ (для глобальных не разбиваемых циклов $j_\xi^{gl} = j_\xi$, $i_\xi^{gl} = i_\xi$, для локальных не разбиваемых циклов $j_\xi^{gl}=1$, $i_\xi^{gl}=1$). Если выполняется условия (7) $\beta \geq \alpha$, то выполняется условия (5) $\vartheta^\beta \geq \vartheta^\alpha$.

Таким образом, условия (6) и (7) являются достаточными для выполнения условий (4) и (5). □

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 переставляемости указанных циклов, а условия (7) достаточны для распределения циклов.

[Привести примеры (см. рисунки выше), когда условия (6) или (7) не выполняются, но тайлинг корректен.]

[В лекции мы понимаем тайлинг как строго определенное преобразование циклов. Можно рассматривать тайлинг в более широком смысле (см. задачи тайлинга в р. 3.5 Курса лекций).]

[Генерация кода: проблема получения параметрического тайлинга (в PLUTO не параметрический тайлинг, в CLooG - параметрический); GenTile, PrimeTile, DynTile]

Влияние избыточных вычислений на время реализации зернистых версий алгоритма Гаусса-Жордана. Еще раз рассмотрим зернистый алгоритм, полученный с помощью разбиения цикла j:

```
do k=1,N
   l(1)=1/a(1,k)
   do p = 2,N
      l(p)= –a(p,k)
   enddo
   do j^gl = 0, Q_2 –1
      do j = max(2 + j^gl r_2 , k +1) , min(1+( j^gl + 1)r_2, N + P)
         u(j)=a(1,j)l(1)
      enddo
      do i = 2, N
         do j = max(2 + j^gl r_2 , k +1) , min(1+( j^gl + 1)r_2, N + P)
            a(i–1,j)=a(i,j)+l(i)u(j)
         enddo
      enddo
      do j = max(2 + j^gl r_2 , k +1) , min(1+( j^gl + 1)r_2, N + P)
         a(N,j)=u(j)
      enddo
   enddo(j^gl)
enddo(k)
```

Метод окаймления в данном примере порождает так называемые пустые тайлы: нижняя граница изменения цикла с параметром j при некоторых j^{gl} превосходит верхнюю границу. Можно избавиться от пустых тайлов: вместо оператора

do j^{gl} = 0, Q_2–1

где $Q_2 = \lceil (N+P-1)/r_2 \rceil$, записать оператор

do j^{gl} = $\lceil k/r_2 \rceil$–1, $\lceil (N+P-1)/r_2 \rceil$–1.

Назовем полученный алгоритм вариантом 1 алгоритма Гаусса-Жордана без избыточных вычислений.

Вариант 2 алгоритма Гаусса-Жордана без избыточных вычислений можно получить, если вместо оператора

