УДК 004.04, 519.688

О распараллеливании рекурсивных блочных алгебраических алгоритмов: алгоритмы и эксперименты

Е. А. Ильченко

Лаборатория алгебраических вычислений,
Тамбовский государственный университет,
ул. Интернациональная, д.33, Тамбов, Россия, 392000

Дается описание общей схемы распараллеливания блочных рекурсивных алгоритмов, которая рассматривается на примере вычисления присоединенной матрицы, приводятся результаты экспериментов на вычислительном кластере «МВС-10Р»

Ключевые слова: параллельный алгоритм; децентрализованное управление; расширенная присоединенная матрица; Adjoint; система Mathpar; SPMD вычислительная парадигма; MPI; многопоточность; hybrid parallel programming model; C++; GMP..

1. Краткий обзор работы

Рассмотрим блочный алгоритм обращения матрицы, имеющий сложность матричного умножения [1,2]. Если $M = \begin{pmatrix} A & B \\ C & D \end{pmatrix}$ является обратимой матрицей и A ее обратимый блок, то верно равенство:

$$M^{-1} = \begin{pmatrix} \boldsymbol{I} & -A^{-1}C \\ 0 & \boldsymbol{I} \end{pmatrix} \times \begin{pmatrix} \boldsymbol{I} & 0 \\ 0 & (D - BA^{-1}C)^{-1} \end{pmatrix} \times \begin{pmatrix} \boldsymbol{I} & 0 \\ -B & \boldsymbol{I} \end{pmatrix} \times \begin{pmatrix} A^{-1} & 0 \\ 0 & \boldsymbol{I} \end{pmatrix},$$

где $\boldsymbol{I}$ - единичная матрица. Раскрывая скобки, получим:

$$M^{-1} = \begin{pmatrix} A^{-1} + A^{-1}C\Big(D - BA^{-1}C\Big)^{-1}BA^{-1} & -A^{-1}C\Big(D - BA^{-1}C\Big)^{-1} \\ -\Big(D - BA^{-1}C\Big)^{-1}BA^{-1} & \Big(D - BA^{-1}C\Big)^{-1} \end{pmatrix}.$$

Обозначим обращение матрицы как функцию $\mathfrak{g}(M) = E$, тогда полученная формула примет вид:

$$\mathfrak{g}(M) = \begin{pmatrix} B_0 & B_1 \\ B_2 & B_3 \end{pmatrix},$$

где

$$B_0 = \mathfrak{g}(A) + \mathfrak{m}\Big(\mathfrak{m}\Big(\mathfrak{m}\big(\mathfrak{g}(A), \mathfrak{m}(C, B_3)\big), B\Big), \mathfrak{g}(A)\Big), B_1 = -\mathfrak{m}\big(\mathfrak{g}(A), \mathfrak{m}(C, B_3)\big),$$

$$B_2 = -\mathfrak{m}\Big(B_3, \mathfrak{m}\big(B, \mathfrak{g}(A)\big)\Big), B_3 = \mathfrak{g}\Big(D - \mathfrak{m}\big(\mathfrak{m}\big(B, \mathfrak{g}(A)\big), C\big)\Big).$$

Как мы видим, в формуле для нахождения $\mathfrak{g}(M)$ в различной форме встречается суперпозиция функций $\mathfrak{m}$ и $\mathfrak{g}$. Рассмотрим один из случаев, например $\mathfrak{m}\big(B, \mathfrak{g}(A)\big)$, что означает матричное умножение $B \times A^{-1}$. Чтобы выполнить матричное умножение, сначала нужно вычислить A^{-1}. Эту и другие зависимости порядка вычислений одних функций от других можно представить в виде графа, где вершинами будут все функции, из которых состоит рекурсивный алгоритм, а связи будут задавать порядок вычислений вершин. К примеру, если между вершинами a и b есть

дуга (a, b), то это означает, что вершина b должна быть посчитана строго после вершины a. Обозначим $-A^{-1} = V$, $V \times C = X$, $B \times V = Y$, $Y \times C = Q$, $(D+Q)^{-1} = Z$, $Z \times Y = W$, тогда формула для нахождения обратной матрицы будет выглядеть следующим образом:

$$M^{-1} = \begin{pmatrix} X \times W + A^{-1} & X \times Z \\ W & Z \end{pmatrix}. \tag{2}$$

Для алгоритма нахождения обратной матрицы мы имеем два типа вершин - функции $\mathfrak{m}$ и $\mathfrak{g}$.

