# МАТЕМАТИЧЕСКОЕ И ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ И КОМПЬЮТЕРНЫХ СЕТЕЙ

УДК 004.75

## ПАРАЛЛЕЛЬНОЕ УМНОЖЕНИЕ МАТРИЦ БОЛЬШИХ РАЗМЕРНОСТЕЙ НА МНОЖЕСТВЕ ЗАДАННЫХ ПРОЦЕССОРОВ

**В. М. Глушань,** д.т.н., профессор кафедры САПР ИКТИБ ЮФУ, Таганрог, Россия;
orcid.org/0000-0001-5822-9295, e-mail: gluval07@rambler.ru
**О. И. Красюк,** ООО «Оджетто», веб-разработчик, Таганрог, Россия;
orcid.org/0000-0003-3153-2651, e-mail: oleg.krasiuk@ictis.sfedu.ru
**А. Ю. Лозовой,** к.п.н., доцент кафедры ИЯ ИУЭС ЮФУ, Таганрог, Россия;
orcid.org/0000-0002-6701-3098, e-mail: lozovoy@sfedu.ru

*Умножение матриц находит применение во многих практических задачах. Умножение матриц малых размерностей не вызывает особых затруднений. Они возникают, когда приходится умножать матрицы с тысячами и миллионами строк и столбцов.* ***Цель статьи*** *состоит в исследовании валидности, разработанной авторами ранее распределенной подсистемы клиент-серверной архитектуры конструкторского проектирования СБИС для умножения матриц больших размерностей. Исследования этой подсистемы показали многократное ускорение времени проектирования СБИС, поэтому возникло естественное желание расширить ее функциональные возможности для решения других задач, требующих распараллеливания. Предложен метод разбиения исходных матриц на блоки для их умножения на заданном числе процессоров. Проведено имитационное моделирование процесса умножения матриц больших размерностей и сравнение полученных результатов с известными решениями. Результаты сравнения показали возможность подсистемы распараллеливать процесс умножения матриц. При этом достигаемое ускорение процесса умножения оказывается не хуже, чем в известных решениях, а в некоторых случаях оно оказывается даже и выше.*

***Ключевые слова:*** *разбиение матриц на блоки, блочное умножение, распараллеливание, клиент-серверная архитектура.*

**DOI:** 10.21667/1995-4565-2020-74-42-55

### Введение

Умножение матриц используется во многих практических задачах. При умножении матриц небольших размерностей особых трудностей не возникает. Они появляются, когда приходится умножать матрицы с тысячами и миллионами строк и столбцов. Приведем некоторые примеры, в которых невозможно обойтись без применения умножения матриц огромных размерностей. Так, например, в [1] отмечается, что для моделирования и оптимизации нестационарных течений газа в системах газоснабжения приходится использовать квадратные матрицы с числом строк и столбцов до $10^3$. При решении краевых задач, связанных с расчетами полей деформации и напряжений различных конструкций, используется численное моделирование и дискретизация области расчета [2]. В результате получаются СЛАУ еще более внушительных размерностей. Так как большинство расчетных областей конструкций имеет сложную геометрическую фигуру, то размерности СЛАУ могут достигать $10^4$-$10^5$. В работах [3, 4] решаются вопросы распараллеливания задач линейного программирования с десятками миллионов неизвестных и сотнями тысяч ограничений.

В данной статье рассматривается вопрос распараллеливания умножения большеразмерных матриц. Вопрос встал не как самоцель, а как побочная возможность использования для этого разработанной авторами ранее распределенной подсистемы конструкторского проектирования электронных схем. Поскольку исследования разработанной подсистемы показали многократное ускорение времени проектирования электронных схем, то возникло естественное желание адаптировать ее для решения других задач, требующих распараллеливания.

Распределенная подсистема архитектурно является многоуровневой иерархической и использует клиент-серверную технологию. Описание подсистемы представлено в опубликованных работах [5-12]. В первой из упомянутых работ детально исследовалась одноуровневая подсистема. В результате этих исследований было установлено, что получить значительный выигрыш от одноуровневой подсистемы практически невозможно. Поэтому дальнейшее направление исследований было направлено на многоуровневые архитектуры. В них с возрастанием числа уровней растет и общее число компьютеров, но при этом увеличивается и скорость работы подсистемы. Расширению функциональных возможностей иерархической подсистемы, направленных на распараллеливание процесса умножения матриц больших размерностей, посвящена данная статья.

**Умножение матриц**

Не все матрицы могут перемножаться. Для перемножения матрицы $A$ и матрицы $B$ должны удовлетворять определенному условию: число столбцов матрицы $A$ должно быть равно числу строк матрицы $B$. А это значит, что число элементов в строках матрицы $A$ должно быть равно числу элементов в столбцах матрицы $B$. Такие матрицы называются *согласованными* или *выполнимыми*. Кроме того, для матриц не выполняется свойство коммутативности. То есть в общем случае: $A \cdot B \neq B \cdot A$.

Приведенное условие выполнимости позволяет сформулировать простое правило получения размерности результирующей матрицы $C$. Размерность матриц, т.е. число строк и столбцов в них, будем обозначать двумя нижними индексами в их именах. Так матрица $A_{mn}$ состоит из $m$ строк и $n$ столбцов, а матрица $B_{nk}$ соответственно из $n$ строк и $k$ столбцов. Тогда очевидно, что эти матрицы соответствуют условию выполнимости, так как число столбцов у матрицы $A$ и число строк у матрицы $B$ одинаковое и равно $n$. Поэтому они могут перемножаться. Размерность результирующей матрицы $C$ удобно определять, записав последовательно буквенные обозначения числовых значений размерностей матриц $A$ и $B$: $m\ n\ n\ k$. В этой последовательности букв исключим две средние одинаковые буквы, а оставшиеся крайние буквы $m$ и $k$ будут соответствовать числу строк и столбцов результирующей матрицы $C$. Следует отметить, что число столбцов $k$ в матрице $C_{mk}$ может быть любым.

Теперь рассмотрим вопрос определения каждого элемента $c_{ij}$ матрицы $C$. Пусть заданы матрицы $A$ и $B$:

$$A = \begin{Vmatrix} a_{11} & a_{12} \ldots a_{1n} \\ a_{21} & a_{22} \ldots a_{2n} \\ . \ . \ . \ . \ . \\ a_{m1} & a_{m2} & a_{mn} \end{Vmatrix}, \qquad B = \begin{Vmatrix} b_{11} & b_{12} \ldots b_{1k} \\ b_{21} & b_{22} \ldots b_{2k} \\ . \ . \ . \ . \ . \\ b_{n1} & b_{n2} & b_{nk} \end{Vmatrix}.$$

Матрица $A$ состоит из $m$ строк и $n$ столбцов, а матрица $B$ из $n$ строк и $k$ столбцов. Поэтому матрица $C$ будет состоять из $m$ строк и $k$ столбцов

$$C = \begin{Vmatrix} c_{11} & c_{12} \cdot \cdot \cdot c_{1k} \\ c_{21} & c_{22} \cdot \cdot \cdot c_{2k} \\ \cdot \cdot \cdot \cdot \cdot \cdot \\ c_{m1} & c_{m2} \cdot \cdot \cdot c_{mk} \end{Vmatrix}.$$

В соответствующих разделах линейной алгебры [13] приводится следующая формула для вычисления любого элемента $c_{ij}$ результирующей матрицы $C$:

$$c_{ij} = a_{i1}b_{1j} + a_{i2}b_{2j} + \ldots + a_{in}b_{nj} = \sum_{q}^{n} a_{iq}b_{qj}, \quad (i = 1,2\ldots, \mathrm{m};\ j = 1, 2, \ldots, k) \tag{1}$$

В вербальном виде правило умножения матриц формулируется следующим образом: чтобы получить элемент матрицы $C$, стоящий в $i$-й строке и $j$-м столбце, нужно элементы $i$-й строки матрицы $A$ умножить на соответствующие элементы $j$-го столбца матрицы $B$ и полученные произведения сложить.