do $j^{gl} = 0, \lceil (N+P-1)/r_2 \rceil - 1$

и оператора

do $j = \max(2 + j^{gl} r_2 , k + 1) , \min(1+(j^{gl}+ 1)r_2, N + P)$

записать оператор

do $j^{gl} = 0, \lceil (N+P-k)/r_2 \rceil - 1$

и оператор

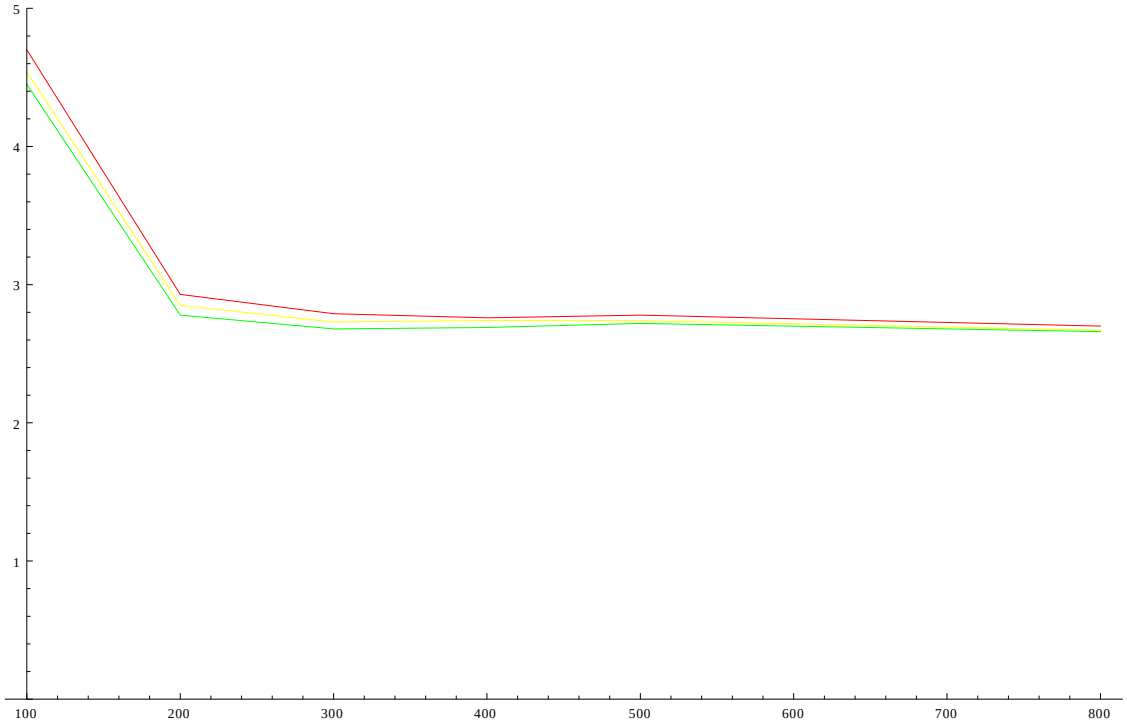
do $j = 1 + j^{gl} r_2 + k, \min((j^{gl}+ 1)r_2 + k, N + P)$

Вариант 2:

```
do k=1,N
   l(1)=1/a(1,k)
   do p = 2,N
      l(p)= –a(p,k)
   enddo
   do j^gl= 0, ⌈(N+P–k)/r2⌉–1
      do j = 1 + j^gl r2+k, min((j^gl+ 1)r2+k, N + P)
         u(j)=a(1,j)l(1)
      enddo
      do i = 2, N
         do j = 1 + j^gl r2+k, min((j^gl+ 1)r2+k, N + P)
            a(i–1,j)=a(i,j)+l(i)u(j)
         enddo
      enddo
      do j = 1 + j^gl r2+k, min((j^gl+ 1)r2+k, N + P)
         a(N,j)=u(j)
      enddo
   enddo(j^gl)
enddo(k)
```

Приведем результаты вычислительных экспериментов.

$N = 1000$



Red – реализация с избыточными вычислениями
Yellow – реализация без избыточных вычислений (вариант 1)
Green – реализация без избыточных вычислений (вариант 2)

$N = 2000$ (разбиваются циклы с параметрами j и i)

r2*r3	С избыточными	Без избыточных (1)	Без избыточных (2)
10*10	43.68	42.009	41.21
15*10	44.45	43.57	43.11
20*10	46.59	45.45	45.26
30*10	38.54	37.68	37.60
20*20	40.58	40.23	39.94
50*10	30.67	30.34	30.14
100*100	43.86	43.84	43.82

Пример 8 (допустимость тайлинга для алгоритма прямого хода метода Гаусса без выбора ведущего элемента). Пусть A – матрица порядка n, b – n-мерный вектор. Введем в рассмотрение расширенную матрицу системы (строка i расширенной матрицы есть i-я строка матрицы системы, дополненная элементом b_i вектора правых частей) и запишем алгоритм прямого хода метода Гаусса решения системы $Ax=b$:

