Параллельные блочные алгоритмы умножения матриц для мультикомпьютеров с распределенной памятью

Фельдман Л.П., Назарова И.А., Хорошилов А.В.

Кафедра ПМиИ, ДонНТУ
feldman@r5.dgtu.donetsk.ua, nazarova@r5.dgtu.donetsk.ua

Abstract

Feldman L.P., Nazarova I.A., Horoshilov A.V. Parallel block algorithms of matrix multiplication for computer systems with distributed memories. *Parallel block algorithms for matrix multiplication are considerated. The potential system and algorithm parallelism is exploited. Obtained algorithms are realized on parallel structures with mesh topology. The estimations of the execution time, acceleration and efficiency parallel solution are defined.*

Введение

Матричное умножение (МУ) – это базовая операция линейной алгебры и доминирующая вычислительная часть многих научных приложений, таких как решение СЛАУ, интегрирование систем дифференциальных уравнений как обыкновенных, так и в частных производных. Существует множество традиционных параллельных алгоритмов вычисления матричного произведения для плотнозаполненных матриц [1-3]. Практически все алгоритмы или их варианты имеют приблизительно линейное ускорение для больших размерностей матриц, и не существует алгоритма, который был бы существенно лучше других. В этой статье анализируется эффективность и масштабируемость традиционных алгоритмов блочного матричного умножения и предлагается параллельный алгоритм быстрого матричного умножения с использованием модификации рекурсивного алгоритма Штрассена для решения систем линейных обыкновенных дифференциальных уравнений [4].

1. Разработка и анализ эффективности параллельного алгоритма на базе систолического матричного умножения

Систолический алгоритм умножения матриц является наиболее эффективным для параллельных компьютеров SIMD архитектуры [3]. Рассмотрим модификацию этого алгоритма с целью использования его на кластерных и MIMD системах. В качестве модели высокопроизводительной параллельной вычислительной системы будем

использовать мультикомпьютер с распределенной памятью и топологией 2D-тор.

Вычислительная схема алгоритма основана на блочном разбиении матриц, и использует как преимущества последовательного умножения матриц, так и систолического алгоритма. Этап предварительной подготовки состоит из разбиения перемножаемых матриц на блоки и отображения блоков на виртуальную топологию тор. Далее над блоками выполняются в точности те же действия, которые выполняются систолическим алгоритмом над элементами матриц, причем сложение и умножение матричных блоков выполняются последовательно одним процессором.

Пусть исходные матрицы A,B имеют размерность $m \times m$, и разбиваются на квадратные блоки порядка $k = \lceil m / p \rceil$. Для простоты рассуждений пусть $m \vdots p$ и $m \vdots k$. Тогда, $m = q \cdot k$, где $k -$ это размер ширины блока, а $q^2 -$ количество блоков.

Операцию умножения матриц A и B в блочном виде можно представить следующим образом:

$$\begin{pmatrix} A_{11} & A_{12} & ... & A_{1q} \\ ... & ... & ... & ... \\ A_{q1} & A_{q2} & ... & A_{qq} \end{pmatrix} \times \begin{pmatrix} B_{11} & B_{12} & ... & B_{q1} \\ ... & ... & ... & ... \\ B_{q1} & B_{q2} & ... & B_{qq} \end{pmatrix} = \begin{pmatrix} C_{11} & C_{12} & ... & C_{q1} \\ ... & ... & ... & ... \\ C_{q1} & C_{q2} & ... & C_{qq} \end{pmatrix},$$

где каждый блок C_{ij} матрицы C определяется в соответствии с выражением: $C_{ij} = \sum_{l=1}^{q} A_{il} \cdot B_{lj}$. На первом этапе алгоритма блочного систолического умножения матриц каждый процессор решетки с номером $< i, j >$ содержит два блока матриц исходных данных: A_{ij} и B_{ij}. В процессе выполнения алгоритма процессор P_{ij} отвечает за вычисление блока матрицы результата C_{ij}.

