УДК 681.3

Б. Я. Штейнберг

БЛОЧНО-РЕКУРСИВНОЕ ПАРАЛЛЕЛЬНОЕ ПЕРЕМНОЖЕНИЕ МАТРИЦ

Предложен новый параллельный алгоритм перемножения матриц с количеством пересылок, меньшим, чем у существующих. Для быстрого алгоритма перемножения матриц Штрассена рассмотрена сложность по обращениям к памяти и обоснована его неэффективность для матриц практически значимой размерности.

Ключевые слова: *параллельные алгоритмы, гиперкуб, размещение данных, межпроцессорные пересылки, сложность по обращениям к памяти.*

Введение. Новые тенденции в развитии вычислительных архитектур изменяют представления о сложности алгоритмов [1]. Если 40 лет назад время доступа к памяти было не существенно по сравнению со временем выполнения арифметических операций, то сейчас время доступа к памяти на порядок больше, а межпроцессорная пересылка требует существенно больше времени, чем доступ к памяти. Если раньше основной характеристикой сложности вычислительного алгоритма считалось количество умножений, то теперь такой характеристикой должно стать количество обращений к памяти, а для параллельных алгоритмов — количество межпроцессорных пересылок.

В настоящей работе предложен новый параллельный алгоритм перемножения матриц с количеством пересылок, меньшим, чем у существующих. Для быстрого алгоритма перемножения матриц Штрассена рассмотрена сложность по обращениям к памяти и обоснована нецелесообразность его использования для современных компьютеров; предложена процедура блочно-аффинного размещения данных. Следует отметить, что параллельные алгоритмы перемножения матриц на вычислительных системах с распределенной памятью, описанные в работах [2—5], предполагают размещение матриц полосами или блоками — это частные случаи блочно-аффинных размещений.

Приведенные в [2—5] алгоритмы требуют n^3 / p параллельных умножений. Объем пересылаемых данных каждого процессора для алгоритмов Фокса и Кэннона [4, 6] составляет $n^2 / p^{1/2}$. Объем пересылаемых данных каждого процессора в алгоритмах, предложенных в [3], равен $n^2 / p^{2/3}$. Объем пересылаемых данных каждого процессора в предлагаемом нами

блочно-рекурсивном алгоритме равен $\frac{n^2 \log_2 p}{p}$. При этом не используются рассылки данных типа „от одного — всем“. Предполагается использование полнодоступного коммутатора или коммуникационной сети „n-мерный куб“. Приводится схема отображения пересылок n-мерного куба на кольцевую сеть и кольцевой сети — на решетку, что позволит использовать данный алгоритм на многопроцессорных системах с соответствующей топологией.

Перемножение матриц при блочно-аффинных размещениях. Возможно размещать элементы массива в параллельной памяти произвольным образом, но необходимо, чтобы к этим элементам массива было удобно обращаться. Выбор способа размещения данных должен зависеть от исполняемой программы. Различные способы размещения данных вне связи с исполняемым кодом описаны в работе [7].

Будем предполагать, что имеется p модулей памяти. Пусть X — некоторый m-мерный массив.

Блочно-аффинные по модулю p размещения данных. Пусть натуральные числа $p, d_1, d_2, ..., d_m$ и целые константы $s_0, s_1, s_2, ..., s_m$ зависят только от m-мерного массива X. Блочно-аффинное по модулю p размещение m-мерного массива X — это такое размещение, при котором элемент $X(i_1, i_2, ..., i_m)$ находится в модуле памяти с номером

$$u = \left(\left\lfloor s_1 / d_1 \right\rfloor i_1 + \left\lfloor s_2 / d_2 \right\rfloor i_2 + \ ... + \left\lfloor s_m / d_m \right\rfloor i_m + s_0\right) \bmod p.$$

При описанном блочно-аффинном способе размещения m-мерный массив представляется как массив блоков размерностью $d_1, d_2, \ldots, d_m$, в котором у каждого блока все элементы находятся в одном модуле памяти.

Описанные далее способы размещения данных являются частными случаями блочно-аффинного по модулю p размещения.

