Лекция 5. Зависимости в простых циклах и их анализ на параллельность

Во многих программах, связанных с математическим моделированием, приходится практически одинаковым образом обрабатывать большие массивы данных. Так что особый интерес представляет анализ существующих последовательных программ на параллелизм по данным. Распараллеливание по данным предполагает разделение массивов на зоны, каждая из которых обрабатывается отдельным исполнителем, — так называемые *зоны ответственности исполнителей*. Подобные вычисления обычно реализуется в последовательном коде с помощью операторов цикла. Мы с вами исследуем наличие зависимостей по данным для циклов, работающих с массивами, и влияние этих зависимостей на возможность параллельного выполнения циклов.

Начнем с простейших циклов — одномерных или невложенных циклов со счетчиком. В различных языках программирования используются похожие, но отличающиеся, конструкции для организации циклов. Тем не менее, для всех циклов со счетчиком (или итерационной переменной) характерно присутствие начального и конечного значения счетчика, и закона его изменения между итерациями. Будем полагать, что счетчик является целой переменной, а законом изменения его значения между итерациями — увеличение или уменьшение значения на некоторую константу. Наиболее просто записать такой цикл в «фортраноподобном» виде

```
do i = b, e, s
        …
enddo
```

Здесь i — итерационная переменная, b — ее начальное значение, e — конечное значение, s — шаг изменения итерационной переменной. Заметим, что счетчик может не принимать своего точного конечного значения при выполнении цикла. Множество всех принимаемых значений итерационной переменной назовем *итерационным пространством* одномерного цикла. Тело цикла лежит между операторами «do» и «enddo».

Путем простой замены итерационной переменной можно привести такой цикл к нормализованному виду

```
do j = 1, jfin, 1
        …
enddo
```

Для краткости записи у нормализованного цикла мы будем опускать шаг счетчика

```
do j = 1, jfin
```

…

enddo

Пусть 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 одномерного массива данных A.

do j = 1, jfin

S_1: …A[f(j)]…

S_2: …A[g(j)]…

enddo

Здесь f(j) и g(j) — некоторые целочисленные функции целого переменного. Для простоты будем считать, что индекс массива A может принимать любое целое значение.

Нашей основной задачей является выяснение того, можно ли разбить итерационное пространство такого цикла — целочисленный отрезок [1, jfin] — на зоны ответственности для параллельного выполнения.

Вспомним, что на самом деле оператор цикла — это просто форма сокращения исходного текста программы. Если убрать это сокращение и развернуть цикл, то получим:

S_1^1: …A[f(1)]…

S_2^1: …A[g(1)]…

S_1^2: …A[f(2)]…

S_2^2: …A[g(2)]…

…

S_1^{jfin}: …A[f(jfin)]…

S_2^{jfin}: …A[g(jfin)]…

Обозначение S_k^i использовано для реинкарнации оператора S_k на *i*-й итерации цикла.

В такой развернутой последовательности следующих друг за другом операторов можно провести анализ их совокупности на зависимость по данным. При анализе нас не будут интересовать зависимости по выходным данным, так как мы знаем, что их легко можно устранить с помощью переименования переменных. Поэтому будем полагать, что для одного оператора в теле цикла обращение к элементу массива A входит в набор входных переменных, а для другого — в набор выходных элементов. Без ограничения общности получаем цикл:

do j = 1, jfin

S_1: A[f(j)] = …

S_2: … = …A[g(j)]…

enddo

Или в развернутом виде:

S_1^1: A[f(1)] = …

S_2^1: … = …A[g(1)]…

S_1^2: A[f(2)] = …

S_2^2: … = …A[g(2)]…

…

S_1^{jfin}: A[f(jfin)] = …

S_2^{jfin}: … = …A[g(jfin)]…

Легко видеть, что условия Бернстайна могут быть нарушены в том случае, если существуют значения итерационной переменной «*j*» λ и κ, $1 \le \lambda \le jfin$, $1 \le \kappa \le jfin$ такие, что $f(\kappa) = g(\lambda)$. Чтобы узнать существуют ли такие значения, нужно решить приведенное уравнение при указанных ограничениях в целых числах. Мы приходим к задаче Диофанта. Возможность решения диофантовых уравнений в общем случае составляет суть десятой проблемы Гильберта, алгоритмическую неразрешимость которой в 1970 году доказал Юрий Матиясевич…