Далее блоки матрицы A косо сдвигаются влево по строкам:

$$\leftarrow_i A = \leftarrow_i \begin{bmatrix} A_{11} & A_{12} & ... & A_{1q} \\ A_{21} & A_{22} & ... & A_{2q} \\ ... & ... & ... & ... \\ A_{q1} & A_{q2} & ... & A_{qq} \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} & ... & A_{1q} \\ A_{22} & A_{23} & ... & A_{21} \\ ... & ... & ... & ... \\ A_{qq} & A_{q1} & ... & A_{qq-1} \end{bmatrix},$$

а блоки матрицы B косо вверх по столбцам:

$$B\uparrow^{j}=\begin{bmatrix} B_{11} & B_{12} & \dots & B_{1q} \\ B_{21} & B_{22} & \dots & B_{2q} \\ \dots & \dots & \dots & \dots \\ B_{q1} & B_{q2} & \dots & B_{qq} \end{bmatrix}\uparrow^{j}=\begin{bmatrix} B_{11} & B_{22} & \dots & B_{qq} \\ B_{21} & B_{32} & \dots & B_{1q} \\ \dots & \dots & \dots & \dots \\ B_{q1} & B_{12} & \dots & B_{q-1q} \end{bmatrix}.$$

Затем, над блоками выполняются в точности те же действия, которые выполняются систолическим алгоритмом над элементами матриц, причем сложение и умножение матричных блоков выполняются последовательно одним процессором. То есть на каждом из $q=p$ шагов выполняется умножение блоков матриц A и B на каждом процессоре по простому последовательному стандартному алгоритму. Затем производится одиночный сдвиг влево для блоков матрицы A и вверх для блоков матрицы B, и описанные действия повторяются. В результате на каждом процессоре с номером $<i,j>$ получается блок матрицы результата C_{ij}. Вычислительная схема описанного алгоритма для мультикомпьютера из p^2 процессоров приведена на рисунке 1.

Пусть время выполнения одной операции умножения/сложения чисел составляет t_{mul}/t_{ad}. Поскольку для большинства RISC архитектур справедливо соотношение: $t_{ad}=t_{mul}=t_{op}$, вводится понятие флопа, как произвольной операции с плавающей точкой, и далее все временные характеристики приводятся к флопу.

Время реализации арифметических операций для блочного систолического алгоритма составляет:

$$T_{p,comp}^{BSys}=q\cdot(T_{A_{ij}\times B_{ij}}+T_{A_{ij}+B_{ij}}),$$

где $T_{A_{ij}\times B_{ij}}/T_{A_{ij}+B_{ij}}$ – время, необходимое для реализации операций умножения/сложения блоков матриц размерности: $k=(m/p)$.

В свою очередь, время вычисления произведения блоков матриц равно:

$$T_{A_{ij}\times B_{ij}}=k^3(t_{mul}+t_{ad})=\frac{2m^3}{p^3}t_{op},$$

здесь $k^2=m^2/p^2$ – количество элементов блока, $k(t_{mul}+t_{ad})$ – время, необходимое для вычисления одного элемента блока.

Время на реализацию сложения блоков матриц составляет:

