ПАРАЛЛЕЛЬНЫЙ АЛГОРИТМ ВЫЧИСЛЕНИЯ ТОЧЕК ГИПЕРПЛОСКОСТИ ФРОНТА ВЫЧИСЛЕНИЙ В ЯЗЫКЕ НОРМА

М.М. Краснов

В статье рассматривается способ параллельного вычисления точек гиперплоскости фронта вычислений в программах, получаемых при трансляции текстов на языке Норма.

Язык Норма [1] – это специализированный язык для записи численных методов решения задач математической физики. Программа на языке Норма содержит описание информационных связей между величинами, задаваемыми на прямоугольных областях (сетках). Операторы языка являются не описаниями действий, которые надо вычислить в данной точке программы, а описаниями соотношений, существующих между вычисляемыми величинами. То, в каком порядке и как (последовательно или параллельно) производить вычисления величин, решает компилятор.

Соотношения между величинами можно условно разделить на простые (когда одна величина зависит от другой) или сложные (когда значения величины в одних точках зависят от значений той же самой величины, но в других точках). К сложным случаям относятся также случаи, когда одна величина (в одних точках) зависит от другой величины, а та, в свою очередь, зависит от значений первой величины (в других точках).

Таким образом, в программе могут возникать циклические зависимости между вычисляемыми величинами. В этом случае на этапе компиляции в графе зависимостей между величинами возникают сильно связанные (циклические) подграфы. Для организации циклических вычислений в выходной программе в этом случае в работах [3], [4] был предложен метод гиперплоскости и фронта вычислений. Выполнение выходной программы состоит из последовательного цикла, продвигающего гиперплоскость фронта вычислений и параллельно исполняемого тела этого цикла. Тело цикла исполняется параллельно по всем точкам гиперплоскости фронта вычислений, лежащих внутри исходной области (сетки). Таким образом, на каждом последовательном шаге решается задача вычисления этого множества точек гиперплоскости. Алгоритм определения этого множества точек, предлагаемый в работах [3], [4] по сути своей последовательный, что снижает степень параллелизма в выходной программе.

В данной работе предлагается алгоритм вычисления искомого множества точек гиперплоскости заранее, до начала выполнения последовательного цикла. Такое множество может быть определено только с избытком – на каждом шаге приходится решать задачу отбрасывания лишних для данного шага точек. Такая организация вычислений значительно повышает степень параллелизма в программе, и особенно хорошо подходит для современных нетрадиционных вычислительных архитектур, о чем подробнее рассказывается в работе.

1. Введение

1.1 Язык Норма

Язык Норма [1] предназначается для записи численных методов решения задач математической физики. Программа на языке Норма, по существу, представляет собой запись расчётных формул и некоторой дополнительной информации, которая необходима транслятору для построения программы на императивном языке программирования.

Программа на Норме не содержит никакой информации о порядке счёта, способах организации циклических процессов. Определение порядка счёта в выходной программе на императивном языке является одной из основных задач транслятора. Информационные связи, задаваемые в предложениях программы, являются исходными данными для решения этой задачи.

В работах [3], [4] предложен алгоритм организации циклического процесса для программ, записанных на языке Норма и позволяющий организовать параллельное исполнение циклов. Опишем кратко этот предложенный алгоритм.

1.2 Основной оператор языка Норма.

Программа на языке Норма содержит описательную и исполнительную части. В описаниях вводятся имена и типы величин, значения которых необходимо вычислить при решении задачи. Исполнительная часть определяет правила, по которым производятся вычисления, в соответствии с семантикой языка [2].

Основным оператором языка является оператор ASSUME, имеющий следующую структуру:

FOR <область> ASSUME <соотношение>.

Определим неформально понятия область и соотношение. Область представляет собой множество

точек с целочисленными неотрицательными координатами в n-мерном пространстве.

Соотношение, указанное в операторе, имеет вид:

$$x^{l}_{i_1,i_2,\dots,i_n} = f_l\left(x^{1}_{A_1},\dots,x^{m}_{A_m}\right), l = 1 \dots m$$

Здесь x^l = имена величин. Множество $\{i_1,i_2,\dots,i_n\}$ представляет множество индексов, по которым ведётся счёт. Индексная конструкция A_k, k=1…m имеет следующую структуру:

$$i_{k_1} \pm \Delta_{k_1},\dots,i_{k_\mu} \pm \Delta_{k_\mu}, 1 \le \mu \le n$$

Δ_{kj} – целые неотрицательные константы.