Аффинные по модулю p размещения данных — такие, при которых элемент массива $X(i_1, i_2, ..., i_m)$ находится в модуле памяти с номером $u = (s_1 i_1 + s_2 i_2 + ... + s_m i_m + s_0) \bmod p$, где $s_0, s_1, s_2, ..., s_m$ — константы, зависящие только от массива X. Число s_0 будем называть сдвигом массива X. Это число соответствует номеру модуля памяти, в котором размещается нулевой элемент $X(0, 0, ..., 0)$.

{-1, 0, +1}-размещения — в них константы $s_1, s_2, ..., s_m$ принадлежат множеству {-1, 0, +1}, к ним, в частности, относятся размещение матрицы по строкам или по столбцам, или по диагоналям, или скошенная форма размещения (хранения) матрицы.

{0, +1}-размещения — в них константы $s_1, s_2, ..., s_m$ принадлежат множеству {0, +1}.

К такому типу размещений относятся размещения матриц по строкам, по столбцам и в скошенном виде. Размещение по диагоналям к такому типу не относится.

Покоординатные размещения со сдвигами — это такие {0, +1}-размещения, в которых лишь одна из величин $s_1, s_2, ..., s_m$ может быть равна единице.

Покоординатные размещения — это такие {0, +1}-размещения, в которых лишь одна из величин $s_1, s_2, ..., s_m$ может быть равна единице и $s_0 = 0$.

Размещения матрицы по строкам или по столбцам являются покоординатными.

Рассмотрим фрагмент программы, вычисляющий произведение двух матриц по стандартному алгоритму

```
do i = 1, n
do j = 1, n
do k = 1, n
X(i,j) = X(i,j)+A(i,k)*B(k,j).
```

Теорема 1. Не существует таких нетривиальных аффинных размещений массивов X, A и B в распределенной памяти, при которых параллельное произведение матриц выполняется без пересылок.

Доказательство. Предположим, что искомые аффинные размещения массивов существуют. Для каждого значения i, j и k элементы $X(i,j)$, $A(i,k)$ и $B(k,j)$ должны оказаться в одном и том же модуле памяти. Это возможно тогда и только тогда, когда существуют такие целые числа $s_1, s_2, t_1, t_2, u_1, u_2$, что выполняются равенства

$$s_1 i + s_2 j = t_1 i + t_2 k = u_1 k + u_2 j \mod p.$$

Это возможно для всех значений i, j, k лишь при выполнении равенств $s_1 = s_2 = t_1 = t_2 = u_1 = u_2 = 0 \mod p$, т.е. когда все данные находятся в одном и том же модуле памяти. Итак, задача вычисления произведения матриц не может быть распараллелена на вычислительной системе с распределенной памятью без межпроцессорных пересылок данных.

Блочно-рекурсивный параллельный алгоритм перемножения матриц. Будем считать, что количество процессорных элементов p вычислительной системы является степенью числа 4.

Рассмотрим задачу умножения квадратных матриц $X = AB$ размерностью n на вычислительной системе с n процессорными элементами. Матрицы A и B рассмотрим как блочные матрицы 2×2 с квадратными блоками размером $(n/2)\times(n/2)$. Всего в каждой матрице получается по 4 блока. Обозначим A_{ij}, $(i, j = 1,2)$, аналогично обозначим блоки матрицы B.

Модуль памяти 0 A_{11}, B_{11}, C_{11}	Модуль памяти 1 A_{12}, B_{21}, C_{12}
Модуль памяти 2 A_{21}, B_{12}, C_{21}	Модуль памяти 3 A_{22}, B_{22}, C_{22}

Рис. 1

Множество всех процессорных элементов разобьем на 4 равных группы с номерами 0, 1, 2, 3 (рис. 1). Разместим элементы обеих матриц по блокам — в каждой группе модулей памяти по одному.

— Блок A_{ij} $(i, j = 1,2)$ матрицы A разместим в группе модулей памяти с номером $(i-1)2 + j - 1$.

— Блок B_{ij} $(i, j = 1,2)$ матрицы B разместим в группе модулей памяти с номером $(i-1)2 + i - 1$.