При умножении матриц больших размерностей для сокращения числа выполняемых операций (умножений и сложений) и соответственно времени решения задачи применяют так называемое блочное умножение матриц. При этом левую матрицу $A$ и правую матрицу $B$ (в силу того, что для матриц закон коммутативности не выполняется, будем говорить о левой и правой матрицах) разбивают соответствующим образом на блоки. В результате получают блочные строки и блочные столбцы. Каждая блочная строка включает некоторое число строк исходной матрицы, а каждый блочный столбец – некоторое число столбцов исходной матрицы.

Для умножаемых матриц в блочном представлении сохраняются те же правила, которым должны удовлетворять исходные матрицы, т.е. длина каждой блочной строки должна быть равна длине соответствующего блочного столбца. Иначе говоря, число блоков в блочной строке (т.е. число блок-столбцов) в матрице $A$ должно быть равно числу блоков в соответствующем блочном столбце (т.е. числу блок-строк) в матрице $B$, и, кроме того, число элементных столбцов в каждом перемножаемом блоке матрицы $A$ должно быть равно числу элементных строк в соответствующем блоке матрицы $B$. Под элементными строками и столбцами здесь понимаются клетки, в которых записываются числа в исходных матрицах, т.е. до их разбиения на блоки.

При соблюдении указанных условий все пары перемножаемых блоков будут выполнимыми, т.е. перемножаемыми. Результатом этого перемножения будет матрица $C$, число блочных строк которой будет равно числу блочных строк матрицы $A$, а число блочных столбцов будет равно числу блочных столбцов матрицы $B$. Размерность каждого блока в результирующей матрице $C$ будет определяться числом элементных строк перемножаемого блока в матрице $A$ и элементных столбцов соответствующего перемножаемого блока в матрице $B$. Очевидно, что все блоки результирующей матрицы формируются независимо. В совокупности они будут представлять ту же матрицу, которая была бы получена при перемножении исходных матриц без разбиения на блоки. А это означает, что процесс перемножения матриц может быть распараллелен на соответствующее число блоков, каждый из которых будет формироваться отдельным процессором. Время формирования результирующей матрицы будет определяться временем работы одного процессора (возможно, и нескольких, но с одинаковым временем работы), загруженного формированием самого большеразмерного блока результирующей матрицы.

Для более четкого представления возможных ситуаций рассмотрим два идентичных случая разбиения двух матриц, дающих разное число блоков в результирующей матрице. Исходная матрица $A$ (без разбиения) имеет размерность $(5\times5)$, а матрица $B$ – размерность $(5\times4)$. Для этих матриц условие выполнимости соблюдено, и результирующая матрица $C$ имеет размерность $(5\times4)$. Жирными линиями показано разбиение всех матриц на блоки. В кружках и эллипсах каждого блока указано соответствующей буквой имя каждого блока с

нижними индексами, указывающими номер блока в матрице. На рисунке 1 приведен первый вариант разбиения исходных матриц на блоки, а на рисунке 2 – второй вариант.

В первом варианте разбиения на блоки матрица $A$ (рисунок 1, *а*) имеет блочную размерность $(2\times3)$, т.е. 2 блочных строки и 3 блочных столбца, матрица $B$ (рисунок 1, *б*) имеет блочную размерность $(3\times2)$, т.е. 3 блочных строки и 2 блочных столбца. Видим, что в первом варианте блочного представления матрицы выполнимы, а результирующая матрица $C$ (рисунок 1, *в*) имеет размерность $(2\times2)$, т.е. 2 блочных строки и 2 блочных столбца.

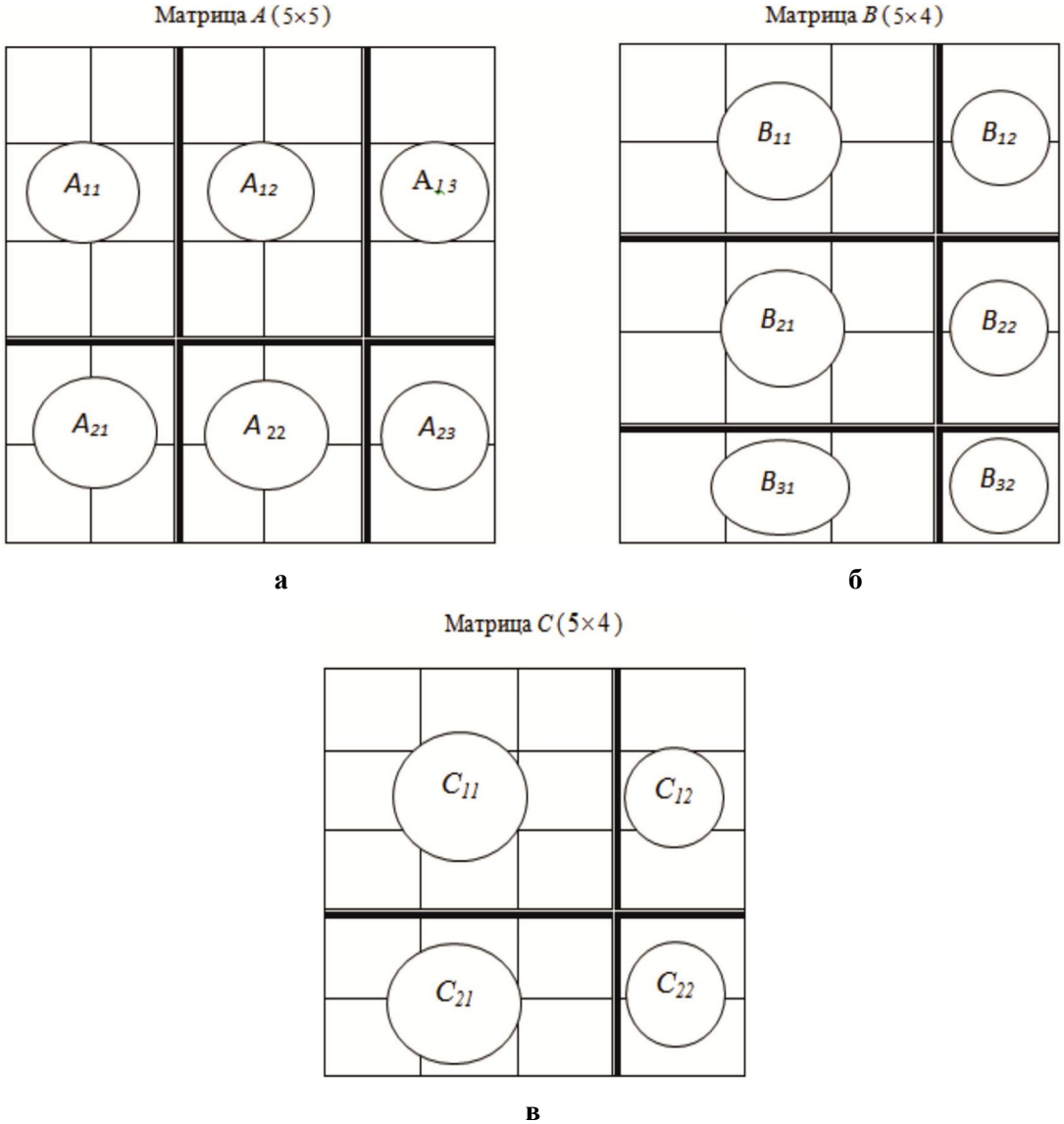


**Рисунок 1 – Первый вариант разбиения исходных матриц на блоки**
**Figure 1 – The first option for dividing the original matrices into blocks**

Во втором варианте разбиения на блоки матрица $A$ (рисунок 2, *а*) имеет размерность $(3\times2)$, т.е. 3 блочных строки и 2 блочных столбца, матрица $B$ (рисунок 2, *б*) имеет размерность $(2\times2)$, т.е. 2 блочных строки и 2 блочных столбца. Видим, что и во втором варианте блочного представления матрицы выполнимы, но результирующая матрица $C$ (рисунок 2, *в*) уже имеет размерность $(3\times2)$, т.е. 3 блочных строки и 2 блочных столбца.

