Космічна наука і технологія. 1998. Т. 4, № 4. С. 93–96.

УДК 681.3

Структуры данных для высокопроизводительных систем обработки космической информации

Б. Т. Деркач

Фізико-механічний інститут ім. Г. В. Карпенка НАН України, Львів

Надійшла до редакції 13.04.98

Розкривається взаємозв'язок між структурами даних і структурою паралельного алгоритму, а також досліджується їх вплив на ефективність реалізації паралельних алгоритмів на обчислювальних системах різної архітектури. Структури даних, що розглядаються, широко застосовуються для розв'язання задач обробки космічної інформації: задачі управління і навігації (лінійна алгебра), кореляційний аналіз, спектральний аналіз (ортогональні перетворення), дослідження і аналіз нестаціонарних сигналів (перетворення wavelet) та інші.

ВВЕДЕНИЕ

Степень распараллеливания процесса вычислений определяется интенсивностью использования возможности одновременного изменения значений многих переменных. Это особенно важно при организации синхронных параллельных вычислений, в которых элементарные операторы изменяют одновременно значения большого числа переменных. На реализацию таких вычислений ориентирована разработка однородных вычислительных систем [1, 2], а также использование матричных и векторных процессоров типа ILLIAC. Существенное значение для разработки параллельных алгоритмов и проектирования новых структур вычислительных систем имеет изучение структур данных и заданных на них преобразований. Использование различных структур данных влияет на эффективность решения задачи на параллельной вычислительной системе. При проектировании специализированных вычислительных систем по заданной структуре алгоритма структуры данных существенным образом влияют на выбор архитектуры реализующей системы.

СТРУКТУРЫ ДАННЫХ В ЗАДАЧАХ ЛИНЕЙНОЙ АЛГЕБРЫ

Формально структуру данных можно определить как пару (D, ρ), где $D = \{d_1, \ldots, d_n\}$ — множество объектов данных и $\rho = \{Q_1, \ldots, Q_r\}$ — множество бинарных отношений, заданных на D, таких, что $Q_i \subseteq D \times D$. Задавая определенные свойства отношений из ρ, можно получать различные структуры данных. Наиболее важной является структура типа

$$(D, \rho) = (D, \{\leq\}), \qquad (1)$$

где $\leq$ обозначает отношение линейного порядка, которое является рефлексивным, асимметричным и транзитивным. Это отношение задает нумерацию на множестве данных $\{d_1, \ldots, d_n\}$, т. е. для каждого элемента множества определяется его относительная позиция в D. Такие структуры получили название линейных списков. Простым обобщением последних являются двумерные и многомерные списки. Двумерный список состоит из линейных списков строк $(R_i, \{\leq R_j\})$, $i = 1, \ldots, m$, и столбцов $(C_j, \{\leq C_j\})$, $j = 1, \ldots, m$. Эти два списка являются взаимно ортогональными, так как задание нумерации в одном из списков определяет положение элементов в другом и наоборот. Очевидно, что матрицу $n \times n$ можно представить двумерным списком одного из вышеназванных видов. Структуру данных при этом определим так:

$$(D, \rho) = \left(\bigcup_{i=1}^{n} R_i, \{\leq R_1, \ldots, \leq R_n\}\right). \qquad (2)$$

Представление метода Гаусса и Гаусса – Жордана в матричной форме позволяет непосредственно перейти к построению параллельных алгоритмов поскольку они описываются с помощью операций

над структурами данных типа (1). Как в том, так и в другом случае вычисления сводятся к выполнению базовых операций: умножению вектора на скаляр и суммированию двух векторов, которые являются операциями над структурами данных типа (1). Под алгоритмом типа (1) будем понимать параллельные вычисления над структурными данными типа (1). При этом справедливы следующие теоремы [2].

Теорема 1. С помощью алгоритма типа (1) прямой ход гауссова исключения для матрицы размерностью $n \times n$ выполняется на n процессорах за $n(n+2)$ тактов.

Теорема 2. Обратный ход с помощью алгоритма типа (1) для матрицы размерностью $n \times n$ на n процессорах выполняется за $2(n-1)$ тактов.

