Распараллеливание рекуррентного процесса с бесконечной историей для решения СЛАУ с треугольной матрицей

Бзикадзе Ал.В., студент 2 курса СПБГУ, alexander.bzikadze@gmail.com

Екатерина Е.А., студент 2 курса СПБГУ, katrine192002@gmail.com

Аннотация

Системы линейных алгебраических уравнений являются важным методом решения множества задач, потому задача эффективного решения самой СЛАУ актуальна. В данной работе рассмотрен метод решения СЛАУ с треугольной матрицей средставми параллельного программирования с помощью ускоренного рекуррентного процесса с бесконечной историей.

Введение

Системы линейных алгебраических уравнений (СЛАУ) являются естественным способом решения многих задач, возникающих в таких областях, как экономика, физика, химия и другие. Решения СЛАУ - одна их классических задач линейной алгебры. Мы рассмотрим частный случай СЛАУ, когда матрица, соответствующая системе, является треугольной. Для начала введём некоторые соглашения [1]. Предполагаем, что:

1. Имеется потенциально неограниченное количество параллельных вычислительных модулей.
2. Имеется потенциально неограниченная память вычислительной системы.
3. Имеется набор параллельных операций, которые называются основными. Будем считать, что каждая такая операция выполняется за один такт вычислительной системы.
4. Другие операции считаются неосновными, время их выполнения пренебрежимо мало.
5. Доступ всех вычислительных модулей в память одинаков и нет коллизий (столкновений) при обращении к памяти.
6. Все исходные данный считаются уже записанными в память.
7. Все результаты вычислений тоже находятся в памяти.

В данных условиях мы и будем рассматривать следующую задачу: решение СЛАУ с треугольной матрицей средствами параллельного программирования.

Сведение к рекуррентному соотношению

Рассмотрим СЛАУ размерности n:

$$\begin{cases} a_{11} * x_1 & = y_1, \\ & \vdots \\ a_{n1} * x_1 + \ldots + a_{nn} * x_n & = y_n \end{cases}$$

Обозначим как A соответствующую ей матрицу, X - вектор неизвестных, Y - вектор свободных членов. В таких обозначениях получаем

$$A = \begin{pmatrix} a_{11} & \ldots & 0 \\ \vdots & \ddots & \vdots \\ a_{n1} & \ldots & a_{nn} \end{pmatrix}, X = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}, Y = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix}$$

СЛАУ приобретает вид $A * X = Y$.

Путем несложных преобразований получаем следующую систему:

$$\begin{cases} x_1 = \frac{y_1}{a_{11}}, \\ \vdots \\ x_n = \frac{y_n}{a_{nn}} - \frac{a_{n\,n-1}}{a_{nn}} * x_{n-1} - \ldots - \frac{a_{n1}}{a_{nn}} * x_1 \end{cases}$$

Таким образом, общий член выражается рекурсивно следующим образом:

$$x_j = \frac{y_j}{a_{jj}} - \frac{a_{j\,j-1}}{a_{jj}} * x_{j-1} - \ldots - \frac{a_{j1}}{a_{jj}} * x_1$$

.

Наивное решение с помощью традиционного итерационного процесса

***Теорема* 0.1.** *[1] Для рекуррентного процесса существует параллельная форма высоты* $H = \lceil \log(s+1) \rceil * (\lceil \log(n*r+1) \rceil + 1)$ *и ширины* $W = (\lceil \frac{s+1}{2} \rceil * (n*r+1)^3$

Введем следующие обозначения: j - номер n-мерного вектора, r - величина рекуррентной истории, s - количество векторов X, которых нам надо найти, n - размерность вектора X, A_{ji} - элемент квадратной матрицы n-го порядка, B_j - элемент n-мерного вектора. Тогда получаем

$$\begin{cases} j = 1, .., n, \\ r = n - 1, \\ s = n, \\ X_j = x_j, \\ A_{ji} = -\frac{a_{jj-1}}{a_{jj}}, j > i \\ A_{ji} = 0, else \end{cases}$$

В таких обозначениях получаем следующую запись для общего члена последовательности.

$$X_j = A_{j1} * X_{j-1} + A_{j2} * X_{j-2} \ldots + A_{jr} * X_{j-r} + B_j$$

Применяя **Теорему 0.1** получаем параллельную форму для решения нашего рекуррентного соотношения со следующими параметрами.

$$\begin{cases} H = \lceil \log_2 (n+1) \rceil * (\lceil \log_2 n \rceil + 1) = O(\log^2 n) \\ W = \lceil (n+1)2 \rceil * n^3 = O(n^4) \end{cases}$$

Подобный результат нас не устраивает и мы утверждаем, что данный процесс можно улучшить.