Таким образом, в первом варианте разбиения в результирующей матрице получилось 4 блока, а во втором варианте – 6 блоков. Причем, если в первом варианте суммарное число блоков перемножаемых матриц было 12, а в результирующей матрице – 4, то во втором варианте суммарное число блоков было 10, а в результирующей матрице – 6. Отсюда следует, что по второму варианту процесс умножения данных матриц можно распараллелить на 6

процессоров, а в первом варианте только на 4. Поэтому есть все основания предполагать, что второй вариант может быть реализован за меньшее время, чем первый. Однако для более точного определения временных затрат необходимо знать и размерности блоков результирующей матрицы в первом и во втором варианте.

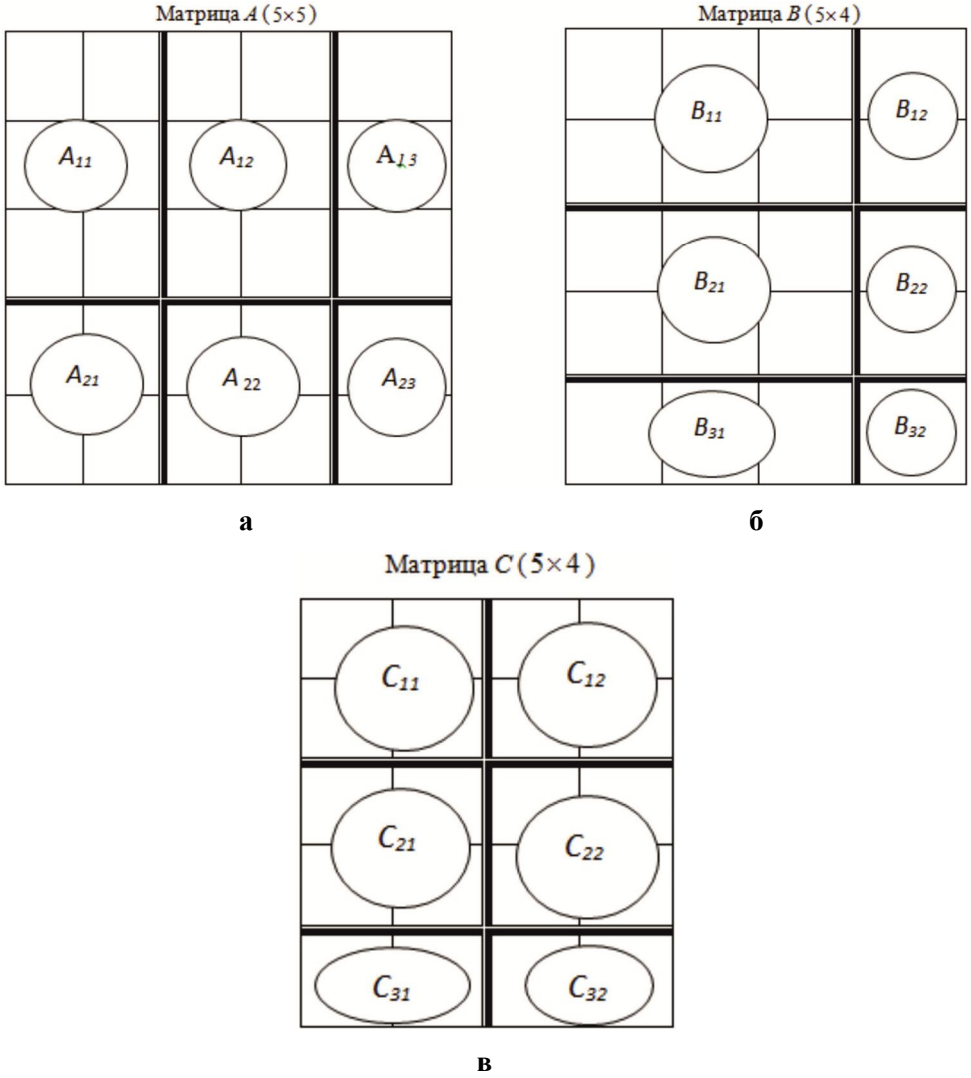


**Рисунок 2 – Второй вариант разбиения исходных матриц на блоки**
**Figure 2 – The second option to divide original matrices into blocks**

### Определение блочно-поэлементной размерности результирующей матрицы

Выше мы рассмотрели блочную размерность результирующих матриц. Но нужно иметь в виду, что эти матрицы должны совпадать с размерностью тех матриц, которые были бы получены без разбиения исходных матриц на блоки, т.е. иметь столько же элементных строк и столбцов и с таким же их поэлементным заполнением.

Для этого приведем анализ результирующих матриц *C*, приведенных на рисунках 1, *в* и 2, *в*. При этом будем опираться на формулу (1). В соответствии с этой формулой каждый блок результирующей матрицы получается путем поблочного перемножения каждого блока матрицы *A* и соответствующих блоков матрицы *B*, а в блоках – поэлементного перемножения каждой строки матрицы *A* и соответствующих элементов столбца матрицы *B*. Результаты всех указанных перемножений складываются. Но сложение возможно лишь при условии, что все размерности складываемых полученных после перемножения блоков были одинаковыми. В противном случае сложение будет невозможно.

Обозначение перемножаемых и результирующего блоков будем использовать таким, как приведено на рисунках, а в скобках будем указывать размерности блоков. Тогда получение всех блоков результирующей матрицы *C* для рисунка 1, *в* можно представить в виде совокупности следующих блочных операций:

$$\begin{cases} A_{11}(3\times2)\times B_{11}(2\times3)+A_{12}(3\times2)\times B_{21}(2\times3)+A_{13}(3\times1)\times B_{31}(1\times3)=C_{11}(3\times3);\\ A_{11}(3\times2)\times B_{12}(2\times1)+A_{12}(3\times2)\times B_{22}(2\times1)+A_{13}(3\times1)\times B_{32}(1\times1)=\ C_{12}(3\times1);\\ A_{21}(2\times2)\times B_{11}(2\times3)+A_{21}(2\times2)\times B_{21}(2\times3)+A_{23}(2\times1)\times B_{31}(1\times3)=C_{21}(2\times3);\\ A_{21}(2\times2)\times B_{12}(2\times1)+A_{22}(2\times2)\times B_{22}(2\times1)+A_{23}(2\times1)\times B_{32}(1\times1)=\ C_{22}(2\times1). \end{cases} \quad (2)$$

Из приведенных операций видим, что в каждой строке каждая пара перемножаемых блочных матриц дает матрицу одинаковой элементной размерности, поэтому их можно складывать. Так, в первой строке все перемножаемые пары дают элементную размерность $(3\times3)$ во второй строке $(3\times1)$ в третьей и четвертой строках соответственно $(2\times3)$ и $(2\times1)$ Указанные элементные размерности будут иметь соответственно блоки: $C_{11}$, $C_{12}$ $C_{21}$, $C_{22}$.