$$T_{A_{ij}+B_{ij}}=k^2\cdot t_{ad}=\frac{m^2}{p^2}\cdot t_{op}.$$

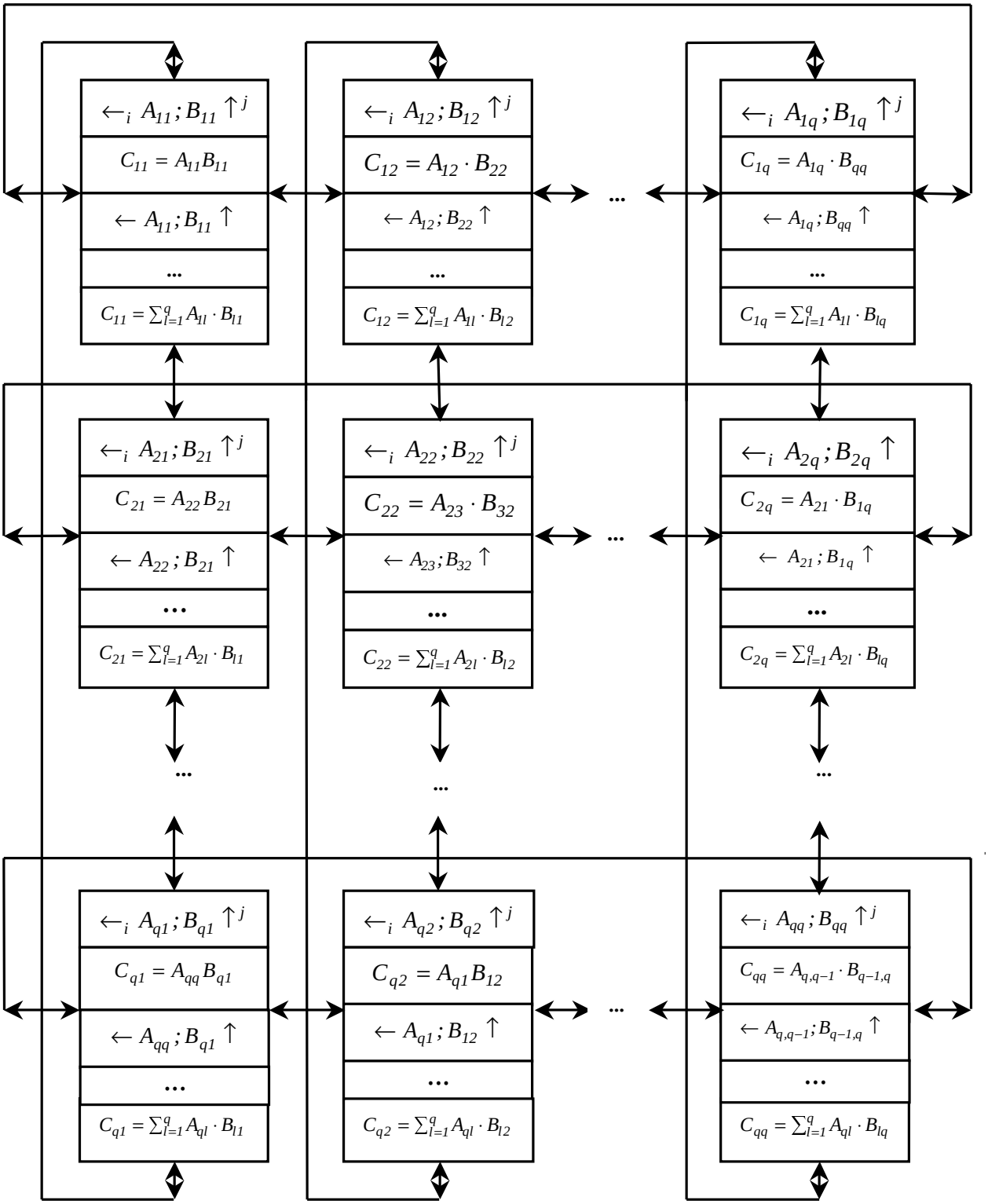

Рисунок 1 – Вычислительная схема блочного систолического умножения квадратных матриц для мультикомпьютера топологии 2D-тор

Таким образом, общее время вычислений блочного систолического алгоритма умножения матриц составляет:

$$T_{p,comp}^{BSys} = \left(\frac{2m^3}{p^2} + \frac{m^2}{p} \right) \cdot t_{op}.$$

Время на обменные операции по блочному систолическому алгоритму равно:

$$T_{p,comm}^{BSys} = 4(p-1) \cdot \left(t_s + \frac{m^2}{p^2} \cdot t_w \right),$$

где t_S – длительность подготовки сообщения для передачи; t_W – время передачи одного слова; m^2 / p^2 – объем передаваемых данных в словах.

Динамическими характеристиками, определяющими качество полученного параллельного алгоритма, являются коэффициенты ускорения и эффективности:

$$S = \frac{\left(2m^3\right) \cdot t_{op}}{\left(\frac{2m^3}{p^2} + \frac{m^2}{p} \right) \cdot t_{op} + 4(p-1) \cdot \left(t_s + \frac{m^2}{p^2} \cdot t_w \right)},$$

$$E = \frac{S}{p^2} = \frac{\left(2m^3\right) \cdot t_{op}}{\left(2m^3 + m^2 p\right) \cdot t_{op} + 4(p-1) \cdot \left(p^2 t_s + m^2 t_w\right)}.$$

Заметим, что везде речь идет о квадратных матрицах, так как интегрирование линейных СОДУ экспоненциальным методом [4] предполагает работу с квадратными матрицами. В произвольном случае при наличии прямоугольных матриц преобразование в квадратные происходит окаймлением нулевыми элементами по строкам или по столбцам.