Соотношение задаёт правило получения значения величины x^l из левой части соотношения по значениям величин из правой части. В каждой точке области определения величина вычисляется только один раз, т.е. переприсваивание величинам в языке Норма запрещено. Оператор ASSUME указывает на необходимость выполнить вычисление значений величины, стоящей в левой части соотношения, для всех точек области, заданной в заголовке оператора. При этом то, в каких местах выходной программы, для каких точек области (всех или части), последовательно или параллельно, будут вычислены величины x^l, решается компилятором. Гарантируется только то, что к концу выполнения выходной программы величина будет вычислена во всех требуемых точках.

Пример. Пусть заданы два оператора:

```
FOR A:(k=1..M) ASSUME
X=f1(Yk-1).
FOR A ASSUME
Y=f2(Xk).
```

Будем считать, что значение величины Y при k=0 вычислено до данных операторов. Очевидно, что порядок выполнения операторов, в котором вначале вычисляются все значения величины X, а затем Y, невозможен. Уже при k=2 для вычисления величины X необходимо, чтобы аргумент, т.е. значение Y при k=1 выло вычислено. Вычисление Y проводится во втором операторе, т.е. в данном случае необходимо поочерёдно проводить вычисления компонентов X и Y.

2. Вычисление параметров гиперплоскости

Одной из задач, которую необходимо решить на этапе компиляции, является задача определения порядка следования операторов (или групп операторов) в выходной императивной программе. Для этого исходная программа на Норме на этапе компиляции представляется в виде графа зависимостей величин.

2.1 Максимально сильно связанный граф

Для исходных отношений определим граф зависимостей Г(V,E,λ), являющийся помеченным направленным графом. V обозначает множество вершин, которые соответствуют именам величин, участвующих в вычислениях. Если величина X вычисляется в нескольких операторах, то ей соответствует несколько вершин. E обозначает множество дуг, отражающих зависимости переменных. Зависимости величины x^l от значений величины x^{l1} с индексной структурой $i_{k1}+\Delta_{k1},\dots,i_{kn}+\Delta_{kn}$ соответствует дуга с пометкой $(\Delta_{k1},\dots,\Delta_{kn})$, идущая от вершины, соответствующей величине x^{l1}, к вершине, соответствующей величине x^l. λ обозначает множество всех пометок дуг графа.

Максимально сильно связанным подграфом (МССГ) графа Г называется сильно связанный подграф графа Г, не содержащийся ни в каком другом сильно связанном подграфе графа Г. Выделив в графе Г все МССГ $Г_1,\dots,Г_n$ (например, методом, предложенным в работе [7]), можно провести частичное упорядочивание $Г_1,\dots,Г_n$: если существуем дуга от $Г_i$ к $Г_j$, то I < j.

Пусть для каждого $Г_i$ построена циклическая программная реализация $P(Г_i)$. Тогда программу по заданному графу Г можно представить в виде последовательности реализаций $P(Г_i)$, i=1…n, учитывая частичный порядок, определённый ранее. Таким образом, достаточно рассмотреть алгоритм построения циклической реализации для операторов, соответствующих вершинам МССГ.

Определим класс операторов, для которых будем проводить построение циклической программы-реализации.

Рассмотрим систему отношений для вычисления величин $x^1,\dots,x^m$ следующего вида:

$$x^{1}_{i_1,\dots,i_n} = f_1\left(x^{1}_{i_1\pm\Delta^1_{11},\dots,i_n\pm\Delta^1_{1n}};\dots;x^{m}_{i_1\pm\Delta^m_{11},\dots,i_n\pm\Delta^m_{1n}}\right)$$

. . . . (1)

$$x^m_{i_1,\dots,i_n} = f_m\left(x^1_{i_1\pm\Delta^1_{m1},\dots,i_n\pm\Delta^1_{mn}};\dots;x^m_{i_1\pm\Delta^m_{m1},\dots,i_n\pm\Delta^m_{mn}}\right)$$

В качестве исходной области, на которой надо вычислить значения этих величин, рассмотрим область, заданную следующими неравенствами:

$M_k \le i_k \le N_k$, k=1…n (2)

Считается, что система (1) соответствует некоторому МССГ.

