УДК 519.688

ОБ ОДНОМ ПОДХОДЕ К АРИФМЕТИКЕ СВЕРХБОЛЬШИХ ПОЛИНОМОВ

© **А.Г. Поздникин**

Ключевые слова: полином на внешнем носителе; умножение полиномов; параллельный алгоритм.

Рассмотрен новый формат хранения во внешней памяти больших полиномов, которые не помещаются в оперативную память. Предлагается параллельный алгоритм умножения для таких полиномов.

1. Введение

Многочлены играют в символьных вычислениях исключительно важную роль [1]. От того, как организована библиотека полиномиальных процедур, во многом зависит эффективность системы компьютерной алгебры в целом.

Аналитические расчеты носят характер задач высокой вычислительной сложности, у которых происходит быстрый рост сложности вычислений при росте объемов входных данных. По этой причине для проведения аналитических вычислений требуется разработка параллельных алгоритмов и проведение расчетов на многопроцессорных вычислительных комплексах [2]. В статьях [3], [4] можно найти информацию о параллельных полиномиальных алгоритмах.

Традиционные системы компьютерной алгебры, такие как Mathematica, позволяют оперировать с полиномами, которые могут разместиться в оперативной памяти. Если же требуется оперировать с большими полиномами, которые не помещаются в оперативной памяти, эти системы оказываются непригодными.

Одной из систем, использующих для полиномиальной арифметики внешние носители информации, является система Form. Система Form предназначена для символьных манипуляций с очень большими алгебраическими выражениями и может применятся в различных областях науки и техники.

В работе [5] был описан формат хранения больших полиномов, которые не помещаются в оперативной памяти. В этой статье приводится описание нового формата хранения таких полиномов. Новый формат позволил реализовать более быстрые алгоритмы сложения и умножения больших полиномов.

2. Строение полинома на внешнем носителе

Для хранения одного полинома в оперативной памяти используется два одномерных массива. Первый массив хранит только ненулевые коэффициенты полинома, а второй массив хранит степени по каждой из переменных. Если в полиноме var переменных, то второй массив содержит в $vars$ раз больше элементов, чем первый. Мономы хранятся в полиноме в обратном лексикографическом порядке.

При операциях с большими полиномами и для пересылки их между процессорами необходимо оперировать с отдельными частями полиномов, результат произведения которых гарантированно помещается в оперативную память. Поэтому полином на внешнем носителе состоит из частей, каждая из которых приближенно равна заданному размеру $partLength$ в байтах. Константа $partLength$ выбирается из таких расчетов, что произведение частей полиномов размерами $partLength$ байт должно быть гарантированно вычислено в оперативной памяти. Сейчас константа $partLength$ выбирается эмпирическим путем, в дальнейшем планируется, что она будет определятся системой автоматически, в зависимости от архитектуры машины.

Для хранения такого полинома во внешней памяти будем использовать два файла. В первый файл записываются части полинома в виде массивов байт. Во втором файле будет содержатся:

1) тип коэффициента, т. е. числовое множество, из которого берутся коэффициенты полинома;

2) количество переменных полинома $(vars)$;

3) общее число ненулевых мономов в полиноме;

4) массив целых чисел, каждый элемент массива — это количество байт, которое занимает соответствующая часть полинома из первого файла в сумме с количеством байт, которое занимают предыдущие части полинома;

5) количество частей $(numparts)$, на которые был разбит исходный полином.

Для отличия полиномов, которые хранятся во внешней памяти, от обычных полиномов будем называть их *файловыми полиномами*.

3. Параллельный алгоритм умножения файловых полиномов

Процедура умножения файловых полиномов использует рекурсивный алгоритм. В основе рекурсивного алгоритма лежит дихотомическое деление полиномов на части.

Выход из рекурсии произойдет, когда исходные полиномы будут полностью разбиты на части, из которых состоят. Тогда операция умножения отдельных частей полиномов может быть выполнена в оперативной памяти.

Графом рекурсивного алгоритма является бинарное дерево. На листах дерева происходит умножение частей полиномов.

В корневой вершине задается интервал с номерами доступных процессоров. Параллельный алгоритм деления полиномов на части сопровождается делением на части интервала с номерами процессоров. Если свободные процессоры заканчиваются, а полином все еще не разбит на необходимое число частей, то выполняется вызов однопроцессорного рекурсивного алгоритма.

Рассмотрим параллельный алгоритм умножения файловых полиномов A и B. Граф алгоритма изображен на рис. 1.

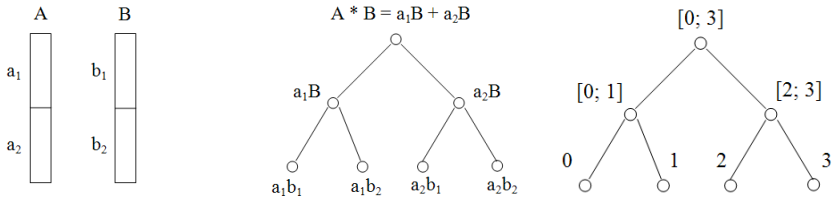

Рис. 1. Граф параллельного алгоритма умножения полиномов A и B и распределение четырех процессоров на узлах данного дерева.

