АНАЛИЗ ПАРАЛЛЕЛЬНЫХ АЛГОРИТМОВ ЧИСЛЕННОГО РЕШЕНИЯ ДИФФЕРЕНЦИАЛЬНЫХ УРАВНЕНИЙ В ЧАСТНЫХ ПРОИЗВОДНЫХ ЭЛЛИПТИЧЕСКОГО ТИПА

Дмитриева О.А.
Кафедра ПМиИ ДонГТУ
E-mail: dmitriv@r5.dgtu.donetsk.ua

Abstract

Dmitrijeva O. The analysis of parallel algorithms for the numerical decision of the partial differential equations of an elliptic type. *This Article deals with the comparative estimations of various parallel algorithms for the decision of the partial differential equations of an elliptic type. The information structure of algorithms is submitted with the influence graph that allow to find set of operations, which can be executed parallel . The qualitative characteristics of the submitted parallel algorithms, in particular, acceleration and efficiency are given.*

Введение

В настоящее время сложность решаемых задач возрастает такими темпами, что поддерживать такие же темпы увеличения быстродействия становится все труднее и дороже. Простой расчет показывает [1], что мы, по-видимому, приближаемся к верхней границе скорости обработки данных, достижимой на цифровых вычислительных машинах, - это следует из основных физических законов. Одним из возможных способов разрешения этого противоречия является установка в вычислительной системе нескольких процессоров, работающих параллельно. Тогда, имея p процессоров, мы получим увеличение быстродействия по сравнению с однопроцессорной ЭВМ в число раз, достаточно близкое к p. Таким образом решается задача более высокого быстродействия. Однако существует ряд трудностей, которые препятствуют развитию параллельных вычислительных систем (ПВС) , к ним относятся [2]: ограниченная область применения ; отсутствие «параллельной математики» ; недостаток знаний о возможности разбиения задач на параллельные ветви ; огромные затраты на разработку языка и программ.

Учитывая все предыдущие замечания, характеризующие достоинства и недостатки параллельных вычислительных систем, можно прийти к выводу, что основная проблема распараллеливания состоит в отсутствии внутреннего параллелизма решаемых задач. Решение этого вопроса может осуществляться при следующем соглашении : задачи не бывают ни полностью параллельными, ни полностью последовательными. И, решая их, мы должны приспосабливаться к этому обстоятельству, а не пытаться бороться с ним. Нельзя полностью исключить последовательные фрагменты алгоритмов, зато можно их минимизировать. Причем степень минимизации зависит от изобретательности автора алгоритма. К тому же существует несколько подходов, применяя которые можно прийти к процессам и структурам данных, лучше приспособленным к параллельной обработке.

1. Анализ информационной структуры программ

Пусть имеется линейная по определению [3] программа, заданная множеством X исполняемых операций. При реализации программы операторы выполняются в строго

определенном порядке, который диктуется самой программой. Этот порядок называется лексикографическим. При реализации на параллельной вычислительной системе алгоритма, записанного в виде последовательной программы, приходится искать ответы на два основных вопроса:

1. Какие подмножества операций в множестве X можно выполнять независимо друг от друга?
2. После каких переупорядочиваний множества X выполнение операций в новом порядке приводит к эквивалентному результату?

Ответ на первый вопрос показывает, какие подмножества операций можно выполнять в параллельном режиме. Ответ на второй вопрос позволяет повысить эффективность использования вычислительной системы за счет изменения порядка вычислений. Для того, чтобы получить содержательные результаты исследований, прежде всего необходимо строго формализовать поставленные вопросы. Поэтому первое, что нужно сделать - это отобразить программу на какой-либо привычный математический объект. Среди таких объектов наиболее подходящими представляются ориентированные графы [4]. Каждая исполняемая операция программы представляет собой вершину графа. На множество вершин естественным образом переносится лексикографический порядок. Пару вершин можно связывать друг с другом в том случае, если между ними имеется какая-то зависимость.

