КОНЦЕПЦИЯ НЕОГРАНИЧЕННОГО ПАРАЛЛЕЛИЗМА

Концепция неограниченного параллелизма – это способ понимания, конструктивный принцип построения параллельных алгоритмов, в основе которого лежит предположение, что алгоритм реализуется на параллельной вычислительной системе, не накладывающей на него никаких ограничений. Считается, что процессоров[1] может быть сколь угодно много, они работают а синхронном режиме, имеют общую память, любые передачи информации осуществляются мгновенно и без конфликтов.

Основная цель – получение алгоритмов минимальной высоты, так как в такой модели вычислений высота определяет время реализации алгоритма.

Утверждение. *Пусть на вычислительной системе, состоящей из s процессоров с пиковой производительностью π, реализуется некоторый алгоритм. Пусть высота параллельной формы, соответствующей реализации алгоритма, равна m и всего в алгоритме выполняется N операций. Тогда максимально возможное ускорение системы равно $\dfrac{N}{m}$.*

Доказательство. Воспользуемся формулой для выражения ускорения системы через загруженности процессоров: $R = \sum\limits_{i=1}^{s} p_i$. Предположим, что за время T реализации алгоритма i-й процессор выполнил N_i операций. По определению $p_i = \dfrac{N_i/\pi}{T}$. Если процессоров достаточно, то операции одного яруса параллельной формы система может выполнить за время, равное или большее времени $\dfrac{1}{\pi}$ выполнения одной операции; время T выполнения всех ярусов больше или равно $\dfrac{m}{\pi}$. Тогда

$$R = \sum_{i=1}^{s} \frac{N_i/\pi}{T} = \frac{N}{\pi T} = \frac{m/\pi}{T} \cdot \frac{N}{m} \le \frac{N}{m}$$

при любом числе процессоров. □

Пример 1 (процесс сдваивания).

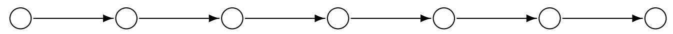
Рассмотрим параллельные формы двух алгоритмов вычисления произведения $a_1 \cdot a_2 \cdot ... \cdot a_N$. Пусть $N = 8$.

1. Обычная схема.

Ярус 1: $a_1 \cdot a_2$
Ярус 2: $(a_1 a_2) \cdot a_3$
Ярус 3: $(a_1 a_2 a_3) \cdot a_4$
Ярус 4: $(a_1 a_2 a_3 a_4) \cdot a_5$
Ярус 5: $(a_1 a_2 a_3 a_4 a_5) \cdot a_6$
Ярус 6: $(a_1 a_2 a_3 a_4 a_5 a_6) \cdot a_7$
Ярус 7: $(a_1 a_2 a_3 a_4 a_5 a_6 a_7) \cdot a_8$

[1] Под процессором понимается одно вычислительное ядро.

Изобразим граф алгоритма:



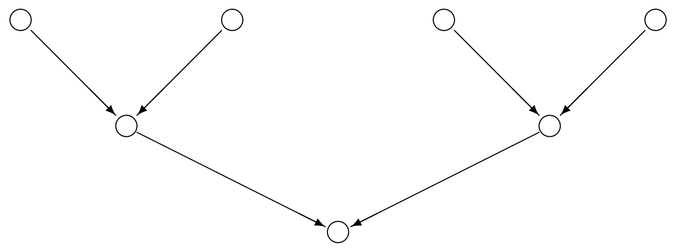
В общем случае высота алгоритма равна $N-1$, ширина алгоритма равна 1.

2. Процесс сдваивания.

Ярус 1: $a_1 \cdot a_2 \quad a_3 \cdot a_4 \quad a_5 \cdot a_6 \quad a_7 \cdot a_8$

Ярус 2: $(a_1a_2) \cdot (a_3a_4) \quad (a_5a_6) \cdot (a_7a_8)$

Ярус 3: $(a_1a_2a_3a_4) \cdot (a_5a_6a_7a_8)$



В общем случае высота алгоритма равна $\lceil \log_2 N \rceil$, ширина равна $\lceil N/2 \rceil$.

Отметим, что рассмотренные алгоритмы математически эквивалентны, но имеют разные вычислительные свойства, в том числе и разные параллельные свойства.

Утверждение. *Пусть с помощью операций, имеющих не более p аргументов, вычисляется значение некоторого выражения, существенным образом зависящего от N переменных. Тогда высота алгоритма, позволяющего вычислить это выражение, не меньше $\log_p N$.*

Действительно, рассмотрим параллельную форму алгоритма вычисления выражения. Пусть на нулевом ярусе расположены операции, соответствующие вводу значений входных переменных, на ярусе T расположена операция, вычисляющая конечный результат; T – высота параллельной формы. Так как любая операция имеет не более p аргументов, то на ярусе $T-1$ находится не более p операций, на ярусе $T-2$ – не более p^2 операций. На нулевом ярусе находится не более p^T операций. Так как $p^T \geq N$, то $T \geq \log_p N$.