Структура данных типа (1) может быть приведена к виду

$$(D, \rho) = \left(\bigcup_{i=1}^{n} R_i, \{\leq\} \right).$$

Такая структура получается путем записи элементов матрицы в лексикографическом порядке (по столбцам). В результате получаем линейный список $R_k = \{R_k|1|, \ldots, R_k|n|\}$, в котором элементам k-го столбца матрицы **A** соответствует подмножество $\{d|(k-1)n+1|, \ldots, d|kn|\}$ множества D. Использование представления матрицы структурой вида (2) позволяет провести глубокое распараллеливание алгоритма решения систем линейных алгебраических уравнений на n^2 процессоров. Параллельные вычисления над структурами данных типа (2) назовем алгоритмом типа (2). При таком представлении системы уравнений справедливы теоремы [2].

Теорема 3. С помощью алгоритма типа (2) прямой ход гауссова исключения для матрицы размерностью $n \times n$ на n^2 процессорах выполняется за $3n$ тактов.

Теорема 4. Число тактов для выполнения обратного хода в алгоритме типа (1) и в алгоритме типа (2) одинаково.

СТРУКТУРЫ ДАННЫХ ДЛЯ ВЫЧИСЛЕНИЙ С РАЗРЕЖЕННЫМИ МАТРИЦАМИ КОЭФФИЦИЕНТОВ

Использование свойств разреженности матрицы коэффициентов имеет существенное значение при построении как последовательных, так и параллельных алгоритмов решения систем линейных алгебраических уравнений большой размерности. В случае последовательных алгоритмов можно, используя специальные приемы, хранить в памяти ЭВМ только ненулевые элементы матрицы [3], а также исключить операции с нулевыми элементами. Это позволяет создавать эффективные алгоритмы решения больших систем уравнений прямыми методами. При построении параллельных алгоритмов, как будет показано ниже, учет разреженности позволяет значительно сократить число процессоров, а в некоторых случаях — число тактов. Нельзя забывать, что при решении системы уравнений с использованием метода исключения Гаусса или другими прямыми методами многие нулевые элементы матрицы становятся ненулевыми. Процесс появления новых ненулевых элементов, называемый заполнением, может значительно снизить эффективность применения прямых методов. Его анализ и ограничение имеют большое значение.

Для описания структуры разреженной матрицы может быть использован граф данных. Вычисления только с ненулевыми элементами матрицы представляются преобразованиями на вершинах графа. Использовать граф для представления структуры разреженной матрицы впервые было предложено в работе [4], а затем этот подход развивался в работах [5, 6]. Такой теоретико-графовый подход дает ряд существенных преимуществ, основным из которых является то, что можно легко осуществлять доступ к ненулевым элементам матрицы. Использование этого свойства позволяет строить эффективные алгоритмы для целого класса матриц, имеющих ту же структуру ненулевых элементов.

Пусть $\mathbf{Ax} = \mathbf{b}$ — система линейных алгебраических уравнений с разреженной матрицей $\mathbf{A} = \|a_{ij}\|$ размерностью $n \times n$. Представим структуру ненулевых элементов ориентированным графом $G_\alpha = (V, E, \alpha)$, где $V = \{1, \ldots, n\}$ — конечное множество элементов, называемых вершинами, а $E \subseteq V \times V$ — множество дуг. Дуга (i, j) существует только в том случае, если $a_{ij} \neq 0$, т. е. $E = \{(i, j) / a_{ij} \neq 0\}$. Если дуга направлена из данной вершины в нее же, то она называется петлей. Число дуг в таком графе равно общему числу ненулевых элементов матрицы. Симметричной матрице соответствует неориентированный граф. Для графа $G_\alpha = (V, E, \alpha)$ нумерацией называется биекция α: $\{1, \ldots, n\} \leftrightarrow V$. Такой граф называется пронумерованным ориентированным. Каждой его вершине поставим в соответствие величину $b(i) = -b_i$, каждой дуге (i, j) — величину $a(i, j) = a_{ij}$, если $i \neq j$, и $a(i, j) + 1$, если $i = j$. Тогда система уравнений запишется следующим образом:

$$Q = \left\{ \sum_{i,j \in E} x(j) a(i, j) + b(i) = x(i) \right\},$$

$$1 \le i \le n,\ 1 \le j \le n.$$

Теперь можно определить процесс гауссова исключения на графе G_α. При переходе от $G_\alpha^{(1)}$ к $G_\alpha^{(2)}$ необходимо провести следующие вычисления на графе: для $(i, 1) \in E$ $b(i) = b(i) \cdot [1 - a(1, 1)]^{-1} \cdot a(i, 1)$. Если в графе есть дуги $(i, 1) \in E^{(1)}$ и $(1, j) \in E^{(1)}$ и при этом $(i, j) \notin E^{(1)}$, то необходимо добавить новую дугу (i, j) с величиной $a(i, j) \ne 0$. В общем случае имеем $a(i, j) = a(i, j) + a(i, k)[1 - a(k, k)]^{-1} \cdot a(k, j)$.