По определению [3], операция x влияет на операцию y, если изменением значения переменной, которую вычисляет операция x можно изменить значение переменной, которую вычисляет операция y. По указанному правилу строится ориентированный граф, причем считают операции алгоритма вершинами графа. Дугами соединяют каждую из пар вершин, в которой одна из соответствующих им операций влияет на другую. Пусть направление дуги совпадает с направлением влияния. Такой граф называется графом влияния. Граф влияния содержит много избыточной информации, и с ним неудобно работать. Поэтому по исходному графу строится минимальный остовный подграф, обладающий следующим свойством: если какая-либо операция x оказывает влияние на операцию y, то в подграфе должен существовать хотя бы один путь, связывающий x с y. Такой граф называют минимальным графом влияния. В силу транзитивности отношения влияния граф влияния оказывается транзитивным замыканием минимального графа.

Характерно, что для операций, которые можно выполнять независимо друг от друга, никакие две соответствующие этим операциям вершины не должны быть связаны дугами графа влияния. Тогда задача распараллеливания сводится к решению задачи отыскания максимально- независимого множества вершин.

2. Распараллеливание решения задачи Дирихле на SIMD структурах

В качестве иллюстрации ко всему вышеизложенному можно рассмотреть решение уравнения в частных производных эллиптического типа. Пусть требуется найти в области G (рис.1) решение дифференциального уравнения (1), удовлетворяющие на границе области G граничным условиям первого рода (2)

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = f(x,y) \qquad (1) \qquad U\big|_{\Gamma}=\varphi(M), \qquad (2)$$

где φ – известная функция, M - точка контура (задача Дирихле).

Для численного решения задачи (1) методом конечных разностей в области G строится сетка равноотстоящих узлов [5]. Поскольку целью данной статьи не является решение конкретной задачи (т.е. получение максимальной точности решения или абсолютной устойчивости), то предпочтение отдается самой простой, исходя из

соображений реализации, разностной схеме. Простейшая схема "крест" осуществляет вычисление значения функции U в заданном узле по формуле :

$$U_{i,j} = \frac{U_{i-1,j} + U_{i+1,j} + U_{i,j-1} + U_{i,j+1}}{4} - h^2 \frac{f_0}{2} \qquad (3)$$

Для упрощения задачи шаги по переменным x и y выбираются равными h, $a_1 = a_2 = 0$, $b_1 = b_2 = k*h$ и вводится сеточную функцию $U_{i,j} = u(x, y)$, определенную только в узлах сетки $\overline{W}_h$. На рис. 1 множество внутренних узлов W_h помечено кружками, множество граничных узлов γ_h - крестиками.

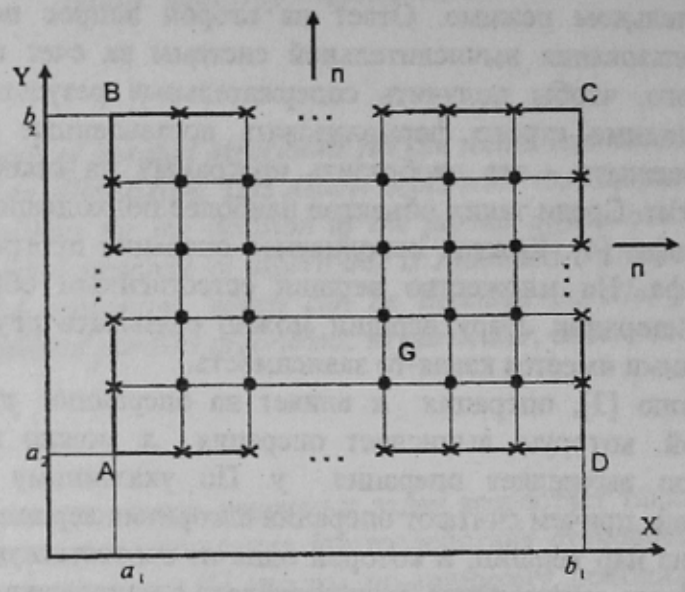

Рис. 1. Общий вид области, для которой находится решение задачи Дирихле

Для получения разностной задачи, которая приближенно будет аппроксимировать на сетке $\overline{W}_h$ краевую задачу (1) - (2), следует заменить дифференциальные операторы в уравнении (1) и граничных условиях (2) конечно-разностными операторами (3). Таким образом, после дискретизации области G и аппроксимации уравнения с граничными условиями исходная краевая задача сведена хотя и к приближенной, но зато более простой разностной задаче. В результате получается линейная система из N^2 уравнений, матрица которой является диагонально разреженной и содержит только пять ненулевых диагоналей. Эффективных прямых методов решения подобных систем в настоящее время не существует. Для больших значений N такие системы решаются итерационными методами.