Пусть $A = (a_1 + a_2)$ и $B = (b_1 + b_2)$ — два файловых полинома, которые требуется перемножить, a_1 и b_1 – части из которых состоит первый полином, a_2 и b_2 – части, из которых состоит второй полином. Каждая из частей a_1 и b_1 , a_2 и b_2 занимает на внешнем носителе приближенно $partLength$ байт.

Произведение может быть найдено в виде суммы четырех слагаемых: $a_1 * b_1 + a_1 * b_2 + a_2 * b_1 + a_2 * b_2$. При этом вычисление каждого из четырех слагаемых можно поручить отдельному процессору. Для этого ему нужно переслать соответствующие сомножители и затем выполнить их умножение.

Выбирается полином, который состоит из большего числа частей и делится на две части. В нашем случае оба полинома состоят из двух частей, так что делим любой из них пополам, пусть это будет полином A . На первом шаге делим полином A на две части a_1 и a_2 , т. е. $A = a_1 + a_2$. Интервал с номерами процессоров [0;3] делится на два интервала $[0; 1]$ и $[2; 3]$. Получим два узла, на которых необходимо выполнить операции умножения $a_1 * B$ и $a_2 * B$. Так как полином A полностью разбит на части, остается разбить на части полином B . Делим B на части b_1, b_2 . Так как мы полностью разбили на части исходные полиномы A , B , то вычисляем произведения $a_1 * b_1$, $a_1 * b_2$, $a_2 * b_1$, $a_2 * b_2$ в оперативной памяти, соответственно на процессорах 0, 1, 2, 3. При передаче вычисленных фрагментов обратно по направлению к корневой вершине будет производится их сложение. Вначале это сумма полиномов $a_1 * b_1 + a_1 * b_2 = a_1 * B$ и $a_2 * b_1 + a_2 * b_2 = a_2 * B$. Результатом умножения станет сумма $a_1 * B + a_2 * B = A * B$, вычисленная в корневой вершине.

4. Заключение

В предыдущей работе автора [5] был описан параллельный алгоритм умножения файловых полиномов, в котором предполагалась, что во время записи и чтения полинома, который хранится на диске, будут выполняться отдельные операции с каждым мономом. Кроме того, использовалась процедура, которая вычисляла приближенное значение количества байт, которое было необходимо для результата произведения двух полиномов.

В настоящей работе предлагается новый формат хранения полиномов, в котором исходный полином разбивается на крупные части, а не на мономы. Размер частей выбирается так, чтобы произведение двух частей можно было найти в оперативной памяти. Отпадает необходимость в процедуре оценки количества памяти для произведения двух полиномов. Кроме того, упростилась процедура пересылки файлового полинома от одного процессора к другому. Эксперименты выполнялись на кластере MVS100K Межведомственного Суперкомпьютерного Центра РАН. Результаты экспериментов будут представлены в докладе.

ЛИТЕРАТУРА

1. *Панкратьев Е.В.* Элементы компьютерной алгебры. Интернет-университет информационных технологий - ИНТУИТ.ру, БИНОМ. Лаборатория знаний, 2007. 248 с.

2. *Малашонок Г.И., Аветисян А.И., Валеев Ю.Д., Зуев М.С.* Параллельные алгоритмы компьютерной алгебры // Труды института системного программирования. 2004. Т. 8. П. 2. С. 169-180.

3. *Малашонок Г.И., Валеев Ю.Д.* Параллельные полиномиальные рекурсивные алгоритмы // Полиномиальная компьютерная алгебра: Международная конференция, Санкт-Петербург, ПОМИ РАН, 2008. С. 41-45.

4. *Валеев Ю.Д., Малашонок Г.И.* О формате полиномов для параллельных вычислений // Вестник Тамбовского университета. Серия Естественные и технические науки. Тамбов, 2004. Т. 9. Вып. 1. С. 149-150.

5. *Поздникин А.Г.* Параллельные полиномиальные вычисления с использованием внешней памяти // Вестник Тамбовского университета. Серия Естественные и технические науки. Тамбов, 2010. Т. 15. Вып. 4. С. 1426-1435.

БЛАГОДАРНОСТИ: Работа выполнена при финансовой поддержке АВЦП «Развитие научного потенциала высшей школы (2009-2010 годы)» (проект № 2.1.1/1853); темплана 1.12.09.

Поступила в редакцию 25 августа 2010 г.

Pozdnikin A. G. An approach to arithmetic of super-large polynomials.

New structure of out-of-core polynomials is discussed. The parallel algorithm for polynomial multiplication is presented.

Key words: out-of-core polynomials; polynomial arithmetic; parallel polynomial multiplication.