По заданной системе соотношений (1) требуется построить циклическую программную реализацию, обеспечивающую вычисление величин $x^1,\dots,x^m$ во всех точках области (2), если такое вычисление возможно.

2.2 Фронт вычислений

Рассмотрим следующую схему вычислений в формулах, соответствующих МССГ. Пусть вычисления проводятся в несколько этапов. На первом этапе вычисляются величины, зависящие только от внешних данных (т.е. величин, которые вычисляются за пределами данного МССГ). Множество точек, в которых вычисляется величина *u* на первом этапе, обозначим S_u^1. На втором этапе вычисляются величины, зависящие как от внешних данных, так и от значений, вычисленных на первом этапе. Множество точек, в которых вычисляется величина *u* на втором этапе, обозначим S_u^2. Данный процесс продолжим до тех пор, пока значения величины *u* не будут вычислены во всех точках исходной области. В результате получим последовательность областей вычисления величины u: $S_u^1,\dots,S_u^k$. Вычисление величины u во всех точках области S_u^i проводится параллельно, переход от одной области к следующей последовательный. Такие же системы областей строятся для всех величин, входящих в МССГ.

Множества S_u^i могут иметь достаточно сложную структуру, которая, в свою очередь, приводит к тому, что окажется невозможным записать в виде эффективно реализуемой системы циклов обход всех точек областей S_u^i.

С другой стороны, кажется приемлемым такое представление, когда множество точек, принадлежащих области, можно задать уравнением гиперплоскости следующего вида:

$$\mathcal{L}_u(K): p_1^u i_1 + \dots + p_n^u i_n + \Delta_u = K \quad (3)$$

где p_i^u, Δ_u, K – целые.

Определим последовательность множеств $\mathcal{L}_u(j)$ следующим образом. Пусть $\mathcal{L}_u(1)$ содержит такие точки, вычисление значений величины u в которых зависит только от внешних данных и данные точки удовлетворяют условию

$$p_1^u i_1 + \dots + p_n^u i_n + \Delta_u = 1$$

Множество $\mathcal{L}_u(2)$ состоит из точек, в которых вычисление значений величины *u* зависит только от внешних данных и от значений, вычисленных в точках множества $\mathcal{L}_u(1)$, и удовлетворяющих условию

$$p_1^u i_1 + \dots + p_n^u i_n + \Delta_u = 2$$

Далее этот процесс продолжим до тех пор, пока не будут вычислены все значения величины *u* в исходной области.

Пример. Рассмотрим соотношения

$X_{i,j}=f_1(X_{i-1,j+1};Y_{i-1,j+2})$

$Y_{i,j}=f_2(X_{i+2,j-1};Y_{i+3,j-2})$

заданные на области $1 \le i \le 7$, $1 \le j \le 7$.

В приведённых ниже таблицах значение в клетке (i,j) указывает номер K соответствующего множества S^K. Например, значение в клетке (4,4) таблицы величины X, равное 7, означает, что величина X в точке (4,4) вычисляется на 7 шаге работы модели.

S_X^K

1	1	1	1	1	1	1	7
1	2	2	2	2	2	2	6
1	7	6	4	4	4	3	5
1	8	8	7	6	5	5	4
1	10	9	9	8	7	6	3
1	12	11	10	10	9	8	2
1	14	13	12	11	11	10	1
1	2	3	4	5	6	7	

S_Y^K

6	5	3	3	3	1	1	7
7	7	5	5	4	1	1	6
9	8	7	6	6	1	1	5
11	10	9	8	7	1	1	4
13	12	11	10	9	1	1	3
14	13	13	12	11	1	1	2
1	1	1	1	1	1	1	1
1	2	3	4	5	6	7	

Очевидно, что построить систему циклов для эффективного обхода точек множеств S^K очень трудно.