В простых случаях, например, когда f и g — линейные функции, определить, существует ли решение и каково оно, конечно, возможно, но в общем случае — нет. Если решения не существует, то все операторы развернутого цикла независимы друг от друга и могут быть выполнены одновременно различными исполнителями, скажем, каждая итерация цикла — на своем исполнителе. Пусть решение существует, и мы нашли соответствующие λ и κ. Условия Бернстайна нарушены — между операторами есть зависимость. В этом случае оператор S_1^{κ} (где элемент массива A — выходная переменная) называют *источником* (*source*) *зависимости*, а оператор S_2^{λ} (где элемент массива A — входная переменная) называют *стоком* (*sink*) *зависимости*. Вычислим величину $D = \lambda - \kappa$ (из итерации стока вычитаем итерацию источника). Эту величину принято называть расстоянием зависимости цикла.

Расстояние зависимости играет важную роль при анализе цикла на параллельность. Его значение позволяет определять тип возникающей зависимости по данным и возможность разбиения итерационного пространства на зоны ответственности для параллельного исполнения. Разберем несколько примеров.

Пример 5.1. Пусть дан цикл

do i = 1, u

S_1: a[i] = d[i]+5*i

S_2: c[i] = a[i+1]*2

enddo

Вычислим расстояние зависимости, определим тип зависимости и возможность распараллеливания цикла.

Для этого развернем цикл в итерационном пространстве. Нам достаточно развернуть несколько первых итераций:

S_1^1: a[1] = d[1]+5*1

S_2^1: c[1] = a[2]*2

S_1^2: a[2] = d[2]+5*2

S_2^2: c[2] = a[3]*2

…

Как видим, операторы S_2^1 и S_1^2 используют один и тот же элемент массива a[2]. При этом S_1^2 является источником зависимости, а S_2^1 — стоком. Соответственно, $\kappa = 2$, $\lambda = 1$. Расстояние зависимости $D = \lambda - \kappa = -1$. При последовательном выполнении операторов значение a[2] сначала используется, а затем изменяется — это антизависимость. Можно ли выполнить первую итерацию цикла на одном исполнителе, а вторую — на другом? Можно, если первый исполнитель использует старое значение a[2] до того, как второй исполнитель изменит его. Для этого достаточно перед выполнением цикла продублировать необходимые входные данные на исполнителях. Допустимое количество различных исполнителей совпадает с верхней границей цикла u, а возможное ускорение составляет $\approx u$.

Общее утверждение 5.1. Если расстояние зависимости $D < 0$, то между операторами тела цикла существует антизависимость. Цикл может быть распараллелен так, что каждая итерация будет выполняться отдельным исполнителем, если перед началом выполнения итераций продублировать необходимые входные данные на исполнителях.

Пример 5.2. Пусть дан цикл

do i = 1, u

S_1: a[i] = d[i]+5*i

S_2: c[i] = a[i-1]*2

enddo

Вычислим расстояние зависимости, определим тип зависимости и возможность распараллеливания цикла.

Для этого развернем цикл в итерационном пространстве. Нам достаточно развернуть несколько первых итераций:

S_1^1: a[1] = d[1]+5*1

S_2^1: c[1] = a[0]*2

S_1^2: a[2] = d[2]+5*2

S_2^2: c[2] = a[1]*2

…

Как видим, операторы S_1^1 и S_2^2 используют один и тот же элемент массива a[1]. При этом S_1^1 является источником зависимости, а S_2^2 — стоком. Соответственно, $\kappa = 1$, $\lambda = 2$. Расстояние зависимости $D = \lambda - \kappa = 1$. При последовательном выполнении операторов значение a[1] сначала изменяется, а затем используется — это истинная зависимость. Выполнение итераций параллельно невозможно!

Но всегда ли запрещено распараллеливание при наличии истинной зависимости в цикле? Рассмотрим

Пример 5.3. Пусть дан цикл

do i = 1, u

S_1: a[i] = d[i]+5*i

S_2: c[i] = a[i-2]*2

enddo

Вычислим расстояние зависимости, определим тип зависимости и возможность распараллеливания цикла.