Пример 2 (умножение матрицы на вектор; перемножение матриц).

Рассмотрим задачу умножения матрицы A порядка N на N-мерный вектор b: $c_i = \sum\limits_{j=1}^{N} a_{ij}b_j$. На первом шаге можно вычислить N^2 произведений $a_{ij}b_j$. Далее, используя процесс сдваивания, за $\lceil \log_2 N \rceil$ шагов можно вычислить N сумм, определяющих координаты вектора c. Высота алгоритма имеет порядок $\log_2 N$, ширина алгоритма равна N^2.

Задачу перемножения двух матриц порядка N можно рассматривать как задачу вычисления N произведений одной матрицы на вектор. Если все эти произведения вычислять по описанному алгоритму, то получим алгоритм с высотой порядка $\log_2 N$ и шириной N^3.

Пример 3 (процесс рекуррентного сдваивания; решение треугольной системы).

Пусть заданы матрицы A_{ij}, $1 \leqslant i \leqslant s$, $1 \leqslant j \leqslant r$, векторы $b_1, ..., b_s$ и векторы x_0, x_{-1}, ... , x_{-r+1} порядка n. Требуется вычислить векторы x_i, $1 \leqslant i \leqslant s$, с помощью рекуррентных соотношений

$$x_i = A_{i1}x_{i-1} + ... + A_{ir}x_{i-r} + b_i, \tag{1}$$

или, в более подробной записи,

$$\begin{aligned}
&x_1 = A_{11}x_0 + A_{12}x_{-1} + ... + A_{1r}x_{1-r} + b_1,\\
&x_2 = A_{21}x_1 + A_{22}x_0 + ... + A_{2r}x_{2-r} + b_2,\\
&x_3 = A_{31}x_2 + A_{32}x_1 + ... + A_{3r}x_{3-r} + b_3,\\
&\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\\
&x_s = A_{s1}x_{s-1} + A_{s2}x_{s-2} + ... + A_{sr}x_{s-r} + b_s.
\end{aligned}$$

На основе такого типа рекуррентных соотношений построены многие численные методы линейной алгебры, математической физики и анализа.

Пусть, например, $r = 1$, $A_{i1} = B$, все векторы $b_1, ..., b_s$ равны. Получим метод простой итерации решения систем линейных алгебраических уравнений:

$$x_i = Bx_{i-1} + b.$$

Пусть теперь $s = N$, $r = i$, $n = 1$, $x_0 = 0$, $A_{ij} = -a_{i\ i-j}$ ($i > j$, другие случаи при $r = i$, $x_0 = 0$ не рассматриваются). Получим соотношения

$$\begin{aligned}
&x_1 = b_1,\\
&x_2 = -a_{21}x_1 + b_2,\\
&x_3 = -a_{31}x_1 - a_{32}x_2 + b_3,\\
&\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\\
&x_N = -a_{N1}x_1 - a_{N2}x_2 - ... - a_{NN-1}x_{N-1} + b_N
\end{aligned} \tag{2}$$

для решения системы линейных алгебраических уравнений

$$\begin{cases}
x_1 = b_1,\\
a_{21}x_1 + x_2 = b_2,\\
a_{31}x_1 + a_{32}x_2 + x_3 = b_3,\\
\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\\
a_{N1}x_1 + a_{N2}x_2 + ... + a_{NN-1}x_{N-1} + x_N = b_N
\end{cases} \tag{3}$$

с треугольной матрицей, у которой диагональные элементы равны единице.

Используя рассмотренный алгоритм умножения матрицы на вектор, можно вычислить вектор x_i, задаваемый соотношением (1), примерно за $\log_2 n + \log_2 r = \log_2 nr$ шагов при наличии порядка $n^2 r$ процессоров. Для вычисления всех x_i получим параллельный алгоритм высоты $s\log_2 nr$. Оказывается, вычислить векторы x_i, $1 \leqslant i \leqslant s$, можно за меньшее (примерно $\log_2 s \cdot \log_2 nr$) число шагов.