Аналогично (2) получение всех блоков результирующей матрицы *C* для рисунка 2*c* приведено ниже

$$\begin{cases} A_{11}(2\times2)\times B_{11}(2\times2)+A_{12}(2\times3)\times B_{21}(3\times2)=C_{11}(2\times2);\\ A_{11}(2\times2)\times B_{21}(2\times2)+A_{12}(2\times3)\times B_{22}(3\times2)=C_{12}(2\times2);\\ A_{21}(2\times2)\times B_{11}(2\times2)+A_{22}(2\times3)\times B_{21}(3\times2)=C_{21}(2\times2);\\ A_{21}(2\times2)\times B_{21}(2\times2)+A_{22}(2\times3)\times B_{22}(3\times2)=C_{22}(2\times2);\\ A_{31}(1\times2)\times B_{11}(2\times2)+A_{32}(1\times3)\times B_{21}(3\times2)\ =C_{31}(1\times2);\\ A_{31}(1\times2)\times B_{21}(2\times2)+A_{32}(1\times3)\times B_{22}(3\times2)\ =C_{32}(1\times2). \end{cases} \quad (3)$$

Длительность процесса умножения матриц зависит в основном от числа выполняемых умножений и сложений. Поэтому сравнение длительности определим, опираясь на рисунки 1, 2 и совокупности блочных операций (2) и (3). Очевидно, что при умножении блоков матриц $A_{ij}(n\times m)$ и $B_{ji}(m\times k)$ в соответствии с (1) потребуется выполнить

$$N=n\cdot m\cdot k \quad (4)$$

элементных умножений (напомним, что в данном случае умножаются числа в малых клетках) и все результаты умножений сложить. Здесь $n$ и $m$ – соответственно число строк и столбцов в блоке $A_{ij}$, а $m$ и $k$ – соответственно число строк и столбцов в блоке $B_{ji}$.Число таких сложений будет

$$M=n\cdot(m-1)\cdot k\,. \quad (5)$$

Кроме указанных сложений, как следует из совокупности операций (2) и (3), необходимо сложить результаты умножений всех блоков. А таких сложений в каждой строке (2) и (3) будет на единицу меньше, чем умножаемых блоков. Причем складываться будут матрицы одинаковых размерностей, и число таких сложений будет равно произведению числа строк и числа столбцов, т.е. числу элементов в каждой из складываемых матриц. В принятых в данной работе обозначениях число таких сложений будет

$$L=n\cdot k. \quad (6)$$

На рисунке 1 в строках матрицы *A* и в столбцах матрицы *B* по 3 блока. Поэтому в каждой строке совокупности операций (2) используется 2 сложения. А на рисунке 2 в строках матрицы *A* и в столбцах матрицы *B* по 2 блока. Поэтому в каждой строке совокупности операций (3) используется одно сложение.

Таким образом, можно подсчитать число операций для умножения заданных матриц с разбиением их на блоки по первому (рисунок 1) и второму случаю (рисунок 2). Поскольку процесс умножения выполняется параллельно, то время выполнения умножения будет определяться временем работы самого загруженного процессора. В первом случае это будет про-

цессор, в соответствии с (2) формирующий блок $C_{11}$ с наибольшей размерностью. Число операций при формировании этого блока обозначим: $O^1_{ум}(C_{11})$ – число всех умножений, $O^1_{сл}(C_{11})$ – число всех сложений.

Пользуясь размерностями блоков в первой строке совокупности блочных операций (2) и формулами (4), (5) и (6) для числа умножений и сложений соответственно получим:

$$O^1_{ум}(C_{11}) = 3\cdot 2\cdot 3 + 3\cdot 2\cdot 3 + 3\cdot 1\cdot 3 = 45;$$

$$O^1_{сл}(C_{11}) = 3\cdot 1\cdot 3 + 3\cdot 1\cdot 3 + \underline{2}\cdot 3\cdot 3 + 36.$$

В последнем слагаемом вычисления $O^1_{сл}(C_{11})$ подчеркнута цифра 2, отражая тот факт, что во всех строках совокупности блочных операций (2) используются два сложения (2 знака «+»).

Для второго варианта в соответствии с (3) четыре блока имеют одинаковую наибольшую размерность. Для расчетов можно взять любой из них, выберем блок $C_{11}$. Число операций при формировании этого блока обозначим: $O^2_{ум}$ – число всех умножений, $O^2_{сл}(C_{11})$ – число всех сложений. Пользуясь размерностями блоков в первой строке совокупности блочных операций (3) и формулами (4), (5) и (6) для числа умножений и сложений, соответственно получаем:

$$O^2_{ум}(C_{11}) = 2\cdot 2\cdot 2 + 2\cdot 3\cdot 2 = 20;$$

$$O^2_{ск}(C_{11}) = 2\cdot 2 = 4.$$

Из приведенных подсчетов видим, что во втором варианте число умножений более чем в 2 раза меньше, чем в первом варианте, а число сложений меньше в 9 раз. Этого и следовало ожидать, поскольку во втором варианте использовалось больше процессоров и каждый из них менее загружен.

В приведенных подсчетах не учитывались операции заключительного формирования (сборки) всей матрицы из отдельных блочных матриц $C_{ij}$. Но эта операция (ее можно назвать копированием) в обоих вариантах разбиения будет занимать одинаковое время, поэтому ее вклад в общее число операций можно не учитывать.

**Основополагающие принципы разбиения умножаемых матриц на блоки**

1. Число блок-столбцов у матрицы *A* должно быть равно числу блок-строк матрицы *B*.

2. Число блок-строк *M* и число блок-столбцов *N* в матрицах *A* и *B* зависит от заданного числа процессоров, представляемых в форме квадрата или прямоугольника со сторонами *M* и *N*.

3. Число элементов в каждой элементной строке любого, кроме, возможно, последнего блок-столбца, во всех блок-строках матрицы *A* равно числу элементов в каждом элементном столбце во всех блок-строках, кроме, возможно, последней блок-строки матрицы *B*.

4. Число блок-строк матрицы *C* равно числу блок-строк матрицы *A*, а число блок-столбцов в матрице *C* равно числу *N* блок-столбцов матрицы *B*. Поэтому на основании пунктов 2 и 3 матрица *B* в блочной форме всегда представляет квадрат.

5. Число элементных строк во всех блок-строках матрицы *C* совпадает с числом элементных строк во всех соответствующих блок-строках матрицы *A*, а число элементных столбцов во всех столбцах матрицы *C* совпадает с числом элементных столбцов во всех соответствующих блок-столбцах матрицы *B*.

Формализация приведенных принципов позволяет построить алгоритм блочно-элементного разбиения матриц.

**Алгоритм разбиения матриц на блоки**

1. Заданы матрицы $A(m\times n)$, $B(n\times k)$ и $Q$ – число процессоров, представленных: 1) в форме квадрата $Q = M\times N$, $M = N = \sqrt{Q}$; 2) в форме прямоугольника $Q = M\times N$, $M \neq N$. Процессоры также имеют форму прямоугольника с числом строк *M* и числом столбцов *N*. Соответственно этому матрица *A* разбивается на *M* блок-строк и *N* блок-столбцов.

2. Определение числа элементных строк в блок-строках матрицы $A(m\times n)$ – случай 2.

2.1. Число элементных строк в блок-строках $(1,2,\ldots,M-1)$ матрицы $A$ определяется как целая часть $m_ц$ числа $\lfloor m/M \rfloor + r_1$, т.е. $m_ц = \lfloor m/M \rfloor$, $r_1$ – остаток от деления, $r_1 = m - m_ц \cdot M$.

2.2. Если $r_1 \neq 0$, то число элементных строк $m_M$ в последней блок-строке $M$ определяется выражением $m_M = m_ц + r_1 = m_ц + m - m_ц \cdot M = m + m_ц(1-M)$. Если $r_1 = 0$, то $m_M = m_ц$.