Для вычисления произведения матриц в этом случае требуется два перемножения, две пересылки блоков (рис. 2, *а* — первое перемножение и первая группа пересылок, *б* — второе перемножение и вторая группа) и два сложения (рис. 3).

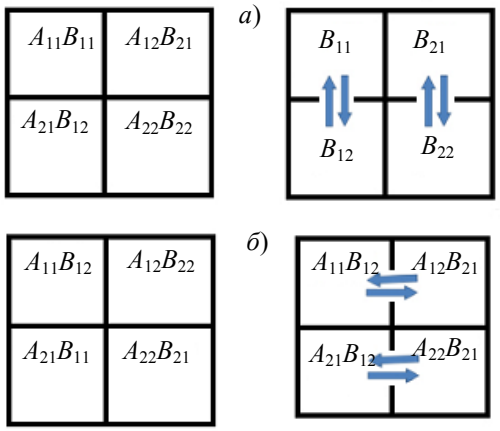

Рис. 2

Модуль памяти 0 $C_{11} = A_{11}B_{11} + A_{12}B_{21}$	Модуль памяти 1 $C_{12} = A_{12}B_{22} + A_{11}B_{12}$
Модуль памяти 2 $C_{21} = A_{21}B_{11} + A_{22}B_{21}$	Модуль памяти 3 $C_{22} = A_{21}B_{12} + A_{22}B_{22}$

Рис. 3

Перемножение блоков в каждой группе процессорных элементов (ПЭ) можно выполнять по такой же схеме рекурсивно. В этом алгоритме возможны $\log_4 p$ шагов. На каждом шаге размерность блоков в два раза уменьшается, но количество блоков в четыре раза возрастает.

На рис. 4, *а* представлен пример разбиения каждой из четырех групп ПЭ на 4 группы второго уровня; рис. 4, *б* — то же разбиение множества процессорных элементов на группы в двоичной кодировке. Нумерация соответствует нумерации узлов четырехмерного куба.

При двойной нумерации цифра перед точкой означает номер группы ПЭ первого уровня, цифра после запятой — номер группы ПЭ второго уровня (см. рис. 4, *а*).

а)

0,2	0,3	1,2	1,3
2,0	2,1	3,0	3,1
2,2	2,3	3,2	3,3
0,0	0,1	1,0	1,1

б)

0000	0001	0100	0101
0010	0011	0110	0111
1000	1001	1100	1101
1010	1011	1110	1111

Рис. 4

Количество параллельных шагов с умножениями (скалярными) в данном алгоритме, как и в других известных параллельных алгоритмах, равно n^3 / p. При наличии p процессорных элементов и при базовом последовательном алгоритме перемножения матриц со сложностью n^3 меньшего порядка сложности достичь невозможно.

Для пересылок данных в этом алгоритме удобно использовать полнодоступный коммутатор или ($\log_2 p$)-мерный куб. Всего в алгоритме получается $2\log_4 p = \log_2 p$ блочных пересылок.

На каждом шаге пересылается n^2 элементов по p штук одновременно (скалярных пересылочных шагов n^2 / p). Общее количество скалярных пересылочных шагов $n^2 \log_2 p / p$. Если $p = n$, то количество одновременных скалярных пересылок равно $n \log_2 n$ — это существенно меньше, чем при размещении матриц по строкам или столбцам, и меньше, чем пересылок в алгоритмах Кэннона и Фокса [4].

Перемножение прямоугольных матриц тоже может быть выполнено с помощью блочно-рекурсивного алгоритма. Действительно, пусть требуется перемножить матрицу A размером $n \times m$ на матрицу B размером $m \times k$. Обозначим $q = p^{1/2}$. Разобьем матрицу A на блоки размером $(n/q) \times (m/q)$, а матрицу B — на $(m/q) \times (k/q)$. Тогда задача перемножения прямоугольных матриц будет сведена к задаче перемножения квадратных блочных матриц размером $q \times q$, где $q = p^{1/2}$.