Построим систему множеств с использованием условия принадлежности плоскости, задаваемой уравнением вида

$\mathscr{L}_X(K)$: -i-2j=K

$\mathscr{L}_Y(K)$: -i-2j+1=K

Ниже приведены таблицы, значения в клетках которых определяют принадлежность соответствующей точки множеству

$\mathscr{L}_X(K)$

7	6	5	4	3	2	1	7
9	8	7	6	5	4	3	6
11	10	9	8	7	6	5	5
13	12	11	10	9	8	7	4
15	14	13	12	11	10	9	3
17	16	15	14	13	12	11	2
19	18	17	16	15	14	13	1
1	2	3	4	5	6	7	

$\mathscr{L}_Y(K)$

8	7	6	5	4	3	2	7
10	9	8	7	6	5	4	6
12	11	10	9	8	7	6	5
14	13	12	11	10	9	8	4
16	15	14	13	12	11	10	3
18	17	16	15	14	13	12	2
20	19	18	17	16	15	14	1
1	2	3	4	5	6	7	

В отличие от элементов множеств S^K множества $\mathscr{L}(K)$ имеют регулярную структуру, что позволяет надеяться на возможность записи последовательности вычислений в виде циклической структуры.

Из приведённых таблиц видно, что на каждом шаге вычислений существует множество точек, в которых величины вычисляются параллельно. Например, значения величины Y в точках (1,7), (3, 6), (5,5), (7,4) и значения величины X в точках (2,6), (4,5), (6,7) вычисляются параллельно на шаге K=8. Все точки, в которых величина X вычисляется параллельно, лежат на одной прямой.

На следующем шаге (при K=9) вычисляются значения величины Y в точках (2,6), (4,5), (6,4) и X в точках (1,6), (3,5), (5,4). (7,3). Все точки, в которых вычисляется величина Y (X) на этом шаге, также лежат на одной прямой.

Предположим, что существует такая гиперплоскость, что можно организовать вычисления, как это показано выше. Введём определение.

Определение. *Фронтом вычислений* величины *и* на шаге K назовём множество точек $\{i_1,\ldots,i_n\}$, в которых значение *и* может быть вычислено параллельно, а сами точки принадлежат гиперплоскости $\mathscr{L}_u$ (K): $p_1^u \cdot i_1 + \ldots + p_n^u \cdot i_n + \Delta^u = K$.

Заметим, что, изменяя K, мы получаем семейство параллельных гиперплоскостей, которые покрывают все точки исходной области.

Далее в работах [3], [4] решается задача определения параметров гиперплоскости фронта вычислений (т.е. целочисленных констант $p_1^u,\ldots,p_n^u$ и Δ^u) методами целочисленного линейного программирования и определения координат точек, лежащих на гиперплоскости фронта вычислений на данном шаге K (т.е. определение параметров циклов, обеспечивающих обход всех точек гиперплоскости).

Проблема состоит в том, что предложенный в работах [3], [4] алгоритм определения множества точек исходной области, лежащих на гиперплоскости на данном шаге, труден для параллельной реализации в выходной программе. Этот подход, возможно, оправдан, когда число процессорных элементов (ПЭ), участвующих в вычислениях, ограничено (единицы или десятки). В современных вычислительных комплексах их число может быть существенно больше и достигать тысяч на один вычислительный модуль. Например, в комплексе К-100 в одном вычислительном модуле около 1,5 тысяч вычислительных ядер трёх графических процессоров. Причём, если при текущем вычислении часть ядер не используются, то они просто простаивают, т.е. «беречь» их большого смысла нет.

3. Параллельное вычисление точек гиперплоскости

Суть предлагаемого подхода к определению точек гиперплоскости на данном шаге вычислений состоит в следующем: мы «расширяем» исходный параллелепипед в некоторых направлениях так, чтобы на всех шагах последовательного цикла вычислений пересечение гиперплоскости фронта вычислений с этим новым расширенным параллелепипедом (её форма и размер) было одним и тем же. Вычисления на каждом шаге ведутся во всех точках этого пересечения. При этом точка может выйти за пределы исходного параллелепипеда («не повезло»). В этом случае вычисления в этой точке не производятся. Если же эта точка оказалась внутри исходного параллелепипеда («повезло»), то в этой точке вычисляется требуемая величина.

По сути, счёт производится «с избытком», «лишние» вычисления просто пропадают. Может даже оказаться так, что на каком-то шаге все вычисления будут «лишними», т.е. на каком-то шаге не будет точек исходного параллелепипеда, лежащих на гиперплоскости фронта вычислений. Важно, что это не влияет на скорость вычислений, т.к. ПЭ всё равно простаивают.

Перейдём к более строгому изложению предмета.

