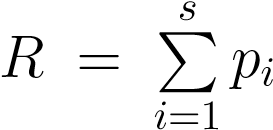
КОНЦЕПЦИЯ НЕОГРАНИЧЕННОГО ПАРАЛЛЕЛИЗМА

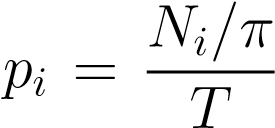
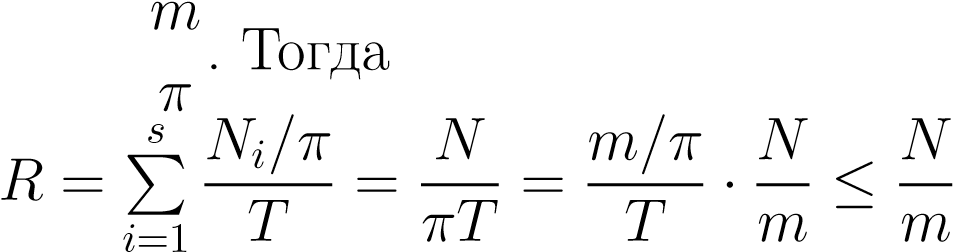
Концепция неограниченного параллелизма – это способ понимания, конструктивный принцип построения параллельных алгоритмов, в основе которого лежит предположение, что алгоритм реализуется на параллельной вычислительной системе, не накладывающей на него никаких ограничений. Считается, что процессоров[[1]](#footnote-1) 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 .

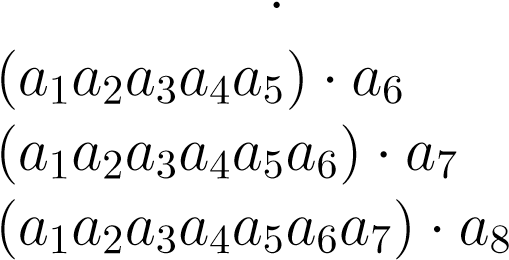
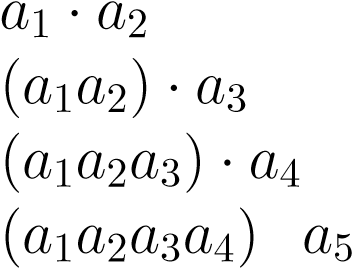
Доказательство. Воспользуемся формулой для выражения ускорения системы через загруженности процессоров: . Предположим, что

за время *T* реализации алгоритма *i*-й процессор выполнил *Ni* операций. По определению . Если процессоров достаточно, то операции одного яруса параллельной формы система может выполнить за время, равное или большее времени  выполнения одной операции; время *T* выполнения всех ярусов больше или равно

при любом числе процессоров.

Пример 1 (процесс сдваивания).

Рассмотрим параллельные формы двух алгоритмов вычисления произведения *a*1 · *a*2 · *...* · *aN.* Пусть *N* = 8*.*

1. Обычная схема.Ярус 1:

Ярус 2:

Ярус 3:

Ярус 4:

Ярус 5:

Ярус 6:

Ярус 7:

Изобразим граф алгоритма:

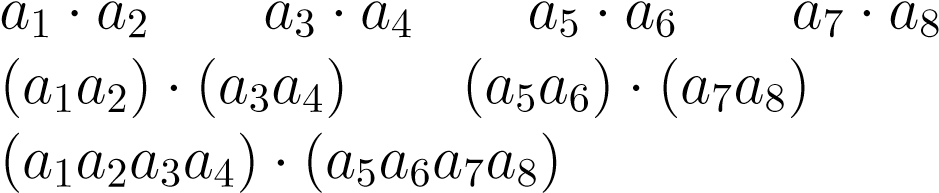
g -g -g -g -g -g -g

В общем случае высота алгоритма равна *N* − 1, ширина алгоритма равна 1*.*

1. Процесс сдваивания.Ярус 1:

Ярус 2:

Ярус 3:

 g@ g g@ g

@ @

@@R @@R g

gHHHHHHHHj g

В общем случае высота алгоритма равна ⌈log2 *N*⌉*,* ширина равна ⌈*N/*2⌉*.*

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*T p*2 операций. На нулевом ярусе находится не более *pT* операций. Так как *p* ≥ *N*, то *T* ≥ log*p N*.