Прямой ход гауссова исключения представляется теперь как цепочка преобразований $G_\alpha^{(1)} \to G_\alpha^{(2)} \to \to \ldots G_\alpha^{(2)}$.

Рассмотрим, используя теоретико-графовый подход, как разреженность матрицы **A** влияет на основные характеристики параллельных алгоритмов, такие как число тактов и процессоров, для алгоритмов типа (1) и (2).

Обозначим через $d_+^{(i)}(i)$, $i \in V^{(k)}$ число дуг, входящих в вершину i, а через $d_-^{(k)}(i)$, $i \in V^{(k)}$ — число дуг, исходящих из вершины i. Будем считать, что граф G_α является сильно связным, т. е. для любой пары вершин i, j в графе существует связывающий их путь.

Пусть задана система линейных алгебраических уравнений вида $\mathbf{Ax} = \mathbf{b}$ с разреженной матрицей $n \times n$. Тогда справедлива следующая теорема [2]:

Теорема 5. Для выполнения прямого хода гауссова исключения по алгоритму типа (1) необходимое число процессоров p определяется следующим образом:

$$p = \max_k d_+^{(k)}(i),\ i \in V,\ k = 1, \ldots, n - 1.$$

Теорема 6. Прямой ход гауссова исключения по алгоритму типа (1) выполняется за T_p тактов, где $T_p = \sum_{k=1}^{n-1} 2d_-^{(k)}(i) + 3$.

Теорема 7. Для вычисления прямого хода гауссова исключения по алгоритму типа (2) необходимое число процессоров составит

$$p = \max_k d_+^{(k)}(i) \cdot d_-^{(k)}(i),\ k = 1, \ldots, n.$$

Теорема 8. Число тактов для разреженной матрицы при использовании алгоритма типа (2), как и для заполненной, равно $T_p = 3n$.

Процесс заполнения можно значительно уменьшить, если использовать стратегию выбора ведущего элемента на каждом шаге гауссова исключения, приводящую к минимальному заполнению. Одним из наиболее простых критериев выбора главного элемента является критерий Марковича. Ведущий элемент выбирается так, чтобы суммарное число ненулевых элементов в k-й строке и в k-м столбце было минимальным.

Как известно, выбор главного элемента сильно влияет на устойчивость численного решения системы. При этом, к сожалению, При этом, к сожалению, стратегия минимального заполнения и численной устойчивости конфликтуют при выборе ведущих элементов. Поэтому, как правило, используются компромиссные стратегии.

СТРУКТУРЫ ДАННЫХ ДЛЯ ОБРАБОТКИ ИЗОБРАЖЕНИЙ

В последние годы активизировался интерес к представлению бинарных изображений с помощью графов. Рассматриваются представления деревьями степени 2 — бинарными деревьями, степени 4 — квадрант-деревьями, степени 8 — октальными деревьями и другие. По-видимому, наиболее оптимальным является представление изображения квадрант-деревьями. Пусть имеется 2^n на 2^n бинарное изображение I. Для него квадрант-дерево строится следующим образом. Если изображение состоит из одних нолей или из одних единиц, то в этом случае изображение состоит из одной вершины, которой присваивается значение 0 или 1. В противном случае из корневой вершины, представляющей все изображение, проводятся дуги к четырем вершинам, соответствующим квадрантам изображения. Для каждой вершины процесс повторяется снова. Процесс заканчивается, если вершина дерева соответствует блоку изображения, состоящему из одних нолей или одних единиц. Максимальное количество уровней в дереве для изображения 2^n на 2^n равно n. На самом нижнем уровне вершина дерева соответствует одному элементу изображения. Каждый узел квадрант-дерева может храниться в форме записи с шестью полями, содержащими указатели на предыдущую вершину, следующие вершины и значение типа узла (черный, белый или серый). Направления к подчиненным вершинам определяются из набора: верхний левый, верхний правый, нижний левый, нижний правый и кодируются соответственно 1, 2, 3, 4. Каждый узел дерева представлен своим кодом размещения, который может определяться из выражения [7]:

$$z_i = \begin{cases} 0, & i = m, \\ z_i + x_{i-1}, & m < i < n, \end{cases}$$

где z — код размещения вершины (номер), а $x_i = 1, 2, 3, 4$.