3.1 Определение шага вычислений для точки

Пусть заданы n индексных направлений, $M_j \le i_j \le N_j$, j=1…n и пусть определена гиперплоскость фронта вычислений, т.е. множество целых чисел $\{p_j\}$, j=1…n, соответствующих индексам i_j. Будем считать, что есть, по крайней мере, два p_j, отличных от нуля. Будем также считать числа p_j, отличные от нуля, взаимно простыми, т.е. НОД(p_1,…,p_n)=1. При этом они не обязательно все попарно взаимно простые, но хотя бы одна пара взаимно простых чисел обязательно найдётся, что важно в дальнейшем изложении. Если хотя бы одно из чисел p_1, p_2 равно нулю или НОД(p_1,p_2)≠1, то перенумеруем индексы так, чтобы p_1 и p_2 стали отличными от нуля и взаимно простыми. Равенство нулю коэффициента p_j, соответствующего индексу i_j, означает, что гиперплоскость фронта вычислений параллельна направлению i_j. В этом случае индекс i_j может принимать любые допустимые значения $M_j \le i_j \le N_j$ и другие индексы никак не зависят от этого индекса.

Если окажется, что НОД(p_1,…,p_n)>1, то можно перейти от базового уравнения к приведённому базовому уравнению, для которого НОД(p_1,…,p_n)=1 (см. [5]).

Пусть задана некоторая точка внутри исходного параллелепипеда, т.е. набор целых чисел i_j, для которых выполнено $M_j \le i_j \le N_j$, j=1..n. Зададимся вопросом, на каком шаге будут вычислены величины в этой точке? Пусть K – шаг последовательных вычислений, K≥0. Надо также учесть константу Δ, представляющую из себя отставание данной величины в общих вычислениях. Если значение Δ для данной величины больше нуля, то на первых Δ шагах вычисление данной величины нужно пропустить. В дальнейших рассуждениях будем считать, что K≥Δ.

Обозначим

$$S_j = \begin{cases} M_j, \text{если } p_j > 0 \\ N_j, \text{если } p_j < 0 \end{cases} \qquad (1)$$

Тогда шаг можно вычислить по формуле:

$$K = \Delta + \sum_{j=1}^{n} (i_j - S_j) * p_j \qquad (2)$$

Очевидно, что фронт вычислений на данном шаге K составляют те точки, для которых формула (2) даёт одно и то же значение K.

3.2 Вычисление координат точек для шага вычислений

Выразим из формулы (2) i_1:

$$i_1 = S_1 + \frac{K - \Delta - \sum_{j=2}^{n} (i_j - S_j) * p_j}{p_1} \qquad (3)$$

Эта запись корректна, так как $p_1 \neq 0$ (см. выше). Обозначим $P1 = |p_1|$. Рассмотрим два возможных варианта: P1=1 и P1 > 1.

Пусть P1=1. Тогда формула (3) примет вид:

$$i_1 = S_1 + sign(p_1) * \{K - \Delta - \sum_{j=2}^{n} (i_j - S_j) * p_j\} \qquad (4)$$

Вычисления в этом случае проводятся следующим образом: берутся все возможные комбинации значений индексов $M_j \le i_j \le N_j$, j=2..n. Всего таких комбинаций

$$L = \prod_{j=2}^{n} (N_j - M_j + 1) \qquad (5)$$

Для каждой комбинации индексов параллельно вычисляется значение индекса i_1 по формуле (4). Если вычисленное i_1 попадает в интервал $M_1 \le i_1 \le N_1$, то величина в полученной точке индексного пространства вычисляется, иначе – не вычисляется.

Пусть теперь P1>1. В формуле (3) i_1 и S_1 целые, значит, в дроби числитель должен быть кратен знаменателю, т.е. должно выполняться:

$$K - \Delta - \sum_{j=2}^{n} (i_j - S_j) * p_j \equiv 0 (mod\ P1) \qquad (6)$$

Выделим из этой формулы i_2:

$$i_2 * p_2 \equiv S_2 * p_2 + K - \Delta - \sum_{j=3}^{n} (i_j - S_j) * p_j (mod\ P1) \qquad (7)$$