Пусть решение исходной задачи (1-2) осуществляется в режиме многопроцессорной обработки. Если можно назначить один процессор на один узел расчета, то становится возможным провести расчеты во всех узлах параллельно. Пусть решение одной и той же задачи осуществляется с помощью двух различных алгоритмов, программы которых принадлежат к линейному классу.

При решении задачи Дирихле для уравнения Лапласа с помощью первого алгоритма после параллельного вычисления значений $U_{i,j,k}$ на k- ом слое проверяется необходимость перехода на $k+1$ слой с помощью последовательного отыскания максимума разности $|U_{i,j,k} - U_{i,j,k+1}|$. При этом само абсолютное значение разности также может находиться параллельно. По описанным выше правилам, строится граф влияния, реализующий алгоритм решения поставленной задачи (рис. 2). Оценку эффективности этого параллельного алгоритма можно осуществлять по различным критериям. Наиболее распространенными критериями являются ускорение и

эффективность. Пусть Q - число параметров задачи и $T_p(Q)$ - время выполнения параллельного алгоритма на параллельной ВС с числом процессоров $p > 1$, а $T_1(Q)$ - время работы "наилучшего" последовательного алгоритма. Тогда (4) называется ускорением, а (5) эффективностью параллельного алгоритма.

$$S_y(Q) = \frac{T_1(Q)}{T_p(Q)} \le p \qquad (4) \qquad E_p(Q) = \frac{S_p(Q)}{p} \le 1 \qquad (5)$$