2. Масштабируемый параллельный блочный полиалгоритм быстрого матричного умножения

Для плотных матриц имеется два принципиально различных класса последовательных алгоритмов умножения матриц: традиционные и рекурсивные методы на базе алгоритма быстрого умножения Штрассена. Вычислительная сложность матричного умножения $C = A \times B$ для исходных данных размерности $m \times m$ составляет: для традиционных алгоритмов – $O(m^3)$; для быстрого умножения по Штрассену – $O(m^{2.81})$ и $O(m^{2.376})$ – по методу Винограда.

В оригинале алгоритм Штрассена-Винограда – это алгоритм умножения блочных матриц половинного размера, где каждый блок квадратный, т.е. размерности матриц должны быть четными числами.

Метод Штрассена-Винограда состоит из 7 блочных умножений матриц и 15 блочных сложений\вычитаний матриц (рис. 2).

$$
\begin{array}{lll}
S_1 = A_{21} + A_{22} & M_1 = S_2 S_6 & T_1 = M_1 + M_3 \\
S_2 = S_1 - A_{11} & M_2 = A_{11} B_{11} & T_2 = T_1 + M_4 \\
S_3 = A_{21} - A_{12} & M_3 = A_{12} B_{21} & T_3 = M_5 + M_6 \\
S_4 = A_{12} - S_2 & M_4 = S_3 S_7 & C_{11} = M_2 + M_3 \\
S_5 = B_{12} - B_{11} & M_5 = S_1 S_5 & C_{12} = T_1 + T_3 \\
S_6 = B_{22} - S_5 & M_6 = S_4 B_{22} & C_{21} = T_2 - M_7 \\
S_7 = B_{22} - B_{12} & M_7 = A_{22} S_8 & C_{22} = T_2 + M_5 \\
S_8 = S_6 - B_{12} & &
\end{array}
$$

$$
A = \left\langle \frac{A_{11}}{A_{21}} \middle| \frac{A_{12}}{A_{22}} \right\rangle, B = \left\langle \frac{B_{11}}{B_{21}} \middle| \frac{B_{12}}{B_{22}} \right\rangle, C = \left\langle \frac{C_{11}}{C_{21}} \middle| \frac{C_{12}}{C_{22}} \right\rangle
$$

Рисунок 2 – Метод быстрого умножения матриц Штрассена-Винограда

Идея Штрассена может быть применена рекурсивно для нахождения произведений блоков матриц $M_i, i = \overline{1,7}$. Если исходные матрицы A и B имели размерность $m \times m$, то алгоритм быстрого умножения можно использовать многократно, получая на самом нижнем уровне рекурсии блоки 1×1. Однако нет никакой необходимости опускаться вниз до уровня блоков единичного порядка. При достаточно малых размерах блока ($k < k_{min}$) может оказаться выгодным вычислять блоки, используя стандартный алгоритм. Суммируя изложенное, составим вычислительную схему последовательного алгоритма Штрассена (рис. 3).

Вычислительная сложность предложенной схемы алгоритма быстрого умножения определяется функцией порядка исходных матриц и минимального порядка умножаемых блоков: m, m_{min}. Пусть при вычислении матричного умножения алгоритм Штрассена рекурсивно вызывался d раз, тогда порядок умножаемых матриц равен $m_{min} = m / 2^d$. На первом шаге алгоритм предусматривает 7 обращений к самому себе с матрицами порядка $m/2$ и 15 операций типа сложение для матриц того же порядка. Далее идет развертка рекурсии до достижения минимального размера блока и умножение блоков по традиционному алгоритму МУ.

