Лекция 6. Зависимости во вложенных циклах и их анализ на параллельность

В современных научных исследованиях часто рассматриваются задачи, имеющие более одного измерения. При построении математических моделей таких задач приходится использовать многомерные массивы данных, а для их обработки в программах, реализующих построенные модели, применять вложенные циклы. Нам необходимо уметь применять к подобным программным конструкциям аппарат анализа зависимостей по данным для возможного распараллеливания последовательного кода.

Мы будем рассматривать нормализованные вложенные циклы (благо, нормализация их не сложна).

```
do j1 = 1, u1
    do j2 = 1,u2
        …
            do jn = 1,un
                ...
            enddo
        …
    enddo
enddo
```

В таких циклах конкретная итерация определяется совокупностью значений всех счетчиков $j_1, j_2, \ldots, j_n$. Будем рассматривать их набор как n-мерный вектор $\mathbf{J} = (j_1, j_2, \ldots, j_n)$ и назовем его *итерационным вектором*. Множество всех допустимых значений итерационных векторов образует *итерационное пространство* цикла. В этом пространстве между векторами можно ввести отношения порядка. Будем говорить, что $\mathbf{I} = \mathbf{J}$, если $\forall k, 1 \le k \le n, i_k = j_k$, и что $\mathbf{I} < \mathbf{J}$ в том случае, когда $\exists s, 1 \le s \le n$, такое что $\forall k, 1 \le k < s, i_k = j_k$, а $i_s < j_s$.

Как и в случае с одномерным циклом предположим, что тело цикла состоит из двух операторов S_1 и S_2, в наборы входных и/или выходных данных которых входит обращение к элементам одного и того же массива данных A с размерностью, совпадающей с количеством уровней вложенностей цикла. Пусть индексы массива могут принимать произвольные целочисленные значения. При этом для простоты допустим, что для оператора S_1 элемент массива A принадлежит к выходным переменным оператора, а для оператора S_2 — к входным переменным.

```
do j1 = 1, u1
    do j2 = 1,u2
        …
```

```
            do j_n = 1,u_n
                S_1:    A[f_1(J), …, f_n(J)] = …
                S_2:    … = …A[g_1(J), …, g_n(J)]…
            enddo
        …
    enddo
enddo
```

Здесь функции $f_k(\mathbf{J})$ и $g_k(\mathbf{J})$, $1 \le k \le n$ есть целочисленные функции от n целых переменных. Задачей является выяснение возможности разбиения итерационного пространства такого цикла на зоны ответственности для параллельного выполнения.

Из предыдущей лекции «ежу понятно» [6.1], что условия Бернстайна нарушаются, и, стало быть, зависимость возникает, если имеет решение система уравнений

$$\mathbf{F}(\mathbf{K}) = \mathbf{G}(\mathbf{\Lambda}),$$

где **F** — вектор-функция $(f_1, \ldots, f_n)$, а **G** — вектор-функция $(g_1, \ldots, g_n)$, при соблюдении условий

$$(0, \ldots, 0) \le \mathbf{K} \le (u_1, \ldots, u_n)$$

$$(0, \ldots, 0) \le \mathbf{\Lambda} \le (u_1, \ldots, u_n)$$

Понятно, что поиск решения системы диофантовых уравнений — задача несравненно более сложная, чем поиск решения одного диофантова уравнения. Если решения нет, то нет и зависимости операторов при выполнении цикла — каждая итерация может быть выполнена на своем исполнителе, максимальное количество которых — это $u_1 \times \ldots \times u_n$. Допустим, что с помощью некоторых колдовских приемов нам удалось определить, что решение существует, и найти соответствующие **K** и **Λ**. Введем для цикла понятие *вектора расстояний зависимости* (или просто вектора расстояний) следующим образом:

$$\mathbf{D} = \mathbf{\Lambda} - \mathbf{K}$$

(из вектора итераций, соответствующего итерации стока зависимости, вычитаем вектор итерации, соответствующий итерации источника зависимости).