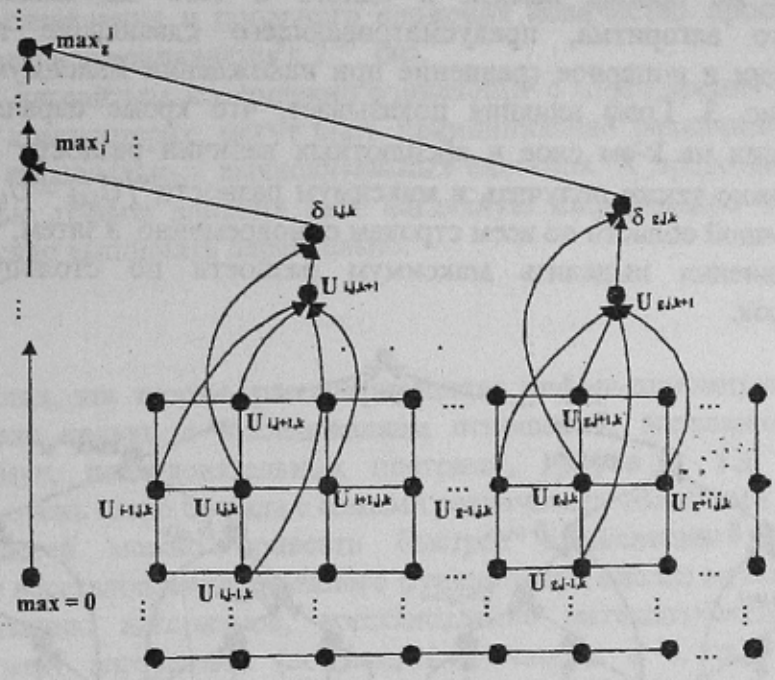

Рис. 2. Граф влияния 1-го алгоритма решения задачи Дирихле

Для расчета ускорения первого параллельного алгоритма необходимо знать время решения, а, следовательно, и количество операций, которые необходимо произвести для нахождения значений на одном слое при решении задачи на последовательной ЭВМ. Если считать, что в исходном уравнении $f(x, y) = 0$, т.е. решается задача Дирихле для уравнения Лапласа, то для всех внутренних узлов $(N-2)^2$ нужно провести 3 сложения и одно деление при вычислении значения функции на новом слое, одно вычитание для оценки абсолютной величины разности $|U_{i,j,k} - U_{i,j,k+1}|$ и одно сравнение с заданной погрешностью ε. Всего на каждом слое при решении задачи на последовательной ЭВМ выполняется $(N-2)^2 * (3+1+1+1) = 6 * (N-2)^2$ операций Количество операций, необходимых для выполнения на каждом слое для первого параллельного алгоритма рассчитывается следующим образом. Одновременно для всех внутренних точек можно получить новые значения за 3 сложения и одно деление, одно вычитание необходимо для отыскания абсолютной величины разности и для всех внутренних точек необходимо выполнить $(N-2)^2$ сравнений с ε. Всего на каждом слое при решении задачи на параллельной ЭВМ по первому алгоритму выполняется $3+1+1+ (N-2)^2 = 5 + (N-2)^2$ операций. Тогда (6) -ускорение первого параллельного алгоритма в системе из N^2 процессоров (без учета времени переноса данных).

$$S_{N^2} = \frac{6*(N-2)^2}{5+(N-2)^2}, \qquad (6) \qquad E = 6/N^2 \qquad (7)$$

если N достаточно велико, $S_{N^2} \approx 6$. Эффективность первого параллельного алгоритма (7). Видно, что *при* $N \to \infty$ она стремится к нулю. Это происходит за счет того, что при отыскании максимального значения абсолютной величины разности $|U_{i,j,k} - U_{i,j,k+1}|$ в

каждом такте фактически остается задействованным только один процессор. Улучшить эту ситуацию можно лишь внеся какие-либо изменения в алгоритм.

Если видоизменить алгоритм, предусмотрев в нем сдваивание при нахождении значения функции на новом слое, а также попарное сравнение разностей при нахождении максимума. то с точки зрения реализации этих программ на однопроцессорных компьютерах фон- неймановского типа они принципиально не будут отличаются. Обе программы выглядят почти одинаково. Их реализация требует одного и того же объема памяти и одного и того же числа операций. Для видоизмененного алгоритма, предусматривающего сдваивание при вычислении значения функции и попарное сравнение при нахождении максимума, граф влияния приведен на рис. 3. Граф влияния показывает, что кроме параллельного расчета значений функции на k-ом слое и абсолютных величин разности $/U_{i,j,k} - U_{i,j,k-1}/$, параллельно можно также получить и максимум разности $/U_{i,j,k} - U_{i,j,k+1}/$для каждой пары узлов сеточной области по всем строкам одновременно, а затем, также с помощью попарного сравнения выделить максимум разности по столбцу получившихся максимумов строк.