Известный параллельный алгоритм классического метода Штрассена-Винограда был применен на Intel Paragon и показал хорошие результаты по сравнению с традиционными алгоритмами матричного умножения [6].

$$
\begin{aligned}
&\textit{function } C = Strassen(A, B, m_{min}) \\
&m = row(A) \\
&\textit{if } m \le m_{min} \textit{ then} \\
&\quad C = A \underset{ordinary}{\times} B \\
&\textit{else} \\
&\quad n = m/2;\ u = 1:n;\ v = n+1:m; \\
&\quad M_1 = Strassen(A(u,u) + A(v,v), B(u,u) + B(v,v), m_{min}) \\
&\quad M_2 = Strassen(A(v,u) + A(v,v), B(u,u), m_{min}) \\
&\quad M_3 = Strassen(A(u,u), B(u,v) - B(v,v), m_{min}) \\
&\quad M_4 = Strassen(A(v,v), B(v,u) - B(u,u), m_{min}) \\
&\quad M_5 = Strassen(A(u,u) + A(u,v), B(v,v), m_{min}) \\
&\quad M_6 = Strassen(A(v,u) - A(u,u), B(u,u) + B(u,v), m_{min}) \\
&\quad M_7 = Strassen(A(u,v) - A(v,v), B(v,u) + B(v,v), m_{min}) \\
&\quad C(u,u) = M_1 + M_4 - M_5 + M_7 \\
&\quad C(u,v) = M_3 + M_5 \\
&\quad C(v,u) = M_2 + M_4 \\
&\quad C(v,v) = M_1 + M_3 - M_2 + M_6 \\
&\textit{end} \\
&\textit{end Strassen}
\end{aligned}
$$

Рисунок 3 – Вычислительная схема последовательного рекурсивного алгоритма Штрассена-Винограда

Существенным недостатком этого алгоритма является отсутствие масштабируемости, так как требуемое число процессоров для него равно степени 7. Это ограничение является жестким и не естественным для большинства параллельных архитектур. Для преодоления указанного недостатка воспользуемся полиалгоритмическим подходом. Под полиалгоритмом подразумевается комбинация двух или более алгоритмов в одну вычислительную схему, реализующую поставленную задачу с целью сокращения вычислительных и/или емкостных затрат. Алгоритм быстрого умножения рекурсивен, поэтому имеется возможность построить полиалгоритм из некоторого традиционного алгоритма умножения матриц на верхнем уровне рекурсии и метода Штрассена на нижнем уровне. Алгоритм Штрассена является блочным, поэтому естественно комбинировать его со стандартным алгоритмом МУ, также использующим блочное разбиение данных. Вычислительная схема полиалгоритма блочного систолического алгоритма между процессорами и серии применений метода Штрассена на каждом процессоре приведена на рисунке 4.