Т.к. $p_2 \neq 0$ и НОД(p_1,p_2)=1 (см. выше), то существует p_2^{-1} (mod P1). Тогда

$$i_2 \equiv S_2 + p_2^{-1} * \{K - \Delta - \sum_{j=3}^{n} (i_j - S_j)\} (mod\ P1) \qquad (8)$$

Вычисления в этом случае проводятся следующим образом: берутся все возможные комбинации значений индексов $M_j \le i_j \le N_j$, j=3…n. Всего таких комбинаций

$$L_1 = \prod_{j=3}^{n}(N_j - M_j + 1) \qquad (9)$$

Для каждой такой комбинации индексов нужно вычислить все значения индекса i_2 в интервале $M_2 \le i_2 \le N_2$, удовлетворяющие формуле (8). Сначала вычисляем базисное (минимальное неотрицательное) значение индекса i_2^0 по формуле (8). Если при этом окажется, что $i_2^0 < M_2$, то прибавляем

$$i_2^0 = i_2^0 + P1 * \left\lceil (M_2 - i_2^0)/P1 \right\rceil \qquad (10)$$

Мы получим минимальное значение $i_2^0 \ge M_2$, удовлетворяющие формуле (8). Если $N_2 - M_2 < P1 - 1$, то на каких-то шагах K может оказаться, что $i_2^0 > N_2$, т.е. никакие значения индекса i_2 не попадают в допустимый интервал $M_2 \le i_2 \le N_2$. Это означает, что на данном шаге нет точек исходной области, лежащих на гиперплоскости фронта вычислений. Вычисления на данном шаге не проводятся, и делается переход на следующий шаг.

Если i_2^0 попало в допустимый интервал $M_2 \le i_2^0 \le N_2$, то вычисляем

$$i_2^k = i_2^0 + P1 * k, k = 1 \dots k_2 \qquad (11)$$

где k_2 вычисляется по формуле:

$$k_2 = \lfloor (N_2 - M_2)/P1 \rfloor \qquad (12)$$

Получим $L_2 = 1 + k_2$ возможных значений индекса i_2. Последние одно или несколько значений индекса i_2^k могут оказаться недопустимыми, т.е. стать больше, чем N_2.

Всего число возможных значений комбинаций индексов j=2…n.

$$L = L_1 * L_2 \qquad (13)$$

Теперь для каждой такой комбинации индексов параллельно вначале проверяется допустимость индекса i_2, и если условие $M_2 \le i_2 \le N_2$ выполнено, то вычисляется значение индекса i_1 по формуле (3). Если вычисленное i_1 попадает в интервал $M_1 \le i_1 \le N_1$, то величина в полученной точке индексного пространства вычисляется. Если хотя бы один из индексов i_2 и i_1 не попал в допустимый интервал, то величина в точке не вычисляется.

В частном случае n=2 формула (8) принимает вид:

$$i_2 \equiv S_2 + p_2^{-1} * (K - \Delta)(mod\ P1) \qquad (14)$$

В этом случае $L_1 = 1$ и $L = L_2$.

При параллельном вычислении для каждой комбинации значений индексов j=2…n ПЭ получает, помимо прочих, два параметра: порядковый номер комбинации индексов A, $0 \le A < L$ и шаг последовательных вычислений $K \ge 0$.

Покажем, как по значению A вычислить значения всех индексов i_j, j=2…n на неком паскале-подобном языке:

```
for j=n…3
      ij=Mj+A mod (Nj-Mj+1);
A=A div (Nj-Mj+1);
end;
if P1=1 then
      i2=M2+A;
else
      <вычисляем i2^0 по формулам (8) и (10)>
      i2= i2^0+P1*A;
      if i2>N2 then exit;
end;
<вычисляем i1 по формуле (3)>
if i1<M1 or i1>N1 then exit;
<вычисляем требуемую величину в точке i1…in>
```

Кажется логичным попытаться, насколько возможно, уменьшить число L (общее число комбинаций значений индексов, которые надо проверить), чтобы уменьшить требования на число ПЭ. Для этого нужно путём перестановки номеров индексов выбрать такую комбинацию, при которой L минимально. Обязательное условие – в полученной перестановке индексов индексы p_1 и p_2 должны быть взаимно простыми.

В частности, если n=2, то надо сравнить два числа: $(N_2 - M_2)/|p_1|$ и $(N_1 - M_1)/|p_2|$ (делить на p_1 и p_2 можно, так как выше мы уже договорились, что они оба отличны от нуля). Если второе число окажется меньше, то переставить индексы местами.