Так, например, определим для двумерного цикла

```
do i = 1, u_1
    do j = 1,u_2
        S_1:    a[i, j] = b[i, j]*2
        S_2:    c[i,j] = a[i, j-1] + 1
    enddo
enddo
```

значение вектора расстояний. Для этого развернем наш цикл в итерационном пространстве:

$S_1^{(1,1)}$: a[1, 1] = b[1, 1]*2

$S_2^{(1,1)}$: c[1, 1] = a[1, 0] + 1

$S_1^{(1,2)}$: a[1, 2] = b[1, 2]*2

$S_2^{(1,2)}$: c[1, 2] = a[1, 1] + 1

…

$S_1^{(2,1)}$: a[2, 1] = b[2, 1]*2

$S_2^{(2,1)}$: c[2, 1] = a[2, 0] + 1

…

Легко видеть, что операторы $S_1^{(1,1)}$ и $S_2^{(1,2)}$ используют один и тот же элемент массива a[1, 1], при этом оператор $S_1^{(1,1)}$ является источником зависимости, а оператор $S_2^{(1,2)}$ является ее стоком. Поэтому $\mathbf{K} = (1, 1)$, $\mathbf{\Lambda} = (1, 2)$, и вектор расстояний есть $\mathbf{D} = (0, 1)$.

Определить тип существующей зависимости по данным (а в нашем случае — это истинная зависимость) и возможность распараллеливания цикла по виду вектора расстояний не так просто. Поэтому вводится понятие *вектора направлений* для цикла. Понятие, с одной стороны, — достаточно простое, но с другой стороны — несколько странное. Может быть, при его первом появлении в научной статье возникла опечатка, перекочевавшая в остальные научные работы. Я не знаю этого, ибо найти оригинальную статью, где вводился вектор направлений, мне не удалось. В чем заключается странность, поймем из определения. Компоненты вектора направлений **d** (а это — символьный вектор) определяются следующим образом:

$$d_i = \begin{cases} "=", \text{если } D_i = 0 \\ ">", \text{если } D_i < 0 \\ "<", \text{если } D_i > 0 \end{cases}$$

В нашем примере получаем **d** = (“=”, “<”).

Теперь на простых примерах мы разберем, как с помощью вектора направлений можно определять тип зависимости по данным между операторами, и сформулируем ряд общих утверждений, доказательство которых оставим для вашей самостоятельной работы. Все примеры будут для уровня вложенности циклов равных 2.

Начнем со случая вектора направлений **d** = (“=”, “=”).

Пример 6.1. Пусть дан цикл

do i = 1, u_1

do j = 1,u_2

S_1: a[i, j] = b[i, j]*2

S_2: c[i,j] = a[i, j] + 1

enddo

enddo

Вычислим вектор расстояний, вектор направлений, определим тип зависимости и возможность распараллеливания цикла.

После разворачивания цикла в итерационном пространстве получим:

$S_1^{(1,1)}$: a[1, 1] = b[1, 1]*2

$S_2^{(1,1)}$: c[1, 1] = a[1, 1] + 1

$S_1^{(1,2)}$: a[1, 2] = b[1, 2]*2

$S_2^{(1,2)}$: c[1, 2] = a[1, 2] + 1

…

$S_1^{(2,1)}$: a[2, 1] = b[2, 1]*2

$S_2^{(2,1)}$: c[2, 1] = a[2, 1] + 1

…

Вектор расстояний определяется элементарно: **D** = (0, 0). Строим вектор направлений **d** = (“=”, “=”). Легко видеть, что в данном случае мы имеем дело с зависимостью, не связанной с циклом (loop independent dependence) — все неприятности спрятаны в теле цикла в пределах одной итерации. Тип зависимости в данном примере — истинная зависимость. Распараллеливание возможно как по любой компоненте итерационного вектора (зоны ответственности нарезаются полосами либо по одному, либо по другому направлению), так и по двум компонентам одновременно (зоны ответственности — прямоугольники в итерационном пространстве). Если операторы в теле цикла поменять местами, то для нового примера, решающего другую задачу, изменится только тип зависимости, а все остальное останется справедливым.

Очень важно то, что в приведенном примере можно поменять местами внешний и внутренний цикл, и при этом результаты вычислений не изменятся.

Общее утверждение 6.1. Если многомерный цикл имеет вектор направлений **d** = (“=”, …, “=”), то цикл может быть распараллелен по произвольному количеству индексов безо всяких ограничений. При этом циклы, соответствующие различным уровням вложенности первоначальной конструкции, можно безопасно менять местами.

Формулировка и разбор примеров для многомерных циклов получаются достаточно громоздкими. Для их упрощения заметим, что в тело цикла для иллюстрации

достаточно включать всего лишь один оператор, для которого элементы одного и того же массива включены и в набор входных, и в набор выходных данных.

Пример 6.2. Рассмотрим цикл

```
do i = 1, u1
    do j = 1,u2
        S:    a[i, j] = a[i, j+1]*2
    enddo
enddo
```

Вычислим вектор расстояний, вектор направлений, определим тип зависимости и возможность распараллеливания цикла.

