УДК 519.213:519.257:519.657

Е.В. Крылов, А.В. Плащинская, П.А. Базурин

ИССЛЕДОВАНИЕ МЕТОДОВ ПОВЫШЕНИЯ БЫСТРОДЕЙСТВИЯ ПРОГРАММЫ ПРИ ИСПОЛЬЗОВАНИИ ОПЕРАЦИЙ НАД МАТРИЦАМИ БОЛЬШОЙ РАЗМЕРНОСТИ

Аннотация: В данной статье выполнено исследование методов повышения быстродействия программы при использовании операций над матрицами большой размерности: распараллеливание вычислительных процессов, оптимизации алгоритмов вычислений и использования специальных библиотек.

Ключевые слова: быстродействия программы, матрицы большой размерности, алгоритмов вычислений, QuickSort.

Вступление

Писать код, который быстро работает, иногда противоречит с написанием кода быстро. К.А.Р. Хор, светила компьютерной науки и первооткрыватель алгоритма QuickSort, громко заявил: “Преждевременная оптимизация является корнем всех зол”. Совершение простых операций на **матрицами большой размерности** могут быть одной из причин “замедления” работы программы и получения требуемых результатов.

Постановка задачи

Быстродействие является одной из решающих характеристик созданной программы. Для достижения максимальных результатов в этой области используется несколько методов:

1. Распараллеливание вычислительных процессов.
2. Оптимизация алгоритмов вычислений.

Существует два подхода к распараллеливанию вычислений в микропроцессорных системах, называемые однопоточным и многопоточным параллелизмом. Однопоточный параллелизм заключается в параллельном выполнении операций внутри одного потока исполнения. Многопоточный параллелизм заключается в использовании нескольких потоков для достижения параллельного исполнения операций.

Конкретизация задачи

Покажем использование методов повышения быстродействия на примере задачи моделирования процесса разрушения плоской детали под действием циклического нагружения. При ее решении используются практически все операции над матрицами большой размерности: умножение, сложение, транспонирование, нахождение обратной матрицы.

Рассмотрим тонкую пластину шириной W и длиной H с центральной трещиной длиной $2\ell_0$, находящуюся под действием циклического нагружения

$$\tilde{\sigma} = \sigma_a g(n), \tag{1}$$

где σ_a- - амплитудное напряжение цикла; $g(n)$ – известная периодическая функция изменения нагрузки от числа циклов нагружения n. Задача состоит в определении скорости роста усталостной трещины.

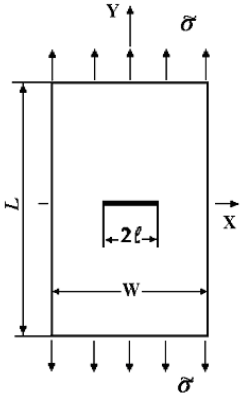

Рис. 1 – Схема нагрузки пластины

Для решения задачи воспользуемся численным методом конечных элементов.

Вкратце алгоритм состоит в следующем:

- зададим геометрию пластины;
- разобьем ее область на треугольные элементы;
- формируем массивы входных данных (матрицы координат узлов, матрицы узловых значений элементов, матрицы граничных условий);
- рассчитываем матрицу жесткости для каждого элемента (ее размерность только 6 x 6):

$$[K]_m = [B]^T [D] [B] D_m; \tag{2}$$

- расширяем эти матрицы до необходимой для пластины размерности $2N \times 2N$ (где N - количество узлов). Для этого записываем рассчитанные для определенного элемента данные в расширенную матрицу, а все остальные значения задаем равными 0.

С этого момента и начинается работа с матрицами большой размерности.