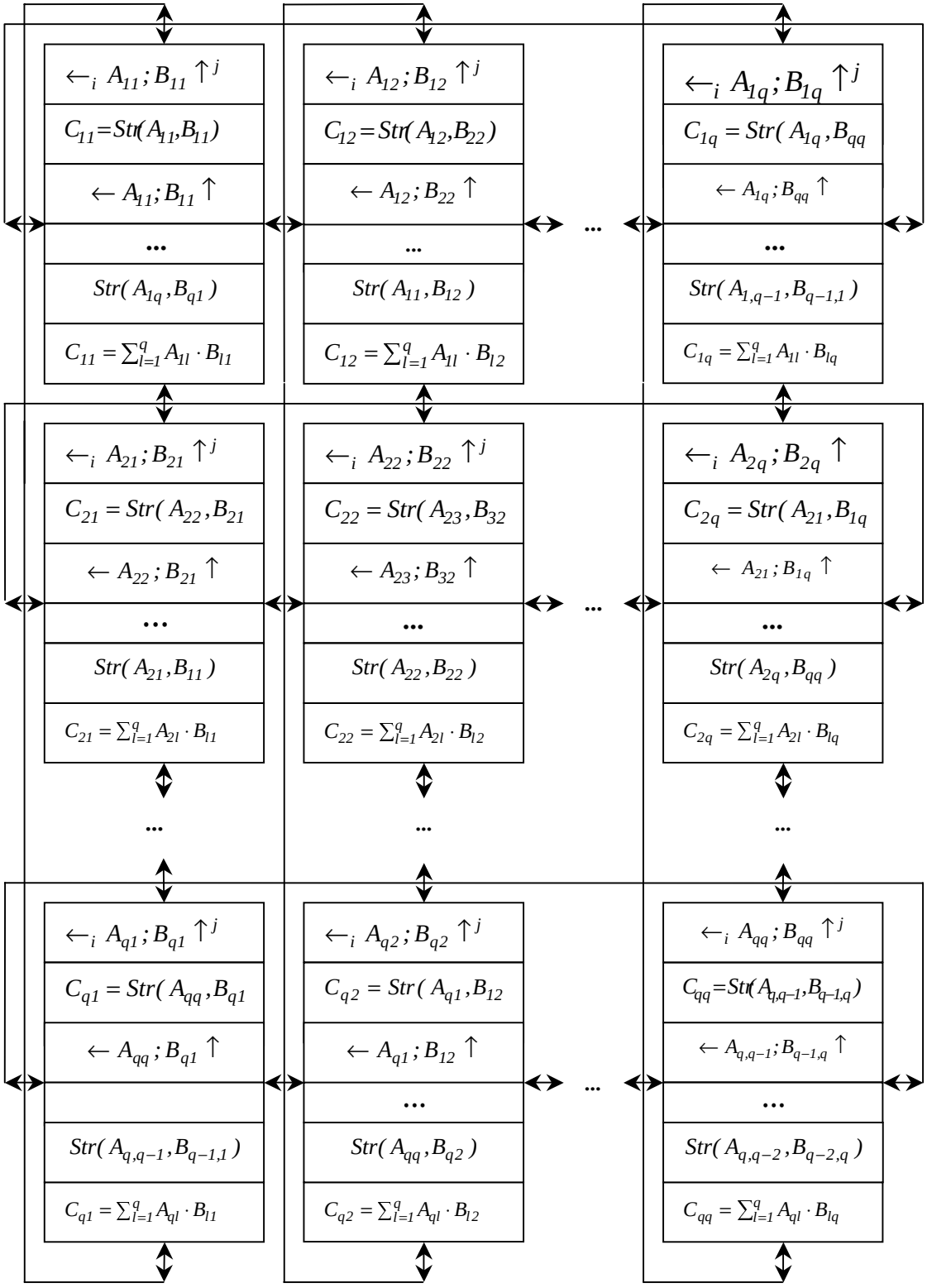

Рисунок 4 – Вычислительная схема полиалгоритма умножения матриц:
блочное систолическое умножение + алгоритм Штрассена
для мультикомпьютера топологии тор

Блоки исходных матриц и результата с координатами $< i, j >$ хранятся в соответствующем процессоре с теми же координатами. Предварительно по вычислительной схеме блочного систолического умножения выполняются:

1) $i \leftarrow A -$ косой сдвиг влево по строкам для блоков матрицы A;

2) $B \uparrow^{j} -$ косой сдвиг вверх по столбцам для блоков матрицы B.

На каждом из p шагов алгоритма производится умножение блоков матриц A и B, хранимых в процессоре с номером $< i, j >$ и сложение с уже вычисленным значением блока матрицы результата, расположенным на этом же процессоре. Затем производится одиночный сдвиг влево по строкам параллельно для блоков матрицы $A: A_{ij} \leftarrow A_{i,j+1}$ и одиночный сдвиг вверх по столбцам для блоков матрицы $B: B_{ij} \leftarrow B_{i+1,j}$. Умножение блоков матриц выполняется внутри одного процессорного элемента по рекурсивному алгоритму Штрассена. На нижнем уровне рекурсии применяется стандартный алгоритм умножения матриц, глубина рекурсии равна d. Разработанный алгоритм является масштабируемым для любого числа процессоров и порядка матриц.

Время реализации полиалгоритма включает время выполнения арифметических и обменных операций. Так, время выполнения вычислений для полиалгоритма равно:

$$T_{p,comp}^{BSys-Str} = q\left(T_{A_{ij} \underset{Str}{\times} B_{ij}} + T_{A_{ij}+B_{ij}} \right),$$

где $T_{A_{ij} \underset{Str}{\times} B_{ij}} -$ время, необходимое для реализации умножения блоков матриц порядка: $k = m / p$;

$T_{A_{ij}+B_{ij}} -$ время сложения блоков матриц той же размерности.