Такой подход к кодированию бинарных изображений позволяет получить значительную экономию памяти, необходимой для хранения изображения, кроме того, позволяет значительно повысить эффективность некоторых операций для обработки изображений.

Одним из возможных способов реализовать такой подход с максимальной эффективностью состоит в использовании ассоциативных алгоритмов и структуры для выполнения операций над изображениями, кодированными с помощью квадрант-деревьев. Кодированное изображение в ассоциативной памяти хранится в виде кодированных наборов. Такое представление позволяет в ассоциативном процессоре эффективно реализовать сортировку вершин по уровням, что в свою очередь значительно убыстряет сравнение исходного изображения с эталоном. Пусть $I^{(k)}$ и $I'^{(k)}$ — множество кодированных наборов k-го уровня квадрант-деревьев изображений I и I'. Если на k-м уровне эти наборы сильно отличаются друг от друга, то тогда отпадает необходимость проведения сравнения на более низких уровнях. Поэтому операция сравнения изображений на ассоциативном процессоре будет очень быстрой.

Очень эффективной будет реализация метода моментов на ассоциативном параллельном процессоре. Момент изображения определяется так: $m_{ij} = \Sigma\Sigma f(x, y)x^i y^i$, где сумма берется по всем точкам изображения. Для бинарного изображения в состав суммы войдут только те точки, в которых изображение равно 1. В частности, момент m_{00} определяется количеством единиц в изображении. Так как момент изображения равен сумме моментов его блоков, то становится ясным смысл применения ассоциативного процессора для вычисления моментов изображения. Вычисление момента сведется к поиску черных вершин в квадрант-дереве, каждое из которых в зависимости от уровня k, представляет собой блок размером 2^k на 2^k.

ЗАКЛЮЧЕНИЕ

На примере алгоритмов решения систем линейных алгебраических уравнений прямыми методами при использовании структур данных типа (1) и типа (2) видно, что получаются параллельные алгоритмы с существенно различными характеристиками.

При решении систем линейных алгебраических уравнений с разреженными матрицами коэффициентов и использовании представления матрицы ориентированным графом показана возможность построения оптимальных по количеству тактов и ширине параллельных алгоритмов. Тем не менее, эти алгоритмы не могут быть эффективно реализованы на параллельных структурах типа векторных или матричных процессоров, поскольку они не позволяют осуществлять эффективный доступ к данным, представленным в виде графовых или древовидных структур. Затраты на межпроцессорные пересылки данных не позволяют получить общий эффект, который мог бы быть достигнут за счет исключения операций с нулевыми элементами матрицы.

Аналогичная ситуация прослеживается на примере алгоритмов обработки изображений. Рассмотренные алгоритмы выполнения операций с изображениями на ассоциативном параллельном процессоре показывают возможность эффективного применения ассоциативной обработки.

Поскольку древовидные структуры данных широко используются и при решении многих других задач, особенно информационно-логического характера, весьма важным является разработка вычислительных структур, позволяющих реализовать такие операции с наибольшей эффективностью.

1. Грицик В. В., Деркач Б. Т., Куликов С. П. и др. Повышение эффективности и производительности решения систем линейных алгебраических уравнений с разреженными матрицами коэффициентов. — Львов, 1980. — 54 с. — (Препринт / АН УССР. Физ.-мех. ин-т; № 31).
2. Параллельная обработка информации. — Киев: Наук. думка, 1986. — Т. 3. — 288 с.
3. Brameller A., Allan R. Sparsity its practical application to system analys. — London: 1976. — 192 p.
4. Duff I. S., Reid J. K. A comparison of sparsity orderings for obtaining a pivot sequence in Gaussian Alimination // J. Inst. Math. and Appl. — 1971. — N 4. — P. 281 — 291.
5. Gentelman W. M., George A. Sparse matrix software // Sparse matrix comput. — New York: Acad. press, 1976. — P. 243 — 262.
6. Heller D. A. Survey of parallel algorithms in numerical linear algebra // SIAM. — 1978. — **20,** N 4. — P. 740 — 777.
7. Samet H. Data structures for quadtree approximation and compression // Communs ACM. — 1985. — **28,** N 5. — P. 973 — 979.

DATA STRUCTURES FOR SPACE DATA COMPUTATION ON HIGH-PERFORMANCE COMPUTER SYSTEMS

B. T. Derkach

Data structures for the most important computing algorithms are proposed. Such data structures can be used for an efficient implementation of the linear algebra, Fourier transform, wavelet transform, image processing, and others algorithms.