3. Определение числа элементных столбцов в блок-столбцах матрицы $A(m\times n)$.

3.1. Число элементных столбцов в блок-столбцах $(1,2,\ldots,N-1)$ матрицы $A$ определяется как целая часть $n_ц$ числа $\lfloor n/N \rfloor + r_2$, т.е. $n_ц = \lfloor n/N \rfloor$, $r_2$ – остаток от деления, $r_2 = n - n_ц \cdot N$.

3.2. Если $r_2 \neq 0$, то число элементных столбцов $n_N$ в последнем блок-столбце $N$ определяется выражением $n_N = n_ц + r_2 = n_ц + n - n_ц \cdot N = N + n_ц(1-N)$. Если $r_2 = 0$, то $n_N = n_ц$.

4. Построение блочно-элементной формы матрицы $A(m\times n)$. На основании вычислений в пп. 2°.1, 2°.2, 3°.1 и 3°.2 она принимает следующий вид:

$$\begin{array}{c|cccc} & 1 & \cdots & N-1 & N \\ 1 & m_ц\times n_ц & \cdots & m_ц\times n_ц & m_ц\times n_K \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ M-1 & m_ц\times n_ц & \cdots & m_ц\times n_ц & m_ц\times n_K \\ M & m_M\times n_ц & \cdots & m_M\times n_ц & m_M\times n_K \end{array}$$

5. Построение блочно-элементной формы матрицы $B(n\times k)$. В этой матрице число блок-строк совпадает с числом блок-столбцов в матрице $A(m\times n)$, т.е. равно $N$, а число блок-столбцов в матрице $B$ по условию определяется числом заданных процессоров в строке и также равно $N$.

5.1. Число элементных строк в каждой из $(N-1)$ блок-строке равно $n_ц$ (эти значения копируются из блок-элементной формы матрицы $A$).

5.2. Число элементных столбцов $k_ц$ в каждом из ($N$-1) блок-столбцов определяется как целая часть $k_ц$ числа $\lfloor k/N \rfloor + r_3$, а число элементных столбцов в последнем блок-столбце $N$, если $r_3 \neq 0$, определяется так: $k_N = k_ц + r_3 = k_ц + k - k_ц \cdot N = k + k_ц(1-N)$. Если $r_3 = 0$, то $k_N = k_ц$.

6. При использовании вычислений, приведенных в п. п. 5°.1 и 5°.2, блочно-элементная форма матрицы $B(n\times k)$ принимает вид

$$\begin{array}{c|cccc} & 1 & \cdots & N-1 & N \\ 1 & n_ц\times k_ц & \cdots & n_ц\times k_ц & n_ц\times k_N \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ N-1 & n_ц\times k_ц & \cdots & n_ц\times k_ц & n_ц\times k_N \\ N & n_N\times k_ц & \cdots & n_N\times k_ц & n_N\times k_N \end{array}$$

7. Построение блочно-элементной формы матрицы $C(m\times n)$ путем копирования элементов $m_ц$ и $m_M$ из блочно-элементной формы матрицы $A$, а $k_ц$ и $k_N$ из блочно-элементной формы матрицы $B$

$$\begin{array}{c|cccc} & 1 & \cdots & N-1 & N \\ 1 & m_ц\times k_ц & \cdots & m_ц\times k_ц & m_ц\times k_N \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ M-1 & m_ц\times k_ц & \cdots & m_ц\times k_ц & m_ц\times k_N \\ M & m_M\times k_ц & \cdots & m_M\times k_ц & m_M\times k_N \end{array}$$

Если элементы в последней строке и в последнем столбце блочно-элементной формы матрицы $C$ получены, когда $r_1 = r_2 = r_3 = 0,$ то все элементы блочно-элементной формы матрицы $C$ будут одинаковыми. Поэтому время обработки любого элемента матрицы $C$ будет определять время умножения исходных матриц $A$ и $B$. Если же элементы или в строке, или в столбце, или и в строке и в столбце блочно-элементной формы матрицы $C$ получены, когда $r_1 \neq 0,\ r_2 \neq 0,\ r_3 \neq 0,$ то элемент матрицы $C$, расположенный в ее последней строке и последнем столбце, будет наибольшим. Он и будет определять время перемножения исходных матриц.

**Оценка сложности алгоритма параллельного умножения матриц**

Классическая оценка времени умножения квадратных матриц алгоритмом Matrix_Multiply на одном процессоре равна $O(n^3)$. Известный алгоритм Штрассена имеет рекордную оценку $O(n^{2,81}),$ где $n$ – число элементных строк и столбцов в умножаемых матрицах [14].

Для случая параллельного умножения квадратных матриц время работы будет определяться числом клеток $l$ в строке или в столбце одного блока и числом пар перемножаемых и поэлементно складываемых блоков матриц $A$ и $B$. Число $l$ зависит от числа процессоров $q^2$ и определяется по формуле $l = n/\sqrt{q}.$ Число пар перемножаемых поэлементно и складываемых блоков $l = \sqrt{q}.$ Тогда оценка параллельного времени умножения будет определяться как $O(\sqrt{q}l^3) = O(\sqrt{q}(n/\sqrt{q})^3).$ Например, для $n = 12,\ q = 9,\ O(\sqrt{9}(12/\sqrt{9})^3) = 192.$ Умножение матриц на одном процессоре с таким же числом $n$ строк и столбцов дает оценку времени умножения $O(12^3) = 1728,$ откуда следует, что процесс блочного параллельного умножения на 9 процессорах происходит в 9 раз быстрее. Для случая $n = 12,\ q = 16,\ O(\sqrt{16}(12/\sqrt{16})^3) = 108,$ а ускорение будет равно 16. Иначе говоря, величина ускорения равна числу параллельно работающих процессоров.

Для случая прямоугольных матриц временная сложность без разбиения на блоки будет равна $O(m \cdot n \cdot k)$. Учитывая, что для получения каждого элемента (клетки) требуется одно и то же число операций, временная сложность при разбиении матриц на блоки будет определяться числом клеток в блоке, расположенном в $M$-й строке и $N$-м столбце матрицы $C$. Это число, обозначим его $D$, будет равно

$$D = m_M \times k_N = (m + m_ц(1 - M)) \times (k + k_ц(1 - K)).$$

Тогда оценка временной сложности будет $O(D)$. Для нашего случая матрицы $C(m \times n) = C(32 \times 24)$

$$D = (32 + 5(1 - 6)) \times (24 + 6(1 - 4)) = 42.$$

Однако это значение является приближенным и заниженным, так как не учитывает все операции, которые были использованы при построении блока $C_{MN}$. Здесь не учтено число элементных столбцов $n_N$ в последнем блок-столбце $N$. Это число определяет количество операций, которое необходимо выполнить для получения результата только в одной клетке блока $C_{MN}$. Для получения всего блока $C_{MN}$ необходимо выполнить попарное перемножение соответствующих блоков последней блок-строки матрицы $A$ и блоков блок-столбцов матрицы $B$ и сложить их. А таких парных блочных операций будет $n$. Поэтому точное значение $D_\text{Т}$ временной сложности получим, домножив $D$ на $n_N$ и $n$, и тогда оно будет иметь вид: $D_T = D \cdot n_N \cdot n$. В нашем случае $n = n_N = 4,$ поэтому точная оценка временной сложности будет равна 672 операции.

Определим ускорение, которое получим при параллельном блочном умножении рассматриваемых матриц на 24-х процессорах, относительно безблочного однопроцессорного умножения. При безблочном умножении потребуется выполнить $32 \times 16 \times 24 = 12288$ операций. Отсюда ускорение составит величину 18,3 раза.