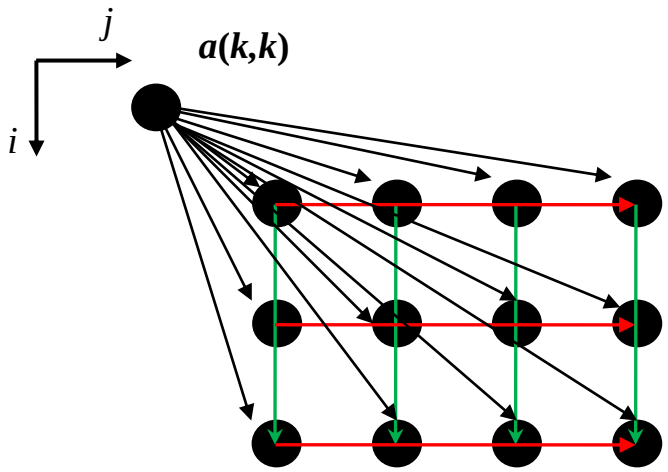
```
do  k= 1, n–1
    do  i= k+1, n
        do  j= k+1, n+1
```

$$S_1(k,i,j):\ a(i,j)=a(i,j) - \frac{a(i,k)}{a(k,k)}\ a(k,j) \qquad (8)$$

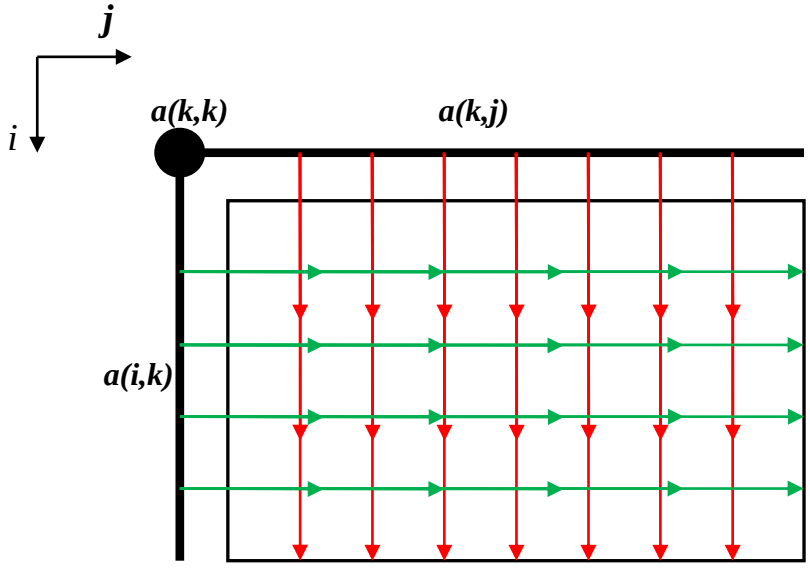
```
        enddo
    enddo
enddo
```

Гнездо циклов содержит один оператор и один массив данных (расширенную матрицу системы). В левой части оператора происходит определение элемента массива. На каждом вхождении в оператор массива a в правой части происходит использование ранее вычисленного элемента массива (при k=1 используются входные данные): $a(i,j)$ – использование прежнего значения обновляемого элемента, $a(i,k)$ – использование столбца, содержащего ведущий элемент, $a(k,j)$ – использование строки, содержащей ведущий элемент, $a(k,k)$ – использование ведущего элемента. Таким образом, вхождение массива a в левую часть оператора и вхождения в правую часть порождают зависимости. Если понимать детали алгоритма, то формализовать зависимости проще.

Изобразим схематично зависимости операций k-го шага, порождаемые ведущим элементом $a(k,k)$, вычисленном на (k–1)-м шаге:



Изобразим теперь схематично зависимости операций k-го шага, порождаемые строками и столбцами (вычисленными на (k–1)-м шаге), содержащими ведущий элемент:



Укажем итерации, порождающие зависимости.