- затем используя уравнение равновесия для всей конструкции

 ISSN 1560-8956

$$\{F\} = [K]\{\delta\} \tag{3}$$

находим необходимые значения узловых сил $\{F\}$ и узловых перемещений $\{\delta\}$. Однако для этого нам понадобится обратная матрица $[K]$, т. е. алгоритм ее получения. После этого нам придется перемножать заданные массивы большой размерности;

- находим перемещение любой точки внутри элемента:

$$\{f(x,y)\} = \left\{ \begin{array}{c} u \\ \upsilon \end{array} \right\} = \left[\begin{array}{cccccc} N_i & 0 & N_j & 0 & N_k & 0 \\ 0 & N_i & 0 & N_j & 0 & N_k \end{array} \right] \left[\begin{array}{c} u_i \\ \upsilon_i \\ u_j \\ \upsilon_j \\ u_k \\ \upsilon_k \end{array} \right] = [N]\{\delta\}_{(m)}; \tag{4}$$

- получаем деформацию и напряжение внутри элемента:

$$\{\varepsilon\} = \left[\begin{array}{c} \varepsilon_x \\ \varepsilon_y \\ \gamma_{xy} \end{array} \right] = \frac{1}{2\Delta_m} \left[\begin{array}{cccccc} b_i & 0 & b_j & 0 & b_k & 0 \\ 0 & c_i & 0 & c_j & 0 & c_k \\ c_i & b_i & c_j & b_j & c_k & b_k \end{array} \right] * \left[\begin{array}{c} u_i \\ \upsilon_i \\ u_j \\ \upsilon_j \\ u_k \\ \upsilon_k \end{array} \right] = [B]\{\delta\}_{(m)}, \tag{5}$$

$$\{\sigma\} = \left[\begin{array}{c} \sigma_{xx} \\ \sigma_{yy} \\ \tau_{xy} \end{array} \right] = \frac{E}{1-\nu^2} \left[\begin{array}{ccc} 1 & \nu & 0 \\ \nu & 0 & 0 \\ 0 & 0 & \frac{1-\nu}{2} \end{array} \right] * \left[\begin{array}{c} \varepsilon_{xx} \\ \varepsilon_{yy} \\ \gamma_{xy} \end{array} \right] = [B]\{\varepsilon\}. \tag{6}$$

Из тензора напряжений определяем эквивалентное напряжение для каждого элемента

$$\sigma_{eqv} = \sqrt{\sigma_{xx}^2 + \sigma_{yy}^2 - \sigma_{xx}\sigma_{yy} + 3\tau_{xy}^2}. \tag{7}$$

Определяя коэффициенты интенсивонсти напряжений по формуле

$$K_I = \sigma_{eqv}\sqrt{2\pi(l_i + r)}. \tag{8}$$

при различных координатах r, линейным экстраполированием координат точек вдоль линии трещины, относящимся к разным элементам получим значение К в вершине трещины ($r = 0$).

Решая задачу (2-8) в цикле ($i = \overline{1, N}$) для трещин различной длины $l_0 \leqslant l_i \leqslant l_R$ путем апроксимации численных значений K_I, определим $\Delta K = K_{\max} - K_{\min}$.

- последним этапом будет вычисление скорости роста трещины в пластине

$$\frac{d\ell}{dn} = \left(1 + \frac{1}{q}\right) D \cdot \frac{(4\sigma_\Upsilon)^{q-2}}{\pi} (\Delta K)^2. \quad (9)$$

При решении данной задачи основной время уходит на работу с данными матрицами большой размерности.

Применение методов увеличения быстродействия

Применим все выше перечисленные метода для программы, реализующей алгоритм расчета, указанный выше.

Распараллеливание вычислительных процессов

Применим этот метод для рассматриваемой задачи. Распараллелим основные процессы для возможности их выполнения несколькими процессорами. Кроме того, второстепенные задачи тоже можно разложить на несколько исполняющих ядер. Об этих методах будет рассказано ниже в разделе оптимизации алгоритмов вычислений.

Рис. 2 – Модель распараллеливания алгоритма расчетов

 ISSN 1560-8956

Оптимизация алгоритмов вычислений

Некоторые операции в программе могут выполняться несколькими методами на основе разнообразия созданных алгоритмов. Одной из таких операций есть нахождение обратной матрицы.

Методы нахождения обратной матрицы:

- точные/прямые методы (метод Гаусса, с помощью союзной матрицы, использование LU/LUP разложения, т.п.);
- итерационные методы (метод Шульца и т.п.).

Для данной задачи подходят только точные методы. Причем для небольших матриц можно воспользоваться модификацией метода Гаусса, например метод Ершова. В данном же случае, надо использовать иные методы, например через LU – разложение или метод Халецкого. Кроме того, в зависимости от структуры матрицы предпочтение может быть дано тому или другому алгоритму.