Запишем рекуррентные соотношения (1) в избыточном виде через матрицы и векторы высшего порядка:

$$\begin{pmatrix} x_i \\ x_{i-1} \\ \dots \\ x_{i-r+1} \\ 1 \end{pmatrix} = \begin{pmatrix} A_{i1} & \dots & A_{ir-1} & A_{ir} & b_i \\ E & \dots & 0 & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & E & 0 & 0 \\ 0 & \dots & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_{i-1} \\ x_{i-2} \\ \dots \\ x_{i-r} \\ 1 \end{pmatrix}, \ r \neq 1.$$

Обозначим матрицу через Q_i, вектор в левой части через y_i. Тогда

$$y_i = Q_i y_{i-1} = \ldots = Q_i Q_{i-1} \ldots Q_1 y_0, \quad 1 \leqslant i \leqslant s.$$

Смысл такой избыточной записи заключается в том, что все $y_1, \ldots, y_s$, а значит и все $x_1, \ldots, x_s$, можно вычислять одновременно. Если, например, $r = 1$, $s = 3$, то $Q_i = \begin{pmatrix} A_{i1} & b_i \\ 0 & 1 \end{pmatrix}$, $y_i = \begin{pmatrix} x_i \\ 1 \end{pmatrix}$,

$$y_1 = \begin{pmatrix} A_{11} & b_1 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x_0 \\ 1 \end{pmatrix},$$

$$y_2 = \begin{pmatrix} A_{21} & b_2 \\ 0 & 1 \end{pmatrix} y_1 = \begin{pmatrix} A_{21} & b_2 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} A_{11} & b_1 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x_0 \\ 1 \end{pmatrix},$$

$$y_3 = \begin{pmatrix} A_{31} & b_3 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} A_{21} & b_2 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} A_{11} & b_1 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x_0 \\ 1 \end{pmatrix}.$$

Видно, что y_1, y_2, y_3, т.е. $\begin{pmatrix} x_1 \\ 1 \end{pmatrix}$, $\begin{pmatrix} x_2 \\ 1 \end{pmatrix}$, $\begin{pmatrix} x_3 \\ 1 \end{pmatrix}$, можно вычислять одновременно.

Матрицы Q_i и векторы y_i имеют порядок $nr + 1$. Согласно алгоритму сдваивания, любое из произведений $Q_i Q_{i-1} \ldots Q_1 y_0$ можно вычислить за $\lceil \log_2(s+1) \rceil$ макроопераций умножения двух матриц порядка $nr + 1$. Все макрооперации во всех s произведениях можно вычислять одновременно. Используя рассмотренный параллельный алгоритм для умножения двух матриц, получим параллельный алгоритм для вычислить всех векторов $x_1, \ldots, x_s$, который имеет высоту порядка $\log_2 s \cdot \log_2 nr$ и ширину порядка $(nr)^3 s^2$. Этот алгоритм получил название процесс рекуррентного сдваивания,

С помощью процесса рекуррентного сдваивания можно решить систему линейных алгебраических уравнений с треугольной $N \times N$ матрицей примерно за $\log_2^2 N$ шагов, задействовав порядка N^5 процессоров. Действительно, за один параллельный шаг, используя около $N^2/2$ процессоров, можно сделать равными 1 все диагональные элементы матрицы, разделив их на соответствующие коэффициенты. Затем следует решить систему (3) с помощью соотношений (2); напомним, $n = 1$, $r = i$, $s = N$.

В случае метода простой итерации ($r = 1$) s итераций можно выполнить за $\log_2 s \cdot \log_2 n$ шагов на $n^3 s^2$ процессорах.

Пример 4 (вычисление обратной матрицы [1]). Пусть A – квадратная матрица порядка N. Можно вычислить A^{-1} за $O(\log_2^2 N)$ шагов на N^4 процессорах).

Получение алгоритмов минимальной высоты – задача не простая (примеры 1 и 2 – исключение). На сегодняшний день достижения в рамках концепции неограниченного параллелизма представляют набор достижений в области численных методов.

На практике алгоритмы небольшой высоты не нашли применения:

– они требуют чрезмерно большого числа процессоров,

– требуют очень много памяти,

– приводят к сложным коммуникационным связям между вычислительными узлами,

– процессоры загружены крайне слабо.

Единственным исключением являются алгоритмы сдваивания для многократного применения ассоциативных операций, например, сложения и умножения чисел, матриц.

Тем не менее, концепция неограниченного параллелизма очень полезна для знакомства с параллельными вычислениями, для лучшего понимания некоторых понятий и проблем параллельных вычислений.

Литература

1. Воеводин В. В., Воеводин Вл. В. Параллельные вычисления. – СПб.: БХВ-Петербург, 2002. – 608 с.

2. Воеводин В. В. Вычислительная математика и структура алгоритмов. – Москва: Изд-во МГУ, 2006. – 112 с. http://parallel.ru/info/parallel/voevodin/