Пример 2 (умножение матрицы на вектор; перемножение матриц). Рассмотрим задачу умножения матрицы *A* порядка *N* на *N*-мерный ∑*N*

вектор *b* : *ci* = *aijbj.* На первом шаге можно вычислить *N*2 произведений

*j*=1

*aijbj.* Далее, используя процесс сдваивания, за ⌈log2 *N*⌉ шагов можно вычислить *N* сумм, определяющих координаты вектора *c.* Высота алгоритма имеет порядок log2 *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 log2 *N* и шириной *N*3*.*

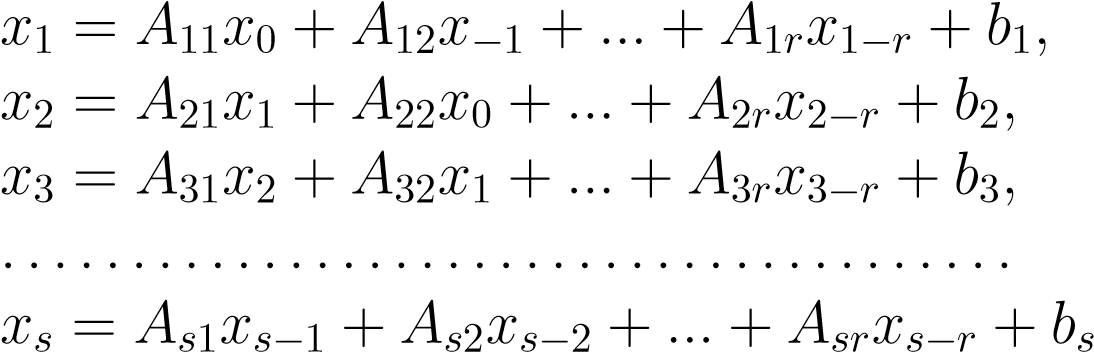
Пример 3 (процесс рекуррентного сдваивания; решение треугольной системы).

Пусть заданы матрицы *Aij*, 1 6 *i* 6 *s*, 1 6 *j* 6 *r*, векторы *b*1*,...,bs* и векторы *x*0, *x*−1, *... , x*−*r*+1 порядка *n*. Требуется вычислить векторы *xi,*

1 6 *i* 6 *s*, с помощью рекуррентных соотношений

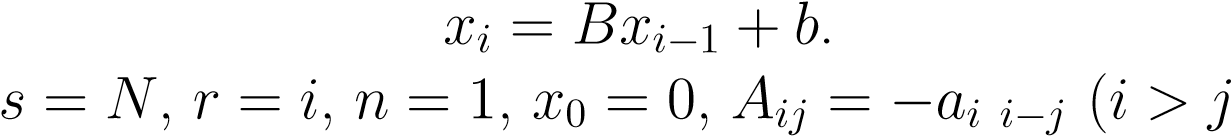
*xi* = *Ai*1*xi*−1 + *...* + *Airxi*−*r* + *bi,* (1)

или, в более подробной записи,

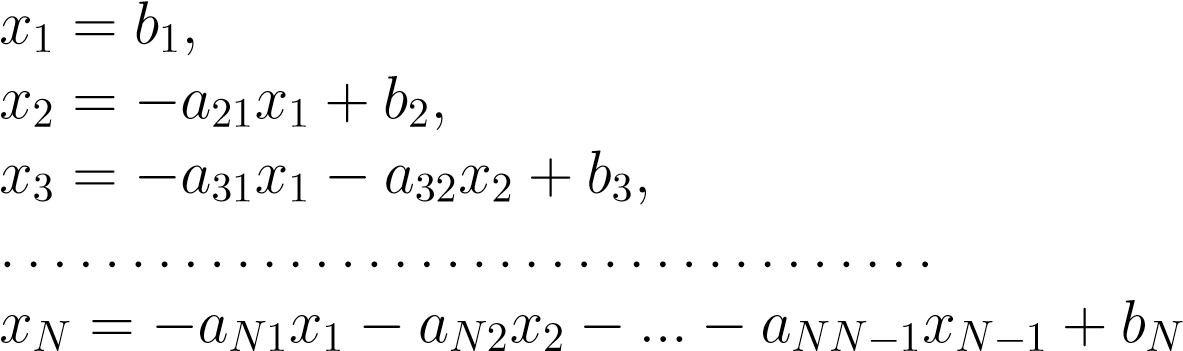
*.*

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

Пусть, например, *r* = 1, *Ai*1 = *B*, все векторы *b*1*,...,bs* равны. Получим метод простой итерации решения систем линейных алгебраических уравнений:

Пусть теперь , другие случаи

при *r* = *i*, *x*0 = 0 не рассматриваются). Получим соотношения

 (2)

|  |  |
| --- | --- |
| для решения системы линейных алгебраических уравнений |  |
| *x*1 = *b*1*,*   *a*2131*x*11 + *x*322 =2*b*2*,* 3 3    *a x* + *a x* + *x* = *b ,* | (3) |

 *aN*1*x*1 + *aN*2*x*2 + *...* + *aNN*−1*xN*−1 + *xN* = *bN ....................................*



с треугольной матрицей, у которой диагональные элементы равны единице.

Используя рассмотренный алгоритм умножения матрицы на вектор, можно вычислить вектор *xi*, задаваемый соотношением (1), примерно за log2 *n* + log2 *r* = log2 *nr* шагов при наличии порядка *n*2*r* процессоров. Для вычисления всех *xi* получим параллельный алгоритм высоты *s*log2 *nr.* Оказывается, вычислить векторы *xi,* 1 6 *i* 6 *s,* можно за меньшее (примерно log2 *s* · log2 *nr*) число шагов.

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

 *...i r*+1 = *... ... ... ... ... ... , r* = 1̸ *. x E ...* 0 0 0 *x*





 *xii* 1   *Ai*1 *... Air*−1 *Air bi*  *xiii*−1*r* 

− −2

*x* 0 *... E* 0 0 *x*

− −

1 0 *...* 0 0 1 1

Обозначим матрицу через *Qi*, вектор в левой части через *yi*. Тогда

*yi* = *Qiyi*−1 = *...* = *QiQi*−1 *...Q*1*y*0*,* 1 6 *i* 6 *s.*

Смысл такой избыточной записи заключается в том, что все *y*1*,...,ys*, а значит и все *x*1*,...,xs*, можно вычислять одновременно. Если, например,

(

*A b x*

*r* = 1, *s* = 3, то *Qi* =*i*1 *i* )*, yi* = ( *i* )*,*

0 1 1

*y*1 = ( 11 1 )( 0 )*,*

*A bx*

0 1 1

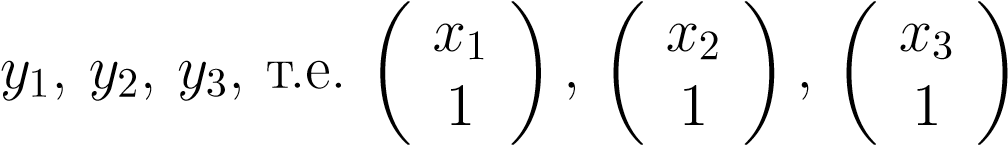
*y*2 = ( 21 2 )*y*1 = ( 21 2 )( 11 1 )( 0 )*,*

*A b A b A b x*

0 1 0 1 0 1 1 *y*3 = ( 31 3 )( 21 2 )( 11 1 )( 0 )*.*

*A b A b A b x*

0 1 0 1 0 1 1

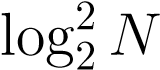
Видно, что *,* можно вычислять одновременно.

Матрицы *Qi* и векторы *yi* имеют порядок *nr* + 1. Согласно алгорит-

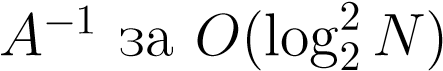
му сдваивания, любое из произведений *QiQi* 1 *...Q*1*y*0 можно вычислить −

за ⌈log2(*s* + 1)⌉ 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*,...,x*3 2 *s*, который имеет высоту порядка log2 *s* · log2 *nr* и ширину порядка

(*nr*) *s* . Этот алгоритм получил название процесс рекуррентного сдваивания,

С помощью процесса рекуррентного сдваивания можно решить систему линейных алгебраических уравнений с треугольной5 *N* ×*N* матрицей примерно за шагов, задействовав порядка *N* 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 log2 *s* · log2 *n* шагов на *n*3*s*2 процессорах.

Пример 4 (вычисление обратной матрицы [1]). Пусть *A* – квадратная матрица порядка *N*. Можно вычислить 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 вычислитель-ными 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/

1. Под процессором понимается одно вычислительное ядро. [↑](#footnote-ref-1)