Рассмотрим пример (рисунок 3) разбиения исходных матриц достаточно большой размерности. Пусть заданы матрицы с элементными размерностями $A(32\times16)$ и $B(16\times24)$, которые нужно разбить на блоки для их перемножения на 24-х компьютерах. В соответствии с пунктами 1 и 2 приведенных принципов должна быть получена результирующая матрица с элементной размерностью $C(32\times24)$. Для получения такой матрицы исходную матрицу $A(32\times16)$ нужно разбить на 6 блочных строк и 4 блочных столбца. А матрица $B(16\times24)$ из условия выполнимости должна быть разбита на 4 блочных строки и, в соответствии с прямоугольной формой заданных процессоров $(6\times4)$, на 4 блочных столбца. В соответствии с п. 4° приведенного алгоритма строим блочно-элементную форму матрицы $A(32\times16)$, которая в наглядно-графическом виде приведена на рисунке 3, *а*. В соответствии с пунктами 5° и 6° строим блочно-элементную форму матрицы $B(16\times24)$, наглядно-графическая форма которой приведена на рисунке 3, *б*. В соответствии с пунктом 7° строим блочно-элементную форму матрицы $C(32\times24)$ наглядно-графическая форма которой приведена на рисунке 3, *в*.

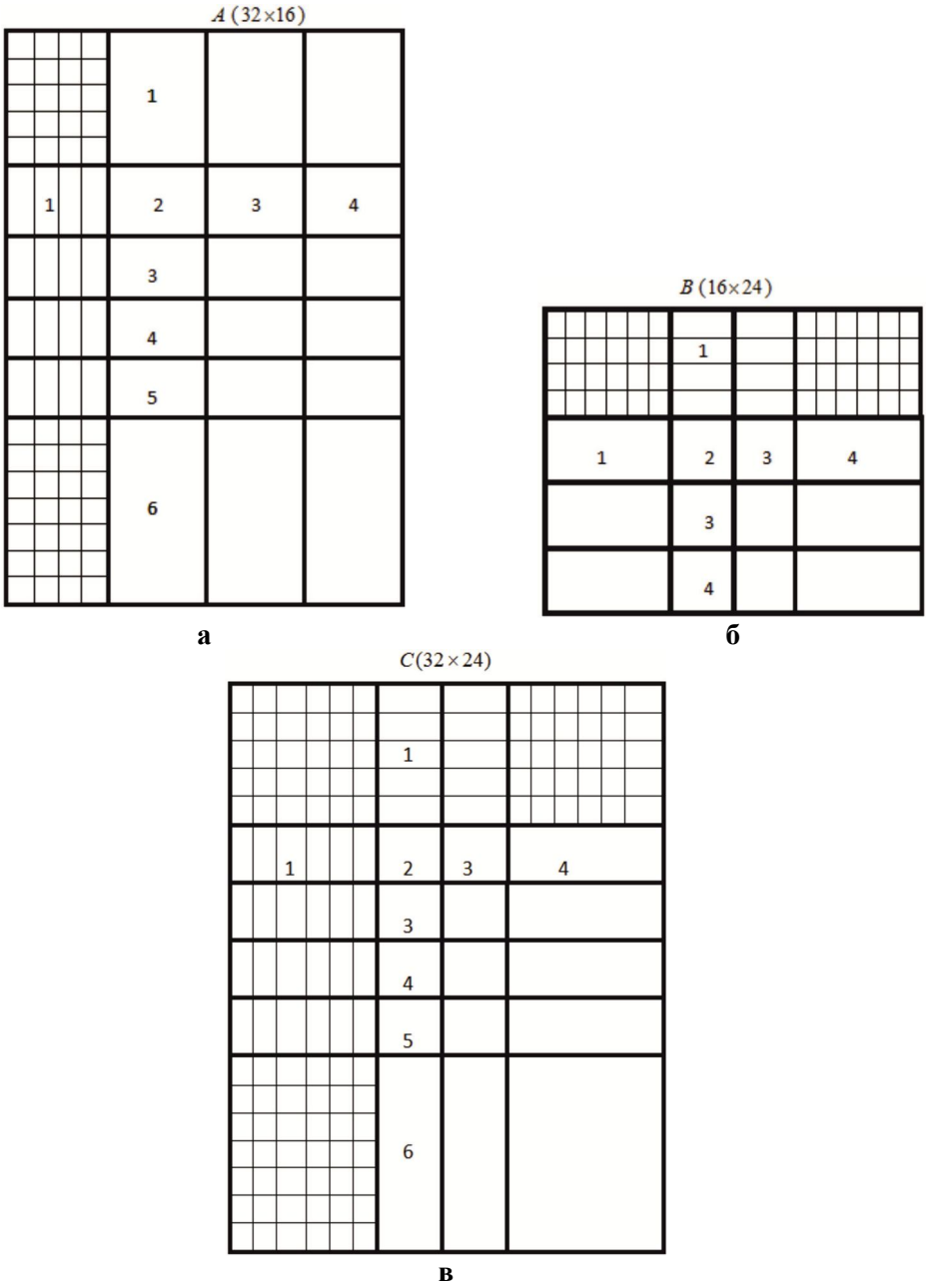


**Рисунок 3 – Представление матриц большой размерности на элементном и блочном уровнях**
**Figure 3 – Representation of large matrices dimensions at element and block levels**

**Имитационное моделирование умножения матриц больших размерностей**

В приведенной таблице представлены результаты имитационного моделирования процессов умножения матриц больших размерностей. Цель имитации состояла в том, чтобы подтвердить или опровергнуть возможность использования иерархической клиент-серверной архитектуры, целенаправленно разработанной для автоматизированного проектирования микроэлектронных схем, для блочного умножения матриц больших размерностей. Для объективного сравнения использованы квадратные матрицы тех же размерностей, которые были использованы в работе [15].

Приведем некоторые пояснения к таблице. В графе «Вид разбиения» указаны варианты разбиения использованных матриц на блоки. Во второй графе «Число процессоров» указано число процессоров, участвующих в умножении при соответствующем разбиении исходных матриц. В графах для каждой размерности матриц указано число шагов (суммарное число операций – умножений, сложений и копирования), а в скобках в каждой клетке под числом шагов приведено ускорение процесса умножения относительно случая без блочного умножения. В каждом «Виде разбиения» указаны три цифры, которые означают следующее: первая цифра означает на сколько блоков левая матрица разбита по вертикали, вторая цифра – на сколько блоков она разбита по горизонтали, и эта же цифра указывает на сколько блоков правая матрица разбита по вертикали. Третья цифра указывает на сколько блоков разбита правая матрица по горизонтали. Например, второе разбиение 2;3;2 означает, что каждая правая матрица соответствующей размерности разбивалась на 2 блока по вертикали и на 3 блока по горизонтали, а левая матрица разбивалась на 3 блока по вертикали и 2 блока по горизонтали.

**Результаты имитации процесса умножения матриц**
**Results of simulation of the process of matrix multiplication**

| Вид разбиения | Число процессоров | | Размерность матрицы | | | | |
|---|---|---|---|---|---|---|---|
| | | | 1000 | 1500 | 2000 | 2500 | 3000 |
| 1. Нет | 1 | Число шагов | $1999 \cdot 10^6$ (1) | $67478 \cdot 10^5$ (1) | $15996 \cdot 10^6$ (1) | $31244 \cdot 10^6$ (1) | $53991 \cdot 10^6$ (1) |
| 2. 2;2;2 | 4 | | $503 \cdot 10^6$ (3,97) | $16942 \cdot 10^5$ (3,98) | $4012 \cdot 10^6$ (3,98) | $7831 \cdot 10^6$ (3,98) | $13527 \cdot 10^6$ (3,99) |
| 3. 2;3;2 | 4 | | $503 \cdot 10^6$ (3,97) | $16942 \cdot 10^5$ (3,98) | $4012 \cdot 10^6$ (3,98) | $7831 \cdot 10^6$ (3,98) | $13527 \cdot 10^6$ (3,99) |
| 4. 3;3;3 | 9 | | $2261 \cdot 10^5$ (8,84) | $7567 \cdot 10^5$ (8,92) | $1797 \cdot 10^6$ (8,91) | $3497 \cdot 10^6$ (8,94) | $6027 \cdot 10^6$ (8,96) |
| 5. 4;4;4 | 16 | | $1280 \cdot 10^5$ (15,62) | $4286 \cdot 10^5$ (15,74) | $1012 \cdot 10^6$ (15,81) | $1972 \cdot 10^6$ (15,84) | $3402 \cdot 10^6$ (15,87) |

Для наглядности табличные результаты имитационного моделирования представлены графически на рисунке 4.