Ускорение итерационного процесса

Для начала, введем вспомогательные теоремы, которые позволят нам перемножать матрицы разных размерностей (естественно, когда такое умножение определено).

***Теорема* 0.2.** *Пусть имеется ассоциативный моноид с 1. Схема сдваивания для произведения n элементов этого моноида имеет высоту* $H = \lceil \log_2(n) \rceil$ *и ширину* $W = \lceil \frac{n}{2} \rceil$

***Теорема* 0.3.** *Дана матрица размерности* $n \times m$ *и вектор размерности* m. *Для перемножения матрицы на вектор можно получить параллельную форму с высотой* $H = \lceil \log_2 m \rceil + 1$ *и шириной* $W = n * m$

Доказательство. На I ярусе параллельной системы происходит n * m умножений $\Rightarrow$ $H_I = 1$, $W_I = n * m$ На II ярусе параллельной системы происходит сложение m элементов по схеме сдваивания $\Rightarrow$ $H_{II} = \lceil \log_2(m) \rceil$, $W_{II} = \lceil \frac{m}{2} * m \rceil$. Таким образом, общие результаты: $H = \lceil \log_2(m) \rceil + 1$, $W = n * m$. ☐

***Теорема* 0.4.** *Даны матрицы* $A_{n \times k}$, $B_{k \times m}$, $C_{n \times k}$, $A * B = C$. *Существует параллельная форма с высотой* $H = 1 + \lceil \log(k) \rceil$ *и шириной* $W = nkm$ *для их перемножения.*

Доказательство. Рассмотрим матрицу B как строчку векторов: $B = \begin{pmatrix} B_1 & \dots & B_m \end{pmatrix}$. Тогда C представляется в виде $C = \begin{pmatrix} A * B_1 & \dots & A * B_m \end{pmatrix}$. Очевидно, что перемножение векторов независимо, что позволяет нам применить **Теорему 0.3**. Так как мы выполняем параллельное перемножение m векторов размера k на матрицу размера $n \times k$, то мы получаем параллельную форму высоты $H = 1 + \lceil \log(k) \rceil$ и ширины $W = nkm$. ☐

Определение 0.1. Рекуррентное соотношение называется "с конечной историей", если любой член последовательности зависит от одного фиксированного конечного значения предыдущих элементов.

Определение 0.2. Рекуррентное соотношение называется "с бесконечной историей", если оно не является рекуррентным соотношением с конечной историей. В частности, соотношение, когда любой член последовательности зависит от всех предыдущих также входит в данную категорию.

***Теорема* 0.5.** *Для рекуррентного процесса с бесконечной историей существует параллельная форма с высотой* $H = O(log^2(n)$ *и шириной* $W = O(n^3)$

Доказательство. Пусть рекуррентное соотношение задано так же, как в секции 2, но без нулевых членов:

$$X_1 = B_1$$

$$\vdots$$

$$X_j = A_{j1} * X_{j-1} + A_{j2} * X_{j-2} \ldots + A_{jj-1} * X_1 + B_j$$

Введем строчку $Q_j = \begin{pmatrix} A_{j1} & \ldots & A_{jj-1} & B_j \end{pmatrix}$ и вектор $X^j = \begin{pmatrix} X_j \\ \vdots \\ X_1 \\ 1 \end{pmatrix}$. Тогда очевидно, что $\begin{pmatrix} Q_j \\ E_j \end{pmatrix} * X^{j-1} = X^j$. Из данного соотношения получаем, что вектор X^n может быть выражен следующим образом:

$$X^n = \begin{pmatrix} Q_n \\ E_n \end{pmatrix} X^{n-1} = \begin{pmatrix} Q_n \\ E_n \end{pmatrix} \begin{pmatrix} Q_{n-1} \\ E_{n-1} \end{pmatrix} \ldots \begin{pmatrix} Q_1 \\ E_1 \end{pmatrix} X_0$$