Раскроем выполнение цикла.

```
(1, 1)        a[1, 1[ = a[1, 2]*2
(1, 2)        a[1, 2] = a[1, 3]*2
…
(2, 1)        a[2, 1] = a[2 ,2]*2
(2, 2)        a[2, 2] = a [2, 3]*2
…
```

Очевидно, что вектор расстояний суть **D** = (0, -1), а вектор направлений **d** = (“=”, “>”). Значения элементов массива сначала используются, а потом заново вычисляются — это антизависимость. Если рассматривать внутренний цикл как один большой оператор, то все зависимости будут скрыты внутри этого большого оператора, и, значит, внешний цикл может быть распараллелен по итерациям без ограничений. Распараллеливание по внутреннему циклу и по двум циклам одновременно может быть осуществлено при условии размножения необходимых входных данных перед выполнением операций.

Отметим, что и в этом примере можно поменять местами внешний и внутренний цикл без изменения результатов вычислений. После такого преобразования вектор расстояний станет **D** = (-1, 0), а вектор направлений **d** = (“>”, “=”). Тип зависимости не меняется. Здесь без ограничений можно распараллеливать внутренний цикл, а для распараллеливания внешнего цикла и по двум направлениям нужно дублировать входные данные.

Пример 6.3. Возьмем программный фрагмент

```
do i = 1, u1
    do j = 1,u2
        S:    a[i, j] = a[i+1, j+1]*2
```

enddo

enddo

Вычислим вектор расстояний, вектор направлений, определим тип зависимости и возможность распараллеливания цикла.

Разворачиваем цикл.

(1, 1) a[1, 1[= a[2, 2]*2

(1, 2) a[1, 2] = a[2, 3]*2

…

(2, 1) a[2, 1] = a[3 ,2]*2

(2, 2) a[2, 2] = a [3, 3]*2

…

Очевидно, что вектор расстояний суть **D** = (-1, -1), а вектор направлений **d** = (“>”, “>”). Значения элементов массива сначала используются, а потом заново вычисляются — это антизависимость. Распараллеливание по внутреннему или по внешнему циклу, или по двум циклам одновременно может быть осуществлено при условии копирования необходимых входных данных перед выполнением операций.

И здесь можно поменять местами внешний и внутренний цикл без изменения результатов вычислений. Проведенный анализ также не изменится.

Общее утверждение 6.2. Пусть многомерный цикл имеет вектор направлений **d**, в состав которого входят только элементы “>” и “=”. Такой цикл может быть распараллелен безо всяких ограничений по любому количеству индексов, соответствующих компонентам “=” в векторе направлений. Распараллеливание по индексам, соответствующим компонентам “>” в векторе направлений, возможно при дублировании необходимых входных данных. Перед распараллеливанием циклы, соответствующие различным уровням вложенности первоначальной конструкции, можно безопасно менять местами.

Пример 6.4. Рассмотрим цикл

do i = 1, u_1

do j = 1,u_2

S: a[i, j] = a[i, j-1]*2

enddo

enddo

Вычислим вектор расстояний, вектор направлений, определим тип зависимости и возможность распараллеливания цикла.

Раскроем выполнение цикла.

(1, 1) a[1, 1[= a[1, 0]*2

(1, 2) a[1, 2] = a[1, 1]*2

…

(2, 1) a[2, 1] = a[2 ,0]*2

(2, 2) a[2, 2] = a [2, 1]*2

…

Очевидно, что вектор расстояний суть **D** = (0, 1), а вектор направлений **d** = (“=”, “<”). Значения элементов массива сначала вычисляются, а потом используются — это истинная зависимость. Если рассматривать внутренний цикл как один большой оператор, то все зависимости будут скрыты внутри этого большого оператора, и, значит, внешний цикл может быть распараллелен по итерациям без ограничений. Распараллеливание по внутреннему циклу и по двум циклам одновременно невозможно.

В таком последовательном примере тоже можно поменять местами внешний и внутренний цикл без изменения результатов вычислений. После такого преобразования вектор расстояний станет **D** = (1, 0), а вектор направлений **d** = (“<”, “=”). Тип зависимости не меняется. Здесь без проблем можно распараллеливать только внутренний цикл.

Пример 6.5. Рассмотрим цикл

```
do i = 1, u1
    do j = 1,u2
        S:    a[i, j] = a[i-1, j-1]*2
    enddo
enddo
```

Вычислим вектор расстояний, вектор направлений, определим тип зависимости и возможность распараллеливания цикла.

Раскроем выполнение цикла.