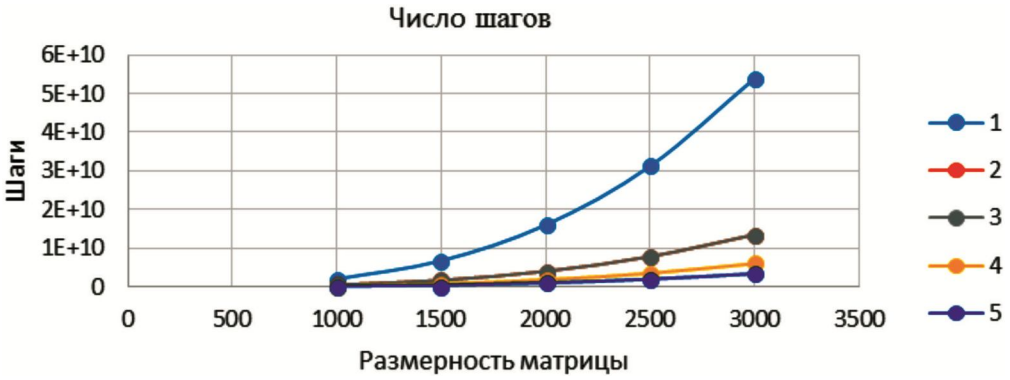


**Рисунок 4 – Графические результаты имитационного моделирования**
**Figure 4 – Graphical results of simulation modeling**

На приведенном рисунке 4 отображены только 4 графика, так как графики для разбиений 2 и 3 полностью совпадают. Сравнивая полученные результаты по критерию ускорения при 4-х параллельно работающих процессорах с результатами, приведенными в [15], видим близкое его значение для умножения матриц размерности $2000\times2000$: в нашем случае оно равно 3,98 и 3,58 в аналоге. Для размерностей матриц $3000\times3000$ в нашем случае ускорение равно 3,99 и 3,64 в аналоге. В проведенных нами исследованиях ускорение несколько больше, чем в аналоге. Но в принципе различие незначительное, и оно может говорить лишь о том, что в нашем исследовании возможно, не полностью были учтены накладные расходы. На незначительном расхождении в результатах мог сказаться и способ разбиения матриц на блоки. Об этом говорит и тот факт, что в при увеличении числа параллельно работающих процессоров ускорение в обоих исследованиях увеличивается пропорционально их числу. Так, в аналоге при 6 процессорах для матриц размерностей $2000\times2000$ и $3000\times3000$ получено ускорение соответственно 5,35 и 5,36, а в нашем случае при 9 и 16 процессорах на тех же размерностях матриц получено ускорение соответственно 8,96 и 15,87.

## Заключение

Изложенные в данной статье исследования дают основание заключить, что распределенные иерархические клиент-серверные архитектуры, предназначенные изначально для проектирования электронных схем, представляют хорошие перспективы их применения для значительного ускорения процесса умножения матриц больших размерностей. Однако следует заметить, что в статье приведены результаты имитационного моделирования одноуровневой иерархической архитектуры. В то же время многоуровневые иерархические архитектуры в статье не рассматривались. В обычных дихотомических иерархических архитектурах число процессоров экспоненциально возрастает как $2^n$ с возрастанием числа уровней $n$. Но в работе [12] предложена так называемая «жадная» архитектура, в которой число процессоров сводится до числа, которое было бы на последнем уровне обычной иерархической архитектуры. Возможности такой архитектуры для умножения матриц больших размерностей и разработка практического программного продукта для реализации распределенной иерархической подсистемы представляют перспективное направление будущих исследований.

*Работа выполнена при поддержке гранта РФФИ (проект № 18-01-00041).*

## Библиографический список

1. Решение разреженных СЛАУ больших размерностей средствами ManagedCuda в.NET. https://habr.com/ru/post/260993/. Дата посещения 15.04.20.

2. **Каледин В. О.** Численно-аналитические модели в прочностных расчетах пространственных конструкций [Текст]/НФИ Кем. ГУ. Новокузнецк, 2000. 204 с.

3. **Голиков А. И., Евтушенко Ю. Г.** Метод решения задач линейного программирования большой размерности. Докл. Академии наук, 2004. Т. 397, № 6. С. 727-732

4. **Гаранжа В. А.** Параллельная реализация метода Ньютона для решения больших задач линейного программирования / В. А. Гаранжа, А. И. Голиков, Ю. Г. Евтушенко // Журн. вычисл. мат. и мат. физики. 2009. Т. 49, № 8. С. 1369-1384.

5. **Глушань В. М., Лаврик П. В.** Распределенные САПР. Архитектура и возможности: монография. Старый Оскол: ТНТ, 2014. 188 с.

6. **Глушань В. М., Лаврик П. В.** Возможности распределенной подсистемы топологического проектирования, построенной на основе клиент-серверных технологий. Проблемы разработки перспективных микро-и нано электронных систем: сб. трудов / под общ. ред. акад. РАН А. Л. Стемпковского. М.: ИППМ РАН, 2014, Ч. II. С. 11-14.

7. **Глушань В. М., Рыбальченко М. В., Лаврик П. В.** О возможностях иерархической клиент-серверной архитектуры для топологического проектирования СБИС. Труды Конгресса по интеллектуальным системам и информационным технологиям «IS&IT'15». Научное издание в 3-х томах. – Таганрог. Изд-во ЮФУ 2015. Т. 1. С. 50-56.

8. **Glushan V. M., Lavrik P. V., Rybalchenko M. V.** Hypergraph model of hierarchical client-server architecture for distributed compyting. 9th International Conference Application of Information and communication technologies – AICT2015. October 2015, Rostov-on-Don, pp. 454-457.

9. **Glushan V. M., Lavrik P. V., Lozovoy A. Yu., Rybalchenko M. V.** Distributed mode of topological design of VLSI by means of hierarchical client-server architecture. Atlantis Press, 2016. Amsterdam-Hong-Kong-Paris. Information Technologies in Science, Management, social Sphere and Medicine (ITSMSSM 2016), pp. 532-536.

10. **Глушань В. М., Дубровский И. А., Красюк О. И., Рыбальченко М. В.** Модификации распределенной иерархической клиент-серверной архитектуры подсистемы конструкторского проектирования СБИС. Труды Конгресса по интеллектуальным системам и информационным технологиям «IS&IT'17». Научное издание в 3-х томах. – Таганрог: Изд-во Ступина С.А. 2017, Т. 1, с. 101-107.

11. **Глушань В. М., Дубровский И. А., Красюк О. И., Рыбальченко М. В.** Анализ трудоемкости конструкторского проектирования СБИС средствами гибридной иерархической клиент-серверной подсистемы // Вестник Рязанского государственного радиотехнического университета. 2017. № 62. С. 71-78.