Время параллельного выполнения умножения блоков выполняется по алгоритму Штрассена и удовлетворяет следующему рекуррентному соотношению:

$$T_{A_{ij} \underset{Str}{\times} B_{ij}} = T_p^{Str}(k) = 7T_p^{Str}\left(\frac{k}{2}\right) + 15\left(\frac{k}{2}\right)^2 t_{ad}.$$

В свою очередь, время реализации алгоритма для матриц порядка $k/2,\ldots,k/2^{d-1}$ соответственно равно:

$$T_p^{Str}\left(\frac{k}{2}\right) = 7T_p^{Str}\left(\frac{k}{4}\right) + 15\left(\frac{k}{4}\right)^2 t_{ad},$$

...

$$T_p^{Str}\left(\frac{k}{2^{d-1}}\right)=7T_p^{Str}\left(\frac{k}{2^d}\right)+15\left(\frac{k}{2^d}\right)^2 t_{ad}.$$

Выполнив подстановки и элементарные преобразования, получим:

$$T_p^{Str}(k)=7^d T_p^{Str}\left(\frac{k}{2^d}\right)+\frac{15}{4}k^2\left(1+\frac{7}{4}+\frac{7^2}{4^2}+...+\frac{7^{d-1}}{4^{d-1}}\right)t_{ad}.$$

Пока определены арифметические операции для развертки рекурсии, рассмотрим внутренний уровень рекурсии, поскольку именно там выполняются операции умножения блоков минимального порядка по стандартному методу МУ:

$$T_p^{Str}\left(\frac{k}{2^d}\right)=k_{min}^3\cdot(t_{mul}+t_{ad})=2\left(\frac{k}{2^d}\right)^3 t_{op}.$$

Таким образом, время реализации быстрого рекурсивного умножения блоков матриц для мультикомпьютера равно:

$$T_p^{Str}(k)=\left[2\left(\frac{7}{8}\right)^d k^3+\frac{15}{4}k^2\left(1+\frac{7}{4}+\frac{7^2}{4^2}+...+\frac{7^{d-1}}{4^{d-1}}\right)\right]t_{op}.$$

Тогда, общее время выполнения арифметических операций для полиалгоритма равно:

$$T_{p,comp}^{BSys-Str}=q\left(T_p^{Str}\left(\frac{m}{p}\right)+T_{A_{ij}+B_{ij}}\right)=\left[2\left(\frac{7}{8}\right)^d\frac{m^3}{p^2}+\left(5\left(\frac{7}{4}\right)^d-4\right)\frac{m^2}{p}\right]t_{op}.$$

Время обменных операций для описанной схемы (рис.4) определяется, как и для блочного систолического алгоритма.

Качество параллельного полиалгоритма оценивается с использованием коэффициентов ускорения и эффективности:

$$S=\frac{T_1^{Str}}{T_p^{BSys-Str}},\quad E=\frac{T_1^{Str}}{T_p^{BSys-Str}\cdot p^2},$$

$$T_1^{Str}(m)=\left[2\left(\frac{7}{8}\right)^d m^3+m^2\left(5\left(\frac{7}{4}\right)^d-4\right)\right]t_{op}.$$

Очевидно, что динамические характеристики параллельных вычислительных схем МУ зависят от соотношения между числом процессоров и размерностью матриц. Для алгоритма Штрассена и полиалгоритмов на его основе существенным параметром является величина глубины рекурсии, d.

Анализ аналитических выражений, характеризующих выполнение параллельных алгоритмов, а также проведенный численный эксперимент, позволяют сделать следующие выводы (рис.5-6):

1) предложенная параллельная вычислительная схема полиалгоритма на основе быстрого умножения Штрассена имеет лучшее время выполнения по сравнению с блочным систолическим алгоритмом в $(8/7)^d$ раз для матриц больших размерностей и невысокой глубине рекурсии;

2) высокий уровень рекурсии отрицательно сказывается, как на времени выполнения, так и на объеме используемой памяти.