$S_1(k–1,i,j)\to S_1(k,i,j)$: данное $a(i,j)$, вычисленное на итерации $(k–1,i,j)$, является аргументом $a(i,j)$ для вычислений на текущей итерации (k,i,j).

$S_1(k–1,i,k)\to S_1(k,i,j)$: $a(i,k)$, вычисленное на итерации $(k–1,i,k)$, является аргументом для вычислений на текущей итерации (k,i,j);

$S_1(k–1,k,j)\to S_1(k,i,j)$: $a(k,j)$, вычисленное на итерации $(k–1,k,j)$, является аргументом для вычислений на текущей итерации (k,i,j);

$S_1(k–1,k,k)\to S_1(k,i,j)$: ведущий элемент $a(k,k)$, вычисленный на итерации $(k–1,k,k)$, является аргументом для вычислений на текущей итерации (k,i,j).

Достаточные условия допустимости тайлинга выполняются: условия (6) выполняются, так как $j>k$, $i>k$; условия (7) выполняются, так как имеется только один оператор.

Выделим в алгоритме (8) преобразование строки, содержащей ведущий элемент:

```
do k= 1, n–1
   do j= k+1, n+1
      S0(k,j): u(k,j)= a(k, j) / a(k, k)
   enddo
   do i= k+1, n                                              (9)
      do j= k+1, n+1
         S1(k,i,j): a(i,j)=a(i,j) – a(i,k) u(k,j)
      enddo
   enddo
enddo
```

Зависимости $S_1(k–1,i,j)\to S_1(k,i,j)$ и $S_1(k–1,i,k)\to S_1(k,i,j)$ для алгоритма (9) такие же, как и для алгоритма (8). Появились новые зависимости:

$S_1(k–1,k,j)\to S_0(k,j)$: $a(k,j)$, вычисленное операцией $S_1(k–1,k,j)$, является аргументом для вычислений операции $S_0(k,,j)$;

$S_1(k–1,k,k)\to S_0(k,j)$: ведущий элемент $a(k,k)$, вычисленный операцией $S_1(k–1,k,k)$, является аргументом для вычислений операции $S_0(k,j)$.

$S_0(k,j)\to S_1(k,i,j)$: $u(k,j)$, вычисленное операцией $S_0(k,j)$, является аргументом для вычислений операции $S_1(k,i,j)$.

Рассмотрим допустимость тайлинга, для которого r_1 – параметр, $r_2=n–1$ (цикл i считается локальным не разбиваемым), r_3 – параметр.

Достаточные условия допустимости тайлинга выполняются для зависимостей $S_1(k–1,i,j)\to S_1(k,i,j)$ и $S_1(k–1,i,k)\to S_1(k,i,j)$ (показано ранее), а также для зависимости $S_0(k,j)\to S_1(k,i,j)$ (условия (6) выполняются, так как

операторы имеют только один общий цикл; условия (7) выполняются, так как $\alpha = 0$, $\beta = 1$).

Достаточные условия допустимости тайлинга не выполняются для зависимостей $S_1(k–1,k,j) \to S_0(k,j)$ и $S_1(k–1,k,k) \to S_0(k,j)$. Исследуем, выполняются ли для этих зависимостей необходимые и достаточные условия (4) и (5) теоремы 1.

Имеем: $c_{1,0}$=1 (один общий цикл в окружении операторов S_1 и S_0); I=(k–1,k,j) (для второй из рассматриваемых зависимостей I=(k–1,k,k)), J=(k,j); $i_1^{gl} = \left\lfloor \frac{k-1-m_1}{r_1} \right\rfloor$, где m_1=1, $j_1^{gl} = \left\lfloor \frac{k-m_1}{r_1} \right\rfloor$ (по формуле (2)). На r_1–1 итерации из каждых r_1 итераций k имеет место $j_1^{gl} = i_1^{gl}$. Если r_1>1, то условие (5) требует, чтобы оператор S_1 встречался в записи алгоритма раньше оператора S_0, что неверно. Таким образом, если r_1>1, тайлинг не допустим. Если r_1=1, то $j_1^{gl} > i_1^{gl}$, в условие (4) выполняется строгое неравенство, тайлинг допустим.