Отображение пересылок между сетями различной топологии. Известна задача переноса программ с одних многопроцессорных вычислительных систем на другие. При этом вычислительные системы могут различаться не только количеством вычислительных устройств, но и коммуникационной системой [8, 9]. Тогда возникает необходимость отображения топологии (моделирования пересылок, ориентированных на одну сеть, на другой сети). Например, в работе [6] рассмотрено моделирование на гиперкубе кольцевой сети с помощью кодов Грея.

Рассмотрим задачу моделирования n-мерного куба на кольце с количеством узлов $p = 2^n$. Напомним, что вершины n-мерного куба кодируются булевыми векторами $(x_1, x_2, \ldots, x_n)$ длиной n.

К сожалению, использовать коды Грея [6] для этой задачи не удобно. Действительно, для $n=3$ расстояние по кольцу между вершинами, соответствующими (0,0,0) и (1,0,0), равно 1, а между (0,0,1) и (1,0,1) — 4, хотя оба этих ребра n-куба соответствуют изменению первой координаты. Во многих задачах пересылки данных по таким ребрам предполагается выполнять одновременно. В данном примере на кольце эти пересылки будут выполняться в разное время.

Для случая $n=2$ задача тривиальна, поскольку двумерный куб представляет собой кольцо с 4 вершинами. Вершинам двумерного куба (0,0), (0,1), (1,1) (1,0) сопоставляются узлы кольцевой сети с номерами 0, 1, 2, 3 соответственно, т.е. $\varphi(0,0)=0$, $\varphi(0,1)=1$, $\varphi(1,1)=2$, $\varphi(1,0)=3$.

Определим сопоставление вершин n-мерного куба узлам кольцевой сети для $n>2$ по следующей формуле:

$$\varphi\left(x_1,x_2,\ldots,x_n\right)=\left(x_1+2x_2+\ldots+2^{n-3}x_{n-2}\right)+2^{n-2}\varphi\left(x_{n-1},x_n\right).$$

Первое слагаемое в данной формуле — это число, состоящее из первых $n-2$ цифр двоичного кода вершины n-мерного куба.

Рассмотрим подробнее указанное сопоставление узлов n-мерного куба узлам кольца. Сопоставление узлов n-мерного куба узлам кольца будем строить по индукции.

В предположении, что построено сопоставление узлов n-мерного куба узлам кольца для $n=k,k>1$, построим его для случая $n=k+1$. Пусть узлы кольца пронумерованы от 0 до $(n-1)$. Рассмотрим подмножество узлов кольца с четными номерами как кольцевую сеть (подкольцо), в которой время пересылки между соседними узлами в 2 раза дольше, чем в исходной кольцевой сети. Это подкольцо будем называть 0-подкольцо. Аналогичным образом узлы с нечетными номерами образуют подкольцо, которое будем называть 1-подкольцом.

Пусть $(x_1,x_2,\ldots,x_{k+1})$ — произвольная вершина $(k+1)$-мерного куба. Рассмотрим подмножество вершин n-мерного куба, состоящее из векторов $(0,x_2,\ldots,x_{k+1})$. Это подмножество порождает подграф, изоморфный k-мерному кубу. Узлы этого k-мерного куба сопоставим узлам 0-подкольца. Аналогично подмножество вершин $(1,x_2,\ldots,x_{k+1})$ также порождает подграф, изоморфный k-мерному кубу. Узлы этого k-мерного куба сопоставим узлам 1-подкольца.

Рассмотрим в n-мерном кубе множество ребер, соединяющих вершины вида $(x_1,\ldots,x_{i-1},0,x_{i+1},\ldots,x_{k+1})$ и $(x_1,\ldots,x_{i-1},1,x_{i+1},\ldots,x_{k+1})$. Ребра n-мерного куба, принадлежащие одному такому множеству, будем называть однотипными. Всего получается n типов ребер. При реализации принципа сдваивания на n-мерном кубе используются одновременные пересылки по однотипным ребрам. Поэтому при моделировании n-мерного куба на кольце важно, чтобы такие пересылки эффективно реализовывались.

Теорема 2. При описанном выше сопоставлении вершинам n-мерного куба узлов кольцевой сети расстояние между концевыми узлами всех однотипных ребер одинаково.