12. **Глушань В. М., Дубровский И. А., Красюк О. И.** Практическая реализация проекта иерархической клиент-серверной подсистемы конструкторского проектирования СБИС // Вестник Рязанского государственного радиотехнического университета. 2018. № 66-1. С. 50-58.

13. **Иванова С. А.** Линейная алгебра: учеб. пособие / С.А. Иванова, В.А. Павский. 2-е изд., перераб. и доп. Кемерово: Кемеровский государственный университет, 2019. 125 с.

14. **Кормен Т., Лейзерсон Ч., Риверст Р., Штайн К.** Алгоритмы: построение и анализ. 2-е изд.: пер. с англ. М.: Издательский дом «Вильямс», 2009. 1296 с.

15. **Гергель В. П.** Введение в методы параллельного программирования. https://ppt-online.org/686059. Дата посещения 23.05.20.

UDC 004.75

# PARALLEL MULTIPLICATION OF LARGE-DIMENSIONAL MATRICES ON MANY GIVEN PROCESSORS

**V. M. Glushan,** Dr. Sc. (Tech.), professor of the CAD Department of IKTIB SFU, Taganrog, Russia; orcid.org/0000-0001-5822-9295, e-mail:gluval07@rambler.ru
**O. I. Krasyuk,** Ogetto LLC, web developer, Taganrog, Russia; orcid.org/0000-0003-3153-2651, e-mail: oleg.krasiuk@ictis.sfedu.ru
**A. Yu. Lozovoy,** Ph.D. (Ped.), associate Professor of the Department of I&A IUPET SFU, Taganrog, Russia; orcid.org/0000-0002-6701-3098, e-mail: lozovoy@sfedu.ru

*Matrix multiplication is applied to many practical problems. Small dimension matrix multi-plication Matrix multiplication is applied to many practical problems. Small dimension matrix multiplication causes no particular difficulties. They arise when you have to multiply matrices with thousands and millions of rows and columns.* ***The object of the article*** *is to study the validity of the previously developed by the authors distributed subsystem of client-server architecture for VLSI design engineering for the multiplication of high-dimensional matrices. Studies of this subsystem have shown multiple acceleration of VLSI design time, so there was a natural desire to expand its functionality to solve other problems requiring parallelization. The article proposes a method to divide initial matrices into blocks for multiplying them on a given number of processors. Simulation modeling of the process of multiplying matrices with large dimensions and comparison of the results obtained with known solutions are carried out. The comparison showed the ability of the subsystem to parallelize the process of matrix multiplication. In this case, the achieved acceleration of multiplication process turns out to be no worse than in the known solutions, and in some cases it turns out to be even higher.*

***Key words****: partitioning of matrices into blocks, block multiplication, parallelization, client-server architecture.*

**DOI:** 10.21667/1995-4565-2020-74-42-55

**References**

1. Reshenie razrezhennyh SLAU bol'shih razmernostej sredstvami ManagedCuda v.NET. https:// habr.com/ru/post/260993/. Data poseshhenija 15.04.20.

2. **Kaledin V. O.** *Chislenno-analiticheskie modeli v prochnostnyh raschetah prostranstvennyh konstrukcij* [Tekst]/NFI Kem. GU. Novokuzneck, 2000. 204 p. (in Russian).

3. **Golikov A. I., Evtushenko Ju. G.** Metod reshenija zadach linejnogo programmirovanija bol'-shoj razmernosti. *Dokl. Akademii nauk*, 2004, vol. 397, no. 6, pp. 727-732. (in Russian).

4. **Garanzha V. A.** Parallel'naja realizacija metoda N'jutona dlja reshenija bol'shih zadach li-nejnogo programmirovanija / V. A. Garanzha, A. I. Golikov, Ju. G. Evtushenko. *Zhurn. vychisl. mat. i mat. fiziki*. 2009, vol. 49, no. 8, pp. 1369-1384. (in Russian).

5. **Glushan' V. M., Lavrik P. V.** *Raspredelennye SAPR. Arhitektura i vozmozhnosti.* Monografija. Staryj Oskol: TNT, 2014. 188 p. (in Russian).

6. **Glushan' V. M., Lavrik P. V.** Vozmozhnosti raspredelennoj podsistemy topologicheskogo proektirovanija, postroennoj na osnove klient-servernyh tehnologij. Problemy razrabotki perspektivnyh mikro-i nano jelektronnyh sistem. 2014. *Sbornik trudov/Pod obshhej red. Akademika RAN A.L. Stempkovskogo*. Moscow: IPPM RAN, 2014, ch. II, pp. 11-14 (in Russian).

7. **Glushan' V. M., Rybal'chenko M. V., Lavrik P. V.** O vozmozhnostjah ierarhicheskoj klient-servernoj arhitektury dlja topologicheskogo proektirovanija SBIS. *Trudy Kongressa po in-tellektual'nym sistemam i informacionnym tehnologijam «IS&IT'15». Nauchnoe izdanie v 3-h tomah*. Taganrog. Izd-vo JuFU. 2015, vol. 1, pp. 50-56. (in Russian).

8. **Glushan V. M., Lavrik P. V., Rybalchenko M. V.** Hypergraph model of hierarchical client-server architecture for distributed compyting. *9th International Conference Application of Information and communication technologies – AICT2015*. October 2015, Rostov-on-Don, pp.454-457. (in Russian).

9. **Glushan V. M., Lavrik P. V., Lozovoy A. Yu., Rybalchenko M. V.** Distributed mode of topological design of VLSI by means of hierarchical client-server architecture. Atlantis Press, 2016. Amsterdam-Hong-Kong-Paris. *Information Technologies in Science, Management, social Sphere and Medicine* (ITSMSSM 2016), pp. 532-536.

10. **Glushan' V. M., Dubrovskij I. A., Krasjuk O. I., Rybal'chenko M. V.** Modifikacii raspredelennoj ierarhicheskoj klient-servernoj arhitektury podsistemy konstruktorskogo proek-tirovanija SBIS. *Trudy Kongressa po intellektual'nym sistemam i informacionnym tehnologijam «IS&IT'17»*. Nauchnoe izdanie v 3-h tomah. Taganrog: Izd-vo Stupina S.A. 2017, vol. 1, pp. 101-107. (in Russian).

11. **Glushan' V. M., Dubrovskij I. A., Krasjuk O. I., Rybal'chenko M. V.** Analiz trudoemkosti konstruktorskogo proektirovanija SBIS sredstvami gibridnoj ierarhicheskoj klient-servernoj podsistemy. *Vestnik Rjazanskogo gosudarstvennogo radiotehnicheskogo universiteta.* 2017, no. 62, pp. 71-78 (in Russian).

12. **Glushan' V. M., Dubrovskij I. A., Krasjuk O. I.** Prakticheskaja realizacija proekta ierar-hicheskoj klient-servernoj podsistemy konstruktorskogo proektirovanija SBIS. *Vestnik Rjazanskogo gosudarstvennogo radiotehnicheskogo universiteta*. 2018, no. 66-1, pp. 50-58 (in Russian).

13. **Ivanova S. A.** *Linejnaja algebra: uchebnoe posobie* / S.A. Ivanova, V.A. Pavskij; Kemerovskij gosudarstvennyj universitet. 2-e izd., pererab. i dop. Kemerovo: Kemerovskij gosudarstvennyj universitet. 2019, 125 p. (in Russian).

14. **Kormen T., Leyzerson Ch., Riverst P., Shtayn K. Algorinmy: postroenie i analiz.** 2-e izd.: per. s angl. M.: Izdatel'skij dom «Vil'jms», 2009. 1296 p. (in Russian).

15. **Gergel' V. P.** Vvedenie v metody parallel'nogo programmirovanija. https://ppt-online.org/686059. Data poseshhenija 23.05.20.