Именно здесь начинается содержательное улучшение рекуррентного процесса. Наивный способ - начать перемножать матрицы по схеме сдваивания. На первом шаге параллельных вычислений мы получим следующую ширину:

$$w = (n+1)n(n-1) + (n-1)(n-2)(n-3) + \ldots = O(n^4)$$

Безусловно, это не отвечает требованиям теоремы. Заметим, что матрицы имеют специальный вид, потому в каждой паре перемножений достаточно перемножать только первую строчку левой матрицы на правую матрицу, а вместо умножения слева на единичную матрицу просто “переписывать” правую матрицу. Воспользуемся схемой сдваивания.

$$X^n = \begin{pmatrix} Q_n \begin{pmatrix} Q_{n-1} \\ E_{n-1} \end{pmatrix} \\ Q_{n-1} \\ E_{n-1} \end{pmatrix} \begin{pmatrix} Q_{n-2} \begin{pmatrix} Q_{n-3} \\ E_{n-3} \end{pmatrix} \\ Q_{n-3} \\ E_{n-3} \end{pmatrix} \ldots \begin{pmatrix} Q_2 \begin{pmatrix} Q_1 \\ 1 \end{pmatrix} \\ Q_1 \\ 1 \end{pmatrix}$$

Оценим ширину данного вычислительного этапа. Заметим, что число перемножений есть $\lceil \frac{n}{2} \rceil$.

$$w = 1 * n(n-1) + 1(n-2)(n-3) + \ldots \leq n^2 + n^2 + \ldots = n^2 * \left\lceil \frac{n}{2} \right\rceil = O(n^3)$$

Отметим, что число оставшихся матриц уменьшилось двое, а размер “содержательной” части увеличился вдвое - до двух строчек. Продолжим перемножение нашим методом: умножаем только первые две строчки левой матрицы. Оценим получившуюся ширину.

$$w = 2 * n(n-2) + 2 * (n-2)(n-4) + \ldots \leq 2 * n^2 + 2 * n^2 + \ldots =$$
$$= 2 * n^2 * \left\lceil \frac{\left\lceil \frac{n}{2} \right\rceil}{2} \right\rceil \approx n^2 * \left\lceil \frac{n}{2} \right\rceil = O(n^3)$$

Очевидно, что данная ситуация повторяется на каждом шаге: количество содержательных строчек в матрице возрастает вдвое, количество матриц уменьшается вдвое, оценка сверху перемножения одной строчки на матрицу справа - $O(n^2)$. Тогда общая оценка на ширину вычислений - $O(n^3)$. Оценим высоту вычислений. Так как мы в действительности применяем схему сдваивания, то высота без учета высоты перемножений равна $O(\log n)$. В то же время высота вычислений перемножения матриц - $O(\log n)$. Тогда итоговая высота есть $O(\log^2 n)$.

□

Применение ускоренного итерационного процесса для решения СЛАУ с треугольной матрицей

***Теорема* 0.6.** *Для решения СЛАУ с треугольной матрицей существует параллельная форма с высотой $H = O(log^2(n))$ и шириной $W = O(n^3)$*

Доказательство. Обратимся к секции 2. Полученное там рекуррентное соотношение можно преобразовать в соотношение с бесконечной историей, выкинув все нулевые множители. Воспользуемся **Теоремой 0.5** и получим параллельную форму с высотой $H = O(log^2(n))$ и шириной $W = O(n^3)$.

□

Заключение

В данной работе был рассмотрен подход решения СЛАУ с треугольной матрицей с помощью традиционного рекурентного процесса. В данной работе был получен эффективный метод решения рекуррентных соотношений с бесконечной историей средствами параллельного программирования. Задача решения СЛАУ с треугольной матрицей была сведена к рекуррентному соотношению с бесконечной историей и предложенный метод применен для решения поставленной задачи.

Литература

[1] И.Г.Бурова, Ю. К. Демьянович, *Алгоритмы параллельных вычислений и программирование*. СПб

[2] A. H. Sameh, R. P. Brent, *Solving triangular systems on a parallel computer* http://maths-people.anu.edu.au/ brent/pd/rpb041.pdf