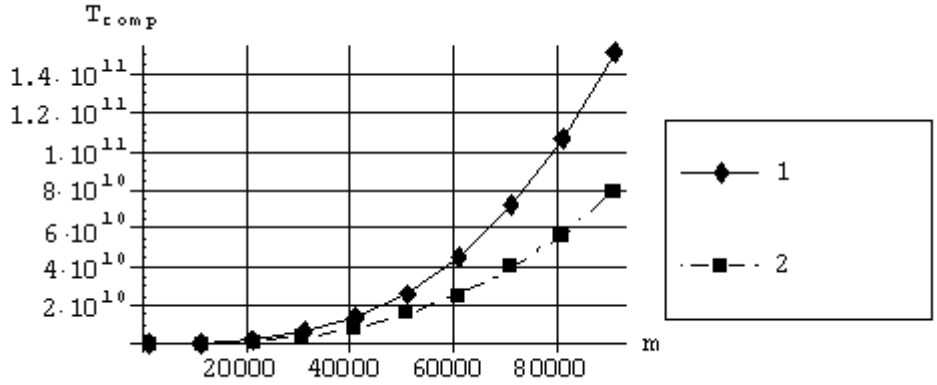

Рисунок 5 – График зависимости времени реализации параллельных алгоритмов от размерности матриц, $d = 4$
1)блочного систолического, 2) Штрассена+блочный систолический

Для параллельных компьютеров общепризнанным средством разработки коммуникаций является среда *MPI* (*Message Passing Interface*), поскольку интерфейс обмена сообщениями предоставляет программисту единый механизм взаимодействия ветвей внутри параллельного приложения независимо от машинной архитектуры и операционной системы. Тестирование алгоритмов в данной работе производилось на базе библиотеки *Argonne National Library MPICH-1.2.5*, одной из наиболее известных реализаций стандарта MPI для OS Windows. Все временные характеристики алгоритмов определялись с использованием функции *MPI_Barrier*. Для сокращения времени выполнения коллективных операций обмена использовались алгоритмы покоординатной маршрутизации.

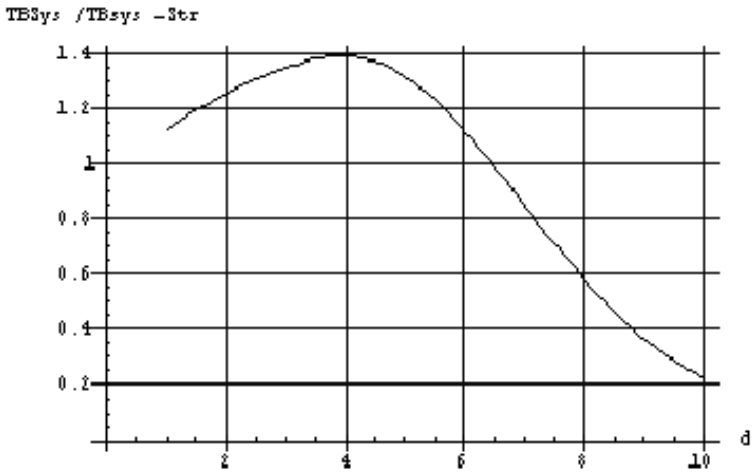

Рисунок 6 – График зависимости отношения времени реализации параллельных алгоритмов блочного систолического к полиалгоритму Штрассена + блочный систолический от глубины рекурсии, $m = 1000, p = 64$

Определение динамических характеристик параллелизма осуществлялось с помощью пакета *Mathematica®*.

Заключение

Аналитические исследования и численный эксперимент, показали, что применение комбинации алгоритма Штрассена и блочного систолического умножения матриц эффективно для матриц больших размерностей и малой глубины рекурсии. Перспективным направлением дальнейших исследований является разработка полиалгоритмов метода Штрассена на основе других традиционных алгоритмов МУ.

Перечень источников

1. Деммель Дж. Вычислительная линейная алгебра. Теория и приложения. Пер. с англ. – М.: Мир, 2001. – 430с.

2. Голуб Д., Ван Лоун Ч. Матричные вычисления: Пер. с англ. – М.: Мир, 1999. – 548с.

3. Kumar V., Gupta A. Analyzing scalability of parallel algorithms and architectures. // Journal of Parallel and Distributed Computing, 22(3), 1994, p.379-391.

4. Назарова И.А., Фельдман Л.П. Масштабируемый параллельный алгоритм численного решения линейных СОДУ для компьютеров с распределенной памятью. // Тезисы докладов V Международной конференции по неравновесным процессам в соплах и струях, Самара, 5-10 июля 2004 г. – М.: Вузовская книга, 2004. – с. 153–155.

5. Douglas C., Heroux M., Slishman G. A portable level 3 BLAS Winograd variants o Strassen’s matrix-matrix multiplay algorithm. / Journal of computational physics, 110, 1994, p.1-10.