Отсюда и далее вершину графа алгоритма со всеми присущими ей особенностями мы будем называть *задачей*. Если более формально, то *задача* - это множество входных данных A для алгоритма, множество выходных данных B, алгоритм F, отображающий $A \to B$. С задачей всегда будет связан некоторый граф G, определяющий подзадачи алгоритма и порядок их вычислений. Еще одно важное свойство любой задачи заключается в том, что она *может* быть посчитана параллельно. Если же некоторую последовательность операций алгоритма нет смысла выполнять параллельно, мы не будем выделять эти действия в отдельную вершину, или создавать для них отдельную задачу, а вынесем эти действия в предварительную или постобработку данных для некоторой вершины.

Рассмотрим вершину с номером 0 для алгоритма нахождения обратной матрицы. Входные данные для задачи - матрица A, выходными будет $\mathfrak{g}(A) = A^{-1}$. Но поскольку нам нужна матрица $-A^{-1}$, а не A^{-1}, то после того, как эта вершина будет посчитана, необходимо выполнить некоторые преобразования матрицы A^{-1}. Действия, заключающиеся в некотором преобразовании выходных данных вершины, мы будем называть *постобработкой*. Действия, заключающиеся в инициализации входных данных вершины, мы будем называть *инициализацией вершины*. Инициализация происходит в тот момент, как только вершина становится *доступной* - то есть тогда, когда все зависимые для нее вершины посчитаны. Постобработка происходит в тот момент, как только получены выходные данные для этой задачи и данные, необходимые для постобработки этой вершины.

Входные данные для вершины с номером k будем обозначать как I_k, результат будем обозначать как R_k (под результатом понимаются данные, полученные по завершению постобработки вершины). Тогда, принимая во внимание описание вершин из таблицы 2, уравнение (2) можно записать в виде:

$$M^{-1} = \begin{pmatrix} R_7 & R_6 \\ R_5 & R_4 \end{pmatrix}.$$

Для реализации поставленной задачи используется стратегия, при которой на каждом вычислительном узле кластера будет запущено по одному MPI-процессу, каждый из которых будет иметь по одному диспетчерскому и несколько вычислительных потоков (в зависимости от количества ядер на узле).

2. Заключение

Для исследований алгоритма была написана программа на C++. Исследовались различные алгоритмы для умножения разреженных матриц, алгоритм вычисления присоединенной матрицы. Для реализации многопоточности была выбрана библиотека Pthreads, для реализации арифметики многократной точности была взята библиотека GMP. Эксперименты проводились на кластере «МВС-10Р», которые показали хорошую эффективность и масштабируемость. Результаты эксперимента с алгоритмом блочного матричного умножения показаны на рисунке 1. Результаты других экспериментов будут представлены в докладе.

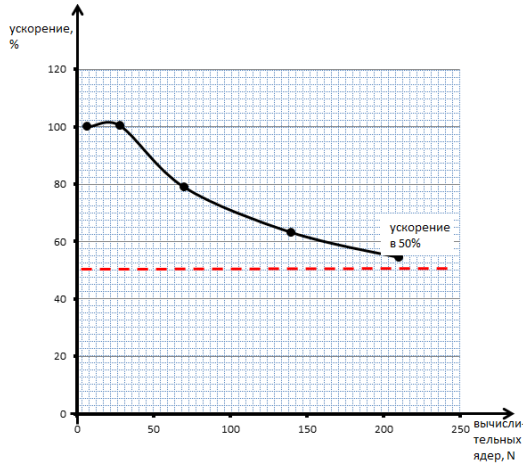

Рис. 1. Эффективность параллельной реализации алгоритма блочного матричного умножения, где размер исходных матриц равнялся 2048x2048, плотность была 100%.

Благодарности

Работа частично поддержана грантом РФФИ № 16-07-00420.

Литература

1. *Strassen V.* Gaussian Elimination is not optimal. Numerische Mathematik. 1969. — **1**3, P.354-356.
2. *Malaschonok G.I.* Matrix calculation methods in commutative rings. —Tambov: Publishing House of Tambov State University, 2002.

UDC 004.04, 519.688

About Parallelizing Recursive Block Algebraic Algorithms: Algorithms and Experiments

E. A. Ilchenko

Laboratory for algebraic computations
Tambov State University
Internatsionalnaya,33, Tambov, Russia, 392000

We describe an algorithm for the decentralized control of parallel computing process which is based on the SPMD computational paradigm and present the results of experiments on a cluster «MVS-10P».

Key words and phrases: Parallel algorithm, decentralized control, Adjugate matrix, the system Mathpar, SPMD computational paradigm, adjoint, MPI, multithreading, hybrid parallel programming model, C++, GMP. .