Если всё же окажется, что ПЭ недостаточно, то можно организовать работу так, чтобы каждый ПЭ получал не порядковый номер комбинации индексов, а диапазон (или список) номеров комбинаций и производил вычисления во всех точках этого диапазона (списка).

3.3 Примеры

В качестве первого примера рассмотрим, как будет выглядеть фронт вычислений для задачи, рассмотренной в предыдущем разделе.

В этой задаче $M_1=M_2=1$, $N_1=N_2=7$, $p_1=-1$, $p_2=-2$, $\Delta_X=0$, $\Delta_Y=1$.

Т.к. $(N_1-M_1)/|p_2|=6/2=3 < (N_2-M_2)/|p_1|=6/1=6$, то индексы нужно поменять местами, т.е. в качестве индекса p_1 должен выступать индекс j, а в качестве p_2 – индекс i. На каждом шаге будут проводиться вычисления в четырёх точках, т.к. $L=L_2=1+(N_1-M_1)/|p_2|=1+3=4$. Фронт вычислений будет выглядеть так:

$\mathscr{L}_X(K)$

1								10
3	2	1						9
5	4	3	2	1				8
7	6	5	4	3	2	1		7
9	8	7	6	5	4	3	2	6
11	10	9	8	7	6	5	4	5
13	12	11	10	9	8	7	6	4
15	14	13	12	11	10	9	8	3
17	16	15	14	13	12	11	10	2
19	18	17	16	15	14	13	12	1
	20	19	18	17	16	15	14	0
			20	19	18	17	16	-1
					20	19	18	-2
							20	-3
1	2	3	4	5	6	7	8	

$\mathscr{L}_Y(K)$

2								10
4	3	2						9
6	5	4	3	2				8
8	7	6	5	4	3	2		7
10	9	8	7	6	5	4	3	6
12	11	10	9	8	7	6	5	5
14	13	12	11	10	9	8	7	4
16	15	14	13	12	11	10	9	3
18	17	16	15	14	13	12	11	2
20	19	18	17	16	15	14	13	1
		20	19	18	17	16	15	0
				20	19	18	17	-1
						20	19	-2
								-3
1	2	3	4	5	6	7	8	

В этих таблицах светло-коричневым цветом выделена исходная область, а розовым – «лишние» точки. Видно, что для разных величин на разных шагах «лишними» могут оказаться от нуля до трёх точек, а на шаге 20 для величины X – все четыре точки. На шаге номер 1 (K=0) величина Y вычисляться не будет, т.к. $\Delta_Y=1$ и шаг с K=0 для величины Y надо пропустить. В принципе несложно определить, что на шаге 20 вычислять величину X уже не нужно и сэкономить на вычислениях.

В качестве второго примера рассмотрим решение двумерной задачи для уравнения теплопроводности на прямоугольной индексной сетке (см. [6], [9]). Применяемые итерационные методы (Якоби, Зейделя, верхней релаксации) состоят в том, что на каждом шаге итерации новое значение для каждой точки находится как среднее арифметическое четырех её соседей. Если мы берём старые значения для всех соседей (метод Якоби), то задача с точки зрения Нормы является простой, т.к. в ней нет зависимостей между величинами и все новые значения можно вычислять параллельно. Если же мы, например, для левого и нижнего соседей берём новые значения, а для верхнего и правого – старые (метод Зейделя и производные от него), то возникает зависимость между величинами и появляется фронт вычислений.

Программа на языке Норма может быть написана примерно так:

```
o:(i=1..M,j=1..N).              ! Вся область
o1:( i=2..N-1,j=2..N-1).        ! Середина области
ol:( i=1,j=1..N).               ! Левая граница
or:( i=M,j=1..N).               ! Правая граница
ot:( i=1,j=2..N-1).             ! Верхняя граница
ob:( i=M,j=2..N-1).             ! Нижняя граница

VARIABLE X DEFINED ON o.    ! Вычисляемая величина
! Начало итерации. X – итерируемая величина, iter – индекс итерации
ITERATION X ON iter
BOUNDARY
        FOR ol,or,ot,ob ASSUME X=XB.  ! Задание граничных условий
END BOUNDARY
INITIAL
        FOR o ASSUME X=XI.            ! Задание начальных условий. Здесь iter=0
END INITIAL
FOR o1 ASSUME X=(X[i-1]+X[j-1]+X[i+1,iter-1]+X[j+1,iter-1])/4.   ! Основное соотношение
EXIT WHEN (условие выхода).   ! Условие выхода из цикла итерации
END ITERATION iter.           ! Конец итерации
```

В этой программе много чего опущено, для нас важно только основное рекурсивное соотношение:

FOR o1 ASSUME X=(X[i-1]+X[j-1]+X[i+1,iter-1]+X[j+1,iter-1])/4.