Доказательство может быть проведено методом математической индукции.

Следствие. Пересылки на кольце, соответствующие пересылкам по однотипным ребрам n-мерного куба, требуют одинакового времени.

В работе [6] рассмотрен способ представления на гиперкубе двумерной решетки, число узлов на сторонах которой является степенью двойки. Представим кольцо на решетке, что позволит на решетке моделировать пересылки n-мерного куба.

Если сеть имеет гамильтонов цикл (цикл, проходящий через каждую вершину строго один раз), то такой цикл и можно рассматривать как кольцевую сеть. Покажем, как построить гамильтонов цикл на прямоугольной решетке типа „тор“.

Допустим, что у прямоугольной решетки хотя бы одна из сторон содержит четное количество узлов. Пусть размерность этой решетки $m(2k)$. Тогда номера узлов, приведенные на рис. 5 (без окаймления), составляют гамильтонов цикл на этой решетке.

Пусть теперь прямоугольная решетка содержит нечетное количество узлов $(2m+1)\times(2k+1)$. В этом случае в решетке выделяется подрешетка размерностью $(2m)\times(2k)$, узлы которой нумеруются в указанной выше последовательности (см. рис. 5, с окаймлением).

На многомерном торе кольцо можно моделировать по правилу индукции, для двумерного тора отображение построено. Предположим, что оно существует для m-мерного тора и построим его для $(m+1)$-мерного тора. Пусть $W_1W_2\ldots W_{m+1}$ — $(m+1)$-мерный тор. Тогда, по предположению индукции, на m-мерном торе $W_2\ldots W_{m+1}$ можно смоделировать кольцо, которое обозначим C. Тогда исходный $(m+1)$-мерный тор можно представить в виде двумерного тора W_1C и на этом двумерном торе смоделировать кольцо.

У многокольцевых соединений [10], как правило, одно из колец является гамильтоновым циклом.

1	2*m*	(2*m*+1)			*m*(2*k*)	*m*(2*k*)+1
2	2*m*–1	(2*m*+2)			*m*(2*k*)–1	*m*(2*k*)+2
…						
(*m*–1)	(*m*+2)	3*m*–1	3*m*+2			
M	(*m*+1)	3*m*	3*m*+1	…	*m*(2*k*–1)	*m*(2*k*+1)
(*m*+1)(2*k*+1)	…	…	…	…	*m*(2*k*+1)+2	*m*(2*k*+1)+1

Рис. 5

О сложности обращений к памяти алгоритма Штрассена. Сложность стандартного алгоритма (в котором, согласно определению, строки одной матрицы скалярно умножаются на столбцы другой) равна n^3. В данном случае n^3 — это количество умножений. У алгоритма Штрассена [5, 11, 12] сложность равна $n^{2,81}$ (более точно, показатель степени равен $\log_2 7$). Этот специфический алгоритм дал импульс развитию теории сложности. Напомним, что сложность алгоритма — это количество операций как функция от количества входных данных, а сложность задачи — это сложность самого быстрого алгоритма. При этом под операциями подразумевались арифметические операции, как правило — умножения, т.е. требующие существенно большего времени, чем сложение. Позднее были найдены алгоритмы перемножения матриц еще менее сложные, чем алгоритм Штрассена. Существуют алгоритмы, представляющие собой комбинацию алгоритма Штрассена и стандартного алгоритма перемножения матриц [13]. Сложность задачи перемножения матриц пока не определена. Есть результаты, показывающие существование таких задач, для которых самого эффективного алгоритма не существует [14] — не исключено, что задача перемножения матриц является такой же.