Таким образом, если r_2=n–1, r_3 – параметр, то преобразование алгоритма (9) посредством тайлинга возможно только в случае r_1=1. В то же время, математически эквивалентный алгоритму (9) алгоритм (8) допускает применение тайлинга с любыми r_1, r_2, r_3.

Заключение. Тайлинг – одно из самых часто используемых преобразований алгоритмов при выполнении как последовательных, так и параллельных вычислений. Применяется тайлинг для разбиения множества операций алгоритма на макрооперации, называемые зернами вычислений или тайлами. Зернистые вычисления позволяют уменьшить накладные расходы на использование иерархической памяти при последовательных вычислениях и на организацию обмена данными при параллельных вычислениях.

Мы ознакомились с основами теории тайлинга, с применением тайлинга для организации параллельных вычислительных процессов и потоков при программировании на компьютерах с распределенной памятью, на многоядерных процессорах, на графических процессорах.

Краткий вопрос для зачета: Использование тайлинга (перечислить, для чего можно использовать разбиение множества операций алгоритма на макрооперации).

Ответ: Разбиение множества операций алгоритма на макрооперации-тайлы можно использовать 1) для улучшения локальности как при последовательных, так и при параллельных вычислениях на компьютерах с общей памятью, 2) для задания зерна вычислений при организации параллельных зернистых вычислительных процессов на

компьютерах с распределённой памятью, 3) для задания потоков и блоков вычислений при организации параллельных вычислений на графических процессорах.

Список использованных источников

1. Renganarayanan L., Kim D., Rajopadhye S., Strout M.M. Parameterized tiled loops for free // ACM SIGPLAN Conference on Programming Language Design and Implementation. San Diego, California, USA, June 2007. P. 126–138.
2. Hartono A., Baskaran M., Ramanujam J., Sadayappan P. DynTile: Parametric tiled loop generation for parallel execution on multicore processors. // Proceedings of 24th International Parallel and Distributed Processing Symposium, Atlanta, April 2010. P.1–12.
3. Baskaran M., Bondhugula U, Krishnamoorthy S., Ramanujam J., Rountev A., Sadayappan, P. Automatic data movement and computation mapping for multi-level parallel architectures with explicitly managed memories // Proceedings of the ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming. February 2008.
4. Baskaran M., Ramanujam J., Sadayappan, P. Automatic C-to- CUDA code generation for affine programs // Proceedings of the Compiler Construction, 19th International Conference. Part of the Joint European Conferences on Theory and Practice of Software. Paphos, Cyprus, March 20–28, 2010.
5. Di P, Ye D., Su Yu, Sui Y., Xue J. Automatic parallelization of tiled loop nests with enhanced fine-grained parallelism on GPUs // 41st International Conference on Parallel Processing, ICPP 2012. Pittsburgh, PA, USA, September 10-13. IEEE Computer Society, 2012: P. 350-359.
6. Kim D., Rajopadhye S. Parameterized tiling for imperfectly nested loops. // Technical Report CS-09-101, Colorado State University, Department of Computer Science, February 2009. 21 P.
7. Tavarageri S., Hartono A., Baskaran M., Pouchet L.-N., Ramanujam J., Sadayappan P. Parametric tiling of affine loop nests // Proc. 15th Workshop on Compilers for Parallel Computers. Vienna, Austria, July 2010.
8. Лиходед Н.А. Обобщенный тайлинг // Доклады НАН Беларуси. 2011. Т. 55, № 1. С. 16-21.