Заметим, что в языке Норма индексы без смещения можно опускать, таким образом, записи X, X[i], X[I,j], X[I,j,iter] эквивалентны.

Кроме того, итерируемая величина (в данном случае X) внутри итерации получает ещё один виртуальный индекс – индекс итерации (в данном случае iter). Этот индекс может принимать всего два значения – iter (текущая итерация) и iter – 1 (предыдущая итерация). Начальное (минимальное) значение индекса итерации – ноль, на самом первом шаге итерации он равен единице.

Итак, имеется сильно связанная (сама с собой) компонента (X) и появляется фронт вычислений. В данном случае параметры гиперплоскости фронта вычислений $p_1=p_2=1$. В случае одной величины $\Delta=0$.

Пусть M=7, N=9. Тогда для внутренней области o1 границы получаются i=2..6, j=2..8. $L=L_2=5$ (по минимальному значению). Фронт вычислений будет выглядеть так (по горизонтали – индекс j, по вертикали – индекс i снизу вверх):

1	2	3	4	5	6	7	8	9	10	11					6
	1	2	3	4	5	6	7	8	9	10	11				5
		1	2	3	4	5	6	7	8	9	10	11			4
			1	2	3	4	5	6	7	8	9	10	11		3
				1	2	3	4	5	6	7	8	9	10	11	2
-2	-1	0	1	2	3	4	5	6	7	8	9	10	11	12	

Заключение.

В данной работе предлагается алгоритм параллельного вычисления координат точек гиперплоскости фронта вычислений для максимально сильно связанного графа вычисляемых величин для языка Норма.

«Платой» за такую возможность является избыточность вычислений. На каждом последовательном шаге вычислений берётся одно и то же фиксированное количество точек гиперплоскости фронта вычислений и все эти точки обрабатываются параллельно. При этом может оказаться, что часть вычисленных точек гиперплоскости (в худшем случае – все) выходят за границы исходной области вычислений. В этом случае вычисления величины в данных точках не производятся.

В работе предлагается алгоритм минимизации упомянутого фиксированного количества точек гиперплоскости фронта вычислений, в которых производятся вычисления на одном шаге. Это позволяет более эффективно использовать имеющиеся процессорные элементы.

ЛИТЕРАТУРА:

1. А.Н. Андрианов, К.Н. Ефимкин, И.Б. Задыхайло, Н.В. Поддерюгина Язык Норма. Препринт ИПМ им. М.В. Келдыша АН СССР, 1985 г., № 165
2. И.Б. Задыхайло, С.П. Пименов Семантика языка Норма. Препринт ИПМ им. М.В. Келдыша АН СССР, 1986 г., № 139
3. А.Н. Андрианов Организация циклических вычислений в языке Норма. Препринт ИПМ им. М.В. Келдыша АН СССР, 1986 г., № 171
4. А.Н. Андрианов Синтез параллельных и векторных программ по непроцедурной записи на языке Норма. Диссертация на соискание ученой степени кандидата физико-математических наук. Москва, 1990 г.
5. А.Н. Андрианов Система Норма. Разработка, реализация и использование для решения задач математической физики на параллельных ЭВМ. Диссертация на соискание ученой степени доктора физико-математических наук. Москва, 2001 г.
6. Leslie Lamport. The Parallel Execution of DO Loops. Communications of the ACM, February 1974, Volume 17, Number 2
7. Э. Рейнгольт, Ю. Нивергельт, Н. Део. Комбинаторные алгоритмы. Теория и практика. М., Мир, 1980 г.
8. Т. Ху. Целочисленное программирование и потоки в сетях. М., Мир, 1974 г.
9. А.О. Лацис Параллельная обработка данных. М., «Академия», 2010 г.