Также необходимо достичь параллельности вычисления. Рассмотрим задачу вычисления обратной матрицы A^{-1} для квадратной матрицы A порядка n. Для ее построения используем теорему Гамильтона-Кэли

$$f(\lambda) = \lambda^n + c_1\lambda^{n-1} + \ldots + c_n, \tag{10}$$

где $f(\lambda)$ - характеристический многочлен матрицы A. Согласно теореме Гамильтона-Кэли подстановка матрицы A в характеристический многочлен получается нулевая матрица. Отсюда находим

$$A^{-1} = -\frac{1}{c_n}(A^{n-1} + c_1A^{n-2} + \ldots + c_{n-1}E), \tag{11}$$

где E – единичная матрица.

Пусть λ_i - корни характеристического уравнения, а s_k - сумма их k степеней

$$s_k = \sum_{i=1}^{n} \lambda_i^k. \tag{12}$$

Известно соотношение Ньютона:

$$\begin{pmatrix} 1 & 0 & 0 & \ldots & 0 & 0 \\ s_1 & 2 & 0 & \ldots & 0 & 0 \\ s_2 & s1 & 3 & \ldots & 0 & 0 \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ s_{n-1} & s_{n-2} & s_{n-3} & \ldots & s_1 & n \end{pmatrix} \begin{pmatrix} c_1 \\ c_2 \\ c_3 \\ \ldots \\ c_n \end{pmatrix} = \begin{pmatrix} -s_1 \\ -s_2 \\ -s_3 \\ \ldots \\ -s_n \end{pmatrix}. \tag{13}$$

Числа s_k можно вычислить, как сумму диагональных элементов матрицы A^k.

Построение параллельной формы для определения элементов матрицы A^k сводится к следующим этапам:

1. По схеме сдваивания находим все степени матрицы до n; это требует выполнения $O(\log_2 n)$ макро-шагов, на каждом их которых выполняется $O(n)$произведения двух квадратных матриц порядка n.
2. На втором этапе вычисляются все числа s_k, как следы матриц A^k, $k = 1, \ldots, n$; поскольку след – сумма диагональных элементов, то требуется вычислить n сумм с n слагаемыми.
3. Теперь отыскиваются коэффициенты c_k характеристического многочлена решением системы n-го порядка с треугольной матрицей.
4. Вычисляется матрица A^{-1} с использованием схемы сдваивания.

Для наглядности:

Таблица 1

Табличное представление распараллеливания процесса расчета обратной матрицы.

Содержание этапа	**Высота**	**Ширина**
Вычисление степеней $A^2, \ldots, A^n$	$O(\log_2^2 n)$	$O(n^4)$
Вычисление следов s_k, $k = 1, \ldots, n$	$O(\log_2^2 n)$	$O(n^2)$
Вычисление коэффициентов c_k	$O(\log_2^2 n)$	$O(n^3)$
Вычисление матрицы A^{-1}	$O(\log_2^2 n)$	$O(n^3)$

Но всегда будет борьба простоты реализации алгоритма и времени вычисления. Реализуем обычный метод Халецкого.

Также при совершении операций над матрицами большой размерности будут использоваться всевозможные алгоритмы нахождения обратных матриц, их транспонирования, стандартные математические методы сложения, вычитания или перемножения матриц. Таких алгоритмов несколько, выберем из них самый эффективный, быстрый, с возможностью разложить расчеты на несколько потоков.

Например, при перемножении двух матриц большой размерности:

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}, B = \begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1q} \\ b_{21} & b_{22} & \cdots & b_{2q} \\ \vdots & \vdots & \ddots & \vdots \\ b_{p1} & b_{p2} & \cdots & b_{pq} \end{bmatrix}. \qquad (14)$$

Нужно получить конечную матрицу:

$$C = \begin{bmatrix} c_{11} & c_{12} & \vdots & c_{1q} \\ c_{21} & c_{22} & \vdots & c_{2q} \\ \cdots & \cdots & \ddots & \cdots \\ c_{m1} & c_{m2} & \vdots & c_{mq} \end{bmatrix}. \qquad (15)$$