Рассмотрим реальные преимущества алгоритма Штрассена по сравнению со стандартным. В 1988 г. на кафедре алгебры и дискретной математики механико-математического факультета Ростовского государственного университета (сейчас Южный федеральный университет) был программно реализован алгоритм Штрассена (студенческая работа, руководитель доц. Козак А.В.). Выяснилось, что алгоритм Штрассена начинает давать преимущества по сравнению со стандартным алгоритмом начиная с размерности матриц $n=32$. В 2002 г. опять был реализован алгоритм Штрассена (тоже студенческая работа, руководитель — автор настоящей работы). В этот раз преимущество алгоритма Штрассена начиналось с размерности матриц $n=512$. Попытаемся объяснить этот эффект. Примерно за 15 лет компьютеры стали совсем другими. Изменилось соотношение времени выполнения арифметических опе-

раций и времени обращения к памяти. А соотношение количества этих операций в алгоритме Штрассена и в стандартном алгоритме различно.

Умножение вещественных чисел современным компьютером может быть в 15 раз быстрее, чем считывание этих чисел из оперативной памяти. Попробуем оценить сложность обращений к памяти алгоритма Штрассена.

Предполагается, что размерность матрицы является степенью двойки: $n = 2^k$.

Исходные и результирующая матрицы представляются как блочные 2×2 матрицы с блоками размерностью $n/2$. Тогда произведение матриц имеет вид

$$\mathrm{C}=\mathrm{AB}$$

$$\mathrm{C} = \{C_{ij}\},\ \mathrm{A} = \{A_{ij}\},\ \mathrm{B} = \{B_{ij}\},\ i, j = 1, 2.$$

Здесь C_{ij}, A_{ij}, B_{ij} $(i, j = 1,\ 2)$ — блоки матриц C, A и B соответственно.

По алгоритму Штрассена вычисляются семь вспомогательных матриц (размером $n/2$):

$$\begin{aligned}
M_1 &= (A_{12} - A_{22})(B_{21} + B_{22}),\\
M_2 &= (A_{11} + A_{22})(B_{11} + B_{22}),\\
M_3 &= (A_{11} - A_{21})(B_{11} + B_{12}),\\
M_4 &= (A_{11} + A_{12})B_{22},\\
M_5 &= A_{11}(B_{12} - B_{22}),\\
M_6 &= A_{22}(B_{21} - B_{11}),\\
M_7 &= (A_{21} + A_{22})B_{11}.
\end{aligned} \tag{1}$$

После этого элементы результирующей матрицы вычисляются по формулам

$$\begin{aligned}
C_{11} &= M_1 + M_2 - M_4 + M_6,\\
C_{12} &= M_4 + M_5,\\
C_{21} &= M_6 + M_7,\\
C_{22} &= M_2 - M_3 + M_5 - M_7.
\end{aligned} \tag{2}$$

В формулах (1) перемножаются матрицы меньшей размерности, что тоже может выполняться рекурсивно по такому же алгоритму.

Обозначим через $F(n)$ количество операций обращения к памяти (чтение или запись) в алгоритме Штрассена.

В формулах (1), (2) матрицы размером $n/2$ встречаются 37 раз. При этом формулы (1) содержат умножения матриц размером $n/2$, каждое из которых потребует $F(n/2)$ обращений к памяти. Итого можно записать рекуррентную формулу

$$F(n) = 7F(n/2) + 37(n/2)^2.$$

Учитывая, что $F(1) = 3$, из рекуррентной формулы можно получить аналитическую

$$\begin{aligned}
F(n) &= 7F(n/2) + 37(n/2)^2 = 7\{7F(n/4) + 37(n/4)^2\} + 37(n/2)^2 =\\
&= 7^k F(1) + 37\{7^{k-1}(n/2^k)^2 + \ldots + 7(n/4)^2 + (n/2)^2\} =\\
&= 7^k \cdot 3 + 37/7n^2\{(7/4)^k + \ldots + (7/4)^2 + 7/4\} =\\
&= 7^k \cdot 3 + 37/7n^2\, 7/3((7/4)^k - 1) =\\
&= 7^k \cdot 3 + 37/3n^2((7/4)^k - 1).
\end{aligned}$$

Пренебрегая в скобках слагаемым (–1) и учитывая, что $k = \log_2 n$, получим, что количество обращений к памяти в алгоритме Штрассена равно

$$F(n) \approx 7^k 3 + 37/3n^2 \left(7/4\right)^k = 7^k 3 + 37/37^k = 7^k 46/3 \approx n^{2,81} 46/3.$$

Рассмотрим количество обращений к памяти в стандартном алгоритме. Каждый элемент результирующей матрицы вычисляется как скалярное произведение двух векторов (строки первой матрицы на столбец второй). Такое скалярное произведение вычисляется накоплением суммы произведений в регистре и требует n^2 чтений и одну запись в память. Итого $2n^3 + n^2 \approx 2n^3$ обращений к памяти.