(1, 1) a[1, 1[= a[0, 0]*2

(1, 2) a[1, 2] = a[0, 1]*2

…

(2, 1) a[2, 1] = a[1 ,0]*2

(2, 2) a[2, 2] = a [1, 1]*2

…

Очевидно, что вектор расстояний суть **D** = (1, 1), а вектор направлений **d** = (“<”, “<”). Значения элементов массива сначала вычисляются, а потом используются — это истинная зависимость. Любое распараллеливание невозможно. И здесь в

последовательном примере тоже можно поменять местами внешний и внутренний цикл без изменения результатов вычислений.

Общее утверждение 6.3. Пусть многомерный цикл имеет вектор направлений **d**, в состав которого входят только элементы “<” и “=”. Такой цикл может быть распараллелен безо всяких ограничений по любому количеству индексов, соответствующих компонентам “=” в векторе направлений. Распараллеливание по индексам, соответствующим компонентам “<” в векторе направлений, проблематично (см. предыдущую лекцию для случая положительного расстояния зависимости). Перед распараллеливанием циклы, соответствующие различным уровням вложенности первоначальной конструкции, можно безопасно менять местами.

Пример 6.6. Берем фрагмент кода

```
do i = 1, u1
        do j = 1,u2
                S:      a[i, j] = a[i+1, j-1]*2
        enddo
enddo
```

Вычислим вектор расстояний, вектор направлений, определим тип зависимости и возможность распараллеливания цикла.

Раскроем выполнение цикла.

```
(1, 1)          a[1, 1[ = a[2, 0]*2
(1, 2)          a[1, 2] = a[2, 1]*2
…
(2, 1)          a[2, 1] = a[3 ,0]*2
(2, 2)          a[2, 2] = a[3, 1]*2
…
```

Очевидно, что вектор расстояний суть **D** = (–1, 1), а вектор направлений **d** = (“>”, “<”). Значения элементов массива сначала используются, а потом определяются — это антизависимость. При этом внутренний цикл не содержит зависимостей и может быть без ограничений распараллелен. Распараллеливание по внешнему циклу или по двум направлениям допустимо при предварительном резервировании входных данных.

В таком последовательном примере **невозможно** менять местами внешний и внутренний цикл без изменения результатов вычислений.

Наконец, **пример 6.7**. Пусть у нас задан цикл

```
do i = 1, u1
        do j = 1,u2
```

```
            S:      a[i, j] = a[i-1, j+1]*2
        enddo
    enddo
```

Что о нем можно сказать?

Раскручиваем цикл.

```
(1, 1)          a[1, 1[ = a[0, 2]*2
(1, 2)          a[1, 2] = a[0, 3]*2
…
(2, 1)          a[2, 1] = a[1 ,2]*2
(2, 2)          a[2, 2] = a[1, 3]*2
…
```

Очевидно, что вектор расстояний суть **D** = (1, -1), а вектор направлений **d** = (“<”, “>”). Значения элементов массива сначала определяются, а потом используются — это потоковая зависимость. При этом внутренний цикл не содержит зависимостей и может быть без ограничений распараллелен. Правда, при распараллеливании внутреннего цикла все исполнители, завершившие очередную внешнюю итерацию, новую внешнюю итерацию обязаны начинать синхронно! И вот почему. Допустим, что каждый исполнитель выполняет строго одну внутреннюю итерацию. Тогда первый исполнитель, посчитав значение a[1, 1] не может начинать вычисление значения a[2, 1] до тех пор, пока второй исполнитель не рассчитал значение a[1, 2], и т.д. Требуется барьерная синхронизация по окончании внутреннего цикла.

Распараллеливание по внешнему циклу или по двум направлениям невозможно.

Теперь мы можем сформулировать **главное утверждение 6.4**:

Пусть для некоторого многомерного цикла определен вектор направлений d. Истинная зависимость в цикле существует тогда и только тогда, когда крайний левый элемент вектора направлений, отличный от “=”, есть “<”.

Для произвольного цикла возможно распараллеливание по любому индексу, соответствующему компоненту “=” в векторе направлений. Уровень вложенности, соответствующий этому компоненту, может быть обменен местами с любым соседним уровнем вложенности с сохранением результата вычислений. Два соседних уровня вложенности, которым соответствуют одинаковые компоненты вектора направлений, также можно поменять местами. Если в цикле существует антизависимость, то распараллеливание возможно по произвольному количеству индексов при дублировании необходимых входных данных. Распараллеливание для циклов с истинной зависимостью может быть проблематично.

Естественно, что для цикла, в котором зависимости возникают по элементам не одного, а нескольких массивов, решение о возможности распараллеливания принимается по результатам анализа всей совокупности зависимостей.