Для этого развернем цикл в итерационном пространстве. Нам достаточно развернуть несколько первых итераций:

S_1^1: a[1] = d[1]+5*1

S_2^1: c[1] = a[-1]*2

S_1^2: a[2] = d[2]+5*2

S_2^2: c[2] = a[0]*2

S_1^3: a[3] = d[3]+5*3

S_2^3: c[3] = a[1]*2

…

Как видим, операторы S_1^1 и S_2^3 используют один и тот же элемент массива a[1]. При этом S_1^1 является источником зависимости, а S_2^3 — стоком. Соответственно, $\kappa = 1$, $\lambda = 3$. Расстояние зависимости $D = \lambda - \kappa = 2$. При последовательном выполнении операторов значение a[1] сначала изменяется, а затем используется — это истинная зависимость. НО!!! Значение, вычисленное на 1-й итерации, используется для вычислений только на 3-й итерации. Значение, вычисленное на 3-й итерации, используется затем только на 5-й итерации, и т.д. Аналогично все обстоит и для четных

итераций. Выполнение итераций параллельно возможно на 2-х исполнителях, один из которых реализует только нечетные итерации, а другой — только четные!

Общее утверждение 5.2. Если расстояние зависимости $D > 0$, то между операторами тела цикла существует потоковая зависимость. При $D > 1$ цикл может быть распараллелен не более чем на D исполнителях.

Неисследованным остался только случай $D = 0$

Пример 5.4. Пусть дан цикл

```
do i = 1, u
    S1:   a[i] = d[i]+5*i
    S2:   c[i] = a[i]*2
enddo
```

Вычислим расстояние зависимости, определим тип зависимости и возможность распараллеливания цикла.

Для этого развернем цикл в итерационном пространстве. Нам достаточно развернуть несколько первых итераций:

```
S1^1:   a[1] = d[1]+5*1
S2^1:   c[1] = a[1]*2
S1^2:   a[2] = d[2]+5*2
S2^2:   c[2] = a[2]*2
…
```

Как видим, операторы S_1^1 и S_2^1 используют один и тот же элемент массива a[1]. При этом S_1^1 является источником зависимости, а S_2^1 — стоком. Соответственно, $\kappa = 1$, $\lambda = 1$. Расстояние зависимости $D = \lambda - \kappa = -1$. При последовательном выполнении операторов значение a[1] сначала вычисляется, а затем используется — это потоковая зависимость. Но вся зависимость локализована внутри одной итерации. Поэтому каждую итерацию можно исполнить на своем исполнителе. Допустимое количество различных исполнителей совпадает с верхней границей цикла u, а возможное ускорение составляет $\approx u$. Если мы поменяем местами операторы S_1 и S_2 в теле цикла, то тип зависимости сменится на антизависимость, расстояние зависимости цикла останется прежним, как и возможность его распараллеливания.

Общее утверждение 5.3. Если расстояние зависимости $D = 0$, то тип зависимости между операторами тела цикла в общем случае неопределен. Цикл может быть распараллелен так, что каждая итерация будет выполняться отдельным исполнителем.

Случай D = 0 принято называть странно звучащем на русском языке термином «*зависимость, не зависящая от цикла*» (*loop independent dependence*). Случай $D \neq 0$ называют *зависимостью, связанной с циклом* (*loop carried dependence*).

Необходимо отметить, что отнюдь не всегда расстояние цикла является константой, но, тем не менее, и в таких ситуациях часто возможно использовать это понятие для анализа циклов на параллельность. Например, для цикла

```
do i = 1, u
    S1:   a[i] = d[i]+5*i
    S2:   c[i] = a[2*i]*2
enddo
```

f(*i*) = *i*, a g(*i*) = 2**i*.

Легко видеть, что f(2) = (1), f(4) = g(2), f(6) = g(3) и т.д. Расстояние зависимости точно определить не удается — D = ?. Для первого равенства D = –1, для второго — D = – 2, для третьего — D = – 3. Однако все эти значения меньше нуля, поэтому все зависимости являются антизависимостями и, следовательно, при необходимом дублировании входных данных возможно распараллеливание цикла по итерациям.

Часто встречаются ситуации, в которых зависимости возникают по элементам не одного, а нескольких массивов. Тогда решение о возможности распараллеливания принимается по результатам анализа всей совокупности зависимостей.