Асимптотически количество обращений к памяти алгоритма Штрассена меньше, чем у стандартного алгоритма. Проследим, начиная с какой размерности матриц n в алгоритме Штрассена количество обращений к памяти будет меньше, чем в стандартном алгоритме — $2n^3 = n^{2,81} 46/3$, $n \approx 25\,000$.

Численный эксперимент для матриц такой размерности не представляется разумным, поскольку матрицы такой размерности могут быть размещены уже не в оперативной, а во внешней памяти.

Следует отметить, что рекурсивность алгоритма Штрассена тоже может вызвать дополнительные обращения к памяти (обращения к стеку, в котором хранятся параметры вызовов). Глубина рекурсивных вызовов равна $k = \log_2 n$. Если столько параметров вызовов не поместится в регистровой или кэш-памяти, то они будут помещаться в оперативной.

Итак, алгоритм Штрассена асимптотически эффективнее, но преимущество перед стандартным алгоритмом начинается с задач, размерность которых превышает практическую значимость.

СПИСОК ЛИТЕРАТУРЫ

1. *Пипер Ш., Пол Д., Скотт М.* Новая эра в оценке производительности компьютерных систем // Открытые системы. 2007. № 9. С. 52—58.
2. *Корнеев В. Д.* Параллельное программирование в MPI. М.—Ижевск: Институт компьютерных исследований, 2003. 304 с.
3. *Gupta H., Sadayappan P.* Communication Efficient Matrix-Multiplication on Hypercubes // Parallel Computing. 1996. N 22. P. 75—99.
4. *Гергель В. П.* Введение в методы параллельного программирования. Образовательный комплекс. Н. Новгород: ННГУ, 2005.
5. *Прангишвили И. В., Виленкин С. Я., Медведев И. Л.* Параллельные вычислительные системы с общим управлением. М.: Энергоатомиздат, 1983. 312 с.
6. *Гергель В. П., Стронгин Р. Г.* Основы параллельных вычислений для многопроцессорных вычислительных систем. Н. Новгород: ННГУ, 2003.
7. *Метлицкий Е. А., Каверзнев В. В.* Системы параллельной памяти. Теория, проектирование, применение. Л.: ЛГУ, 1989. 240 с.
8. *Treleaven P.C.* Parallel architecture overview // Parallel Computing. 1988. N 8. P. 59—70.
9. *Корнеев В. В.* Параллельные вычислительные системы. М.: Нолидж, 1999. 311 с.
10. *Подлазов В. С.* Свойства мультикольцевых и гиперкубовых коммутаторов на произвольных перестановках // РАСО'2001 Тр. Междунар. конф. „Параллельные вычисления и задачи управления". М.: ИПУ РАН, 2001. С. 152—164.
11. *Strassen V.* Gaussian Elimination is not Optimal // Numer. Math. 1969. Vol. 13, N 4. P. 354—356.
12. *Ахо А., Хопкрофт Дж., Ульман Дж.* Построение и анализ вычислительных алгоритмов. М.: Мир, 1979. 536 с.

13. *Елфимова Л. Д.* Быстрый клеточный метод умножения матриц // Кибернетика и системный анализ. 2008. № 3. С. 55—59.

14. *Разборов Л. А.* P=NP? — проблема: взгляд из 90-х // Математические события XX в. М.: ФАЗИС, 2003. С. 399—416.

Сведения об авторе

Борис Яковлевич Штейнберг — д-р техн. наук; Южный федеральный университет, Ростов-на-Дону; кафедра алгебры и дискретной математики; зав. кафедрой;
E-mail: borsteinb@mail.ru

Рекомендована институтом

Поступила в редакцию
10.03.09 г.