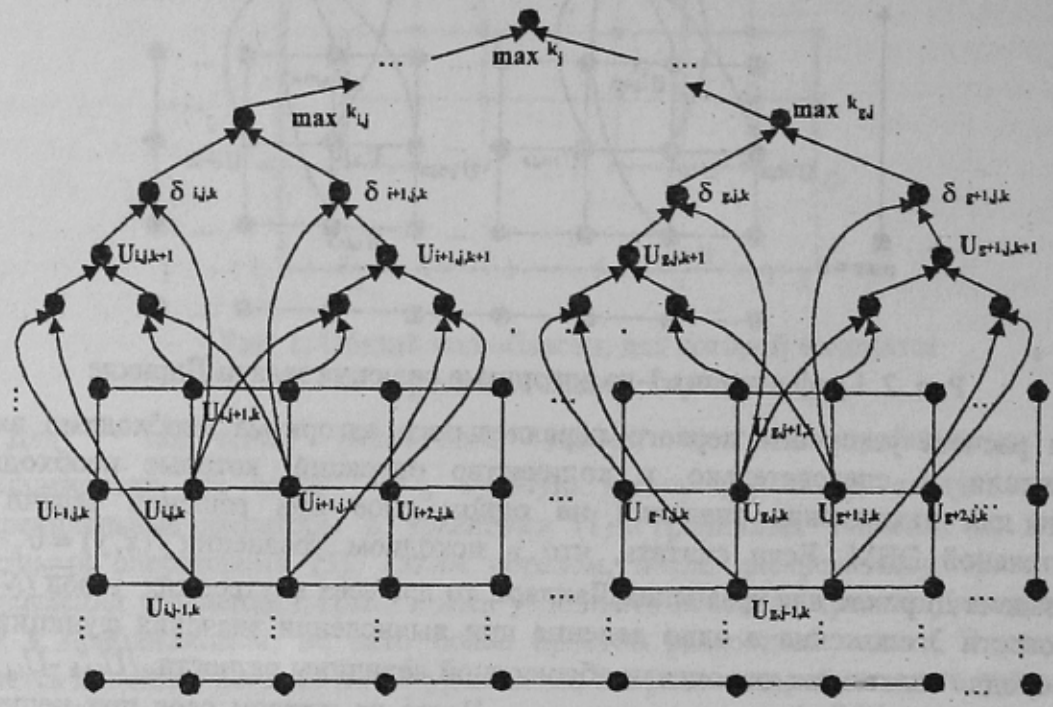

Рис. 3. Граф влияния 2-го алгоритма решения задачи Дирихле

Количество операций, необходимое для нахождения значения функции на новом слое определяется двумя сложениями, одним делением, одним вычитанием для нахождения абсолютной величины разности, $log_2 (N-2)$ сравнениями для нахождения максимальной величины разности в каждой строке, $log_2 (N-2)$ сравнениями для нахождения максимальной величины разности в столбце, составленном из максимумов разностей строк и одного сравнения максимума с ε. Всего $2 + 1 + 1 + log_2 (N-2) + log_2 (N-2) + 1 = 5 + 2*log_2 (N-2)$.

Ускорение второго параллельного алгоритма в системе из N^2 процессоров, если временем переноса данных при этом можно пренебречь:

$$S_{N^2} = \frac{6*(N-2)^2}{5+2*\log_2(N-2)} \approx 3*N^2 / \log_2(N) \qquad (8)$$

Поскольку при вычислении значений функции в каждом узле использовался алгоритм сдваивания, то количество использованных процессоров должно быть вдвое большим, чем количество узлов. Тогда эффективность

$$E=\frac{3*N^2}{\log_2 N*2*N^2}\approx\frac{3}{2*\log_2 N}. \qquad (9)$$

Эффективность алгоритма достаточно мала, а при $N\to\infty$ она стремится к нулю, но гораздо медленнее, чем в описанном выше алгоритме. Это происходит за счет того, что при выполнении сдваивания и попарного сложения количество простаивающих процессоров на k-ом этапе увеличивается в 2^{k-1} раз.

Таким образом, алгоритмы, совершенно одинаковые с точки зрения реализации на последовательных компьютерах, могут быть принципиально различными с точки зрения реализации на параллельных вычислительных системах. А представление этих алгоритмов с помощью графов влияния дает наглядную информацию об участках программ, которые можно выполнять параллельно.

Выводы

На первый взгляд, эта теория, рассматривающая информационную структуру алгоритмов имеет дело только с исследованием отношений, записанных в виде математических формул, последовательных программ, графов и т.д. Однако в действительности она очень тесно связана с самыми различными областями. В качестве примера таких областей можно привести быстрое вычисление градиента и производной, быстрое восстановление линейного функционала, анализ влияния ошибок округления, декомпозицию алгоритмов, восстановление математических формул, обнаружение узких мест алгоритмов (по памяти, точности и т. п.), разработку параллельных численных методов, разработку переносимых прикладных программ, использование распределенной и иерархической памяти, выбор оптимальной архитектуры компьютера, построение математических моделей систолических массивов и других спецпроцессоров, разработку параллелизирующих компиляторов и многие другие области.

Литература

1. Валях Е. Последовательно - параллельные вычисления : Пер. с англ. - М.: Мир, 1985. - 456 с.
2. Ортега Дж. Введение в параллельные и векторные методы решения линейных систем: Пер. с англ. - М.: Мир, 1991. - 367с.
3. Воеводин В.В. Информационная структура алгоритмов.- М.: Изд-во МГУ, 1997.- 139с.
4. Воеводин В.В. Математические основы параллельных вычислений.- М.: Изд-во МГУ, 1991.- 345с.
5. Самарский А.А. Введение в теорию разностных схем. - М.: Наука, 1971.- 388 с.