 ISSN 1560-8956

Результат находится по формуле:

$$c_{i,j} = \sum_{r=1}^{n} a_{i,r} b_{r,j} (i = 1, \ldots, m; j = 1, \ldots, q). \quad (16)$$

При этом результаты для каждого столбца, строки или элемента можно сделать отдельной подзадачей для каждого процессора и выполнять их параллельно:

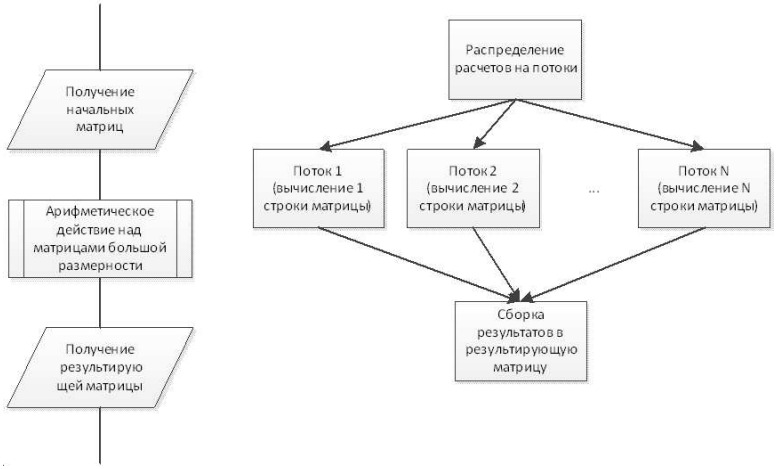

Рис. 3 – Алгоритм проведения параллельных расчетов над матрицами большой размерности

Результаты оптимизации

Результаты оптимизации можно визуально оценить на графиках загруженности ЦПУ и времени выполнения программы, которые получены с помощью стандартного профилирующего инструмента в Visual Studio 2010.

При сравнении графиков на рисунке сверху, визуально заметно, что время расчетов сократилось на несколько секунд (до 4 сек), нагрузка на процессор стала более равномерной, и не такой высокой, как до оптимизации. Можно визуально сделать вывод, что прирост производительности составил около 20%.

Выводы

Для повышения быстродействия программ, в которых используются расчеты с матрицами больших размерностей, целесообразно использовать методы: распараллеливание вычислительных процессов, оптимизации алгоритмов вычислений и использования специальных библиотек. Указанные методы применены на практике. Приведены результаты тестирования программы до и после оптимизации. Доказана целесообразность

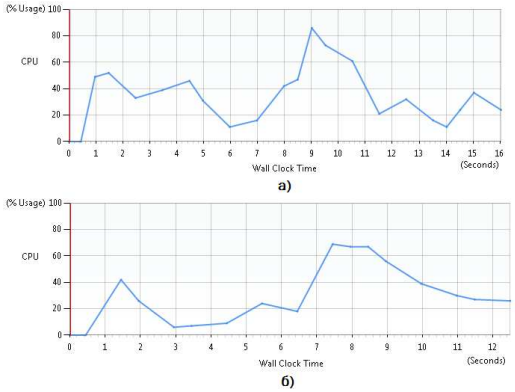

Рис. 4 – Графики зависимости нагрузки на процессор от времени а) до оптимизации и б) после оптимизации рассмотренными выше методами

применения разнообразных методов повышения быстродействия при работе с матрицами большой размерности.

Литература

1. Бурова И.Г., Демьянович Ю.К. Курс лекций “Алгоритмы параллельных вычислений и программирование”. - Санкт-Петербургский государственный университет.
2. Jeff Varszegi. How to write high-performance C# code. -“.NET developers journal”, SYS-CON Media, Inc., September 13,2004.
3. Воеводин В.В. Курс лекций “Параллельная обработка данных”. - Лаборатория параллельных информационных технологий НИВЦ МГУ, 2000.
4. Девид Каллахан. Вопросы проектирования для параллельного программирования. - “MSDN Magazine”, October, 2008.
5. Жуков І.А., Корочкін О.В. Навчальний посібник “Паралельні та розподілені обчислення”. - К.: “Корнійчук”, 2005.-226 с.

Отримано 06.03.2011 р.

 ISSN 1560-8956