Параллельный алгоритм умножения чисел

М.А. Черепнёв

***Аннотация*--- Умножение по модулю, где модуль является *n*-значным числом, при *n* приблизительно равном 2^9, является в настоящий момент основной операцией в алгоритме дискретного логарифмирования. Если рассматривать указанный процесс как умножение многозначных чисел с последующим взятием по модулю (деление с остатком), то наибольшую сложность представляет последнее. Мы предлагаем взглянуть на этот вопрос по-другому. А именно, с точки зрения минимизации времени при неограниченном количестве вычислительных узлов. Сюда же входит задача распараллеливания на кристалле, где узлы, производящие вычисления, многочисленны, но маломощны, прежде всего по памяти, а связи между ними быстрые. Мы предлагаем для взятия по модулю алгоритм с предвычислениями, хранящимися распределенно. Эта же конфигурация позволит значительно сократить время на умножения с фиксированным множителем. Для самого умножения, применяется распараллеленный столбик. Время работы построенного алгоритма на распределенной вычислительной системе оценивается величиной O(log *n*), а с учетом конвееризации выполняется за один такт.**

***Ключевые слова* --- умножение, параллельные вычисления, предвычисления, конвееризация.**

I. Введение

Умножение по модулю M, где M - n-значное число, при n приблизительно равном 2^9, является основной операцией в алгоритме дискретного логарифмирования в адекватных на сегодня параметрах. Если рассматривать указанный процесс как последовательно выполняемые два процесса: умножение многозначных чисел с последующим взятием по модулю (деление с остатком), то наибольшую сложность представляет последний. Собственно умножение чисел может быть проделано многими разными способами от алгоритма Карацубы [1] до алгоритма с быстрым преобразованием Фурье [2]. Эти алгоритмы имеют субквадратичные оценки на число арифметических операций вплоть до линейных. Однако, чем меньше верхняя оценка сложности, тем труднее распараллеливать этот алгоритм. Это происходит из-за внутренних синхронизаций, то есть необходимости закончить все текущие вычисления перед очередным этапом алгоритма.

Мы предлагаем взглянуть на этот вопрос по-другому. А именно, с точки зрения уменьшения времени при неограниченном количестве вычислительных узлов. Сюда же входит задача распараллеливания на кристалле, где узлы, производящие вычисления, многочисленны, но маломощны, прежде всего по памяти, а связи между ними быстрые. Мы предлагаем для взятия по модулю алгоритм с предвычислениями, хранящимися распределенно. Для самого умножения, опишем распараллеленный столбик с $\log n$ синхронизациями как в известном методе Карацубы.

Модульное умножение по Монтгомери [3,4] сводится к двум умножениям чисел с длиной записи не менее n бит, если исходные числа находятся в некоторой стандартной форме. Приведение к этой форме и получение требуемого числа из этой формы также требуют одного умножения. Обозначим $M(n,m)$ --- сложность умножения n-битного числа на m-битное, $M(n) = M(n,n)$. В работе [5] алгоритм умножения по Монтгомери улучшается до оценки сложности $M(n/2)+2M(n,n/2)$ с последующей раздачей частей аргументов на $\theta_1\theta_2$ вычислительных узлов с оценкой общей сложности $2.5M(n/\theta_1, n/\theta_2)$. Некоторый обзор рассматриваемой тематики можно найти в работе [6] гл.2. Помимо алгоритмических, существуют и физические способы ускорения модульного умножения [7].

В этой заметке предлагается алгоритм модульного умножения n битных чисел для большого n. Этот алгоритм может быть использован для реализации арифметики в большом простом поле на многопроцессорной компьютерной технике.

Время работы рассматриваемого алгоритма будем считать в единицах времени, которое необходимо для сложения $t+\log n$ битных чисел (машинных слов) сторонним алгоритмом. Для простоты изложения будем считать, что t делит n, которое является степенью двойки, и $t \geq \log n$.

Символом log обозначаем логарифм по основанию 2. Пересылками в этой статье мы пренебрегаем.

II. Описание алгоритма

Пусть требуется перемножить два n-битных числа

$$a = \sum_{i=0}^{n-1} a_i 2^i, \quad b = \sum_{i=0}^{n-1} b_i 2^i,$$

по модулю n-битного числа M. Будем считать, что в нашем распоряжении имеется 1-я группа из n^2/t процессоров с оперативной памятью объёмом $t+\log n$ бит каждый и 2-я группа из n процессоров или просто кусков оперативной памяти объёмом n бит. Эта группа может быть составлена из процессоров 1-й группы. Алгоритм состоит из 2-х этапов: 1-й - умножение, 2-й - взятие по модулю.

1-й этап: Действуя методом умножения " в столбик ", складываем числа $2^j a$ при $b_j \neq 0$, вычисляя сумму t-разрядных отрезков, находящихся друг под другом при помощи сдваивания на отдельной подгруппе из $n/2$ процессоров 1-й группы. На это потребуется $\log n$ единиц времени. Общая арифметическая сложность этой части: $2n^2/t$ сложений $(t+\log n)$-битных чисел. Сумма, получающаяся при этом на каждой подгруппе, будет иметь длину $t+\log n$. Старшие $\log n$ бит $(j-1)$ -й подгруппы " головной " процессор j -й подгруппы прибавляет к своему результату, получая $t+1$ битное число. Для переноса

последнего лишнего бита (если он есть) каждой подгруппы повторим предыдущую операцию ещё не более 2-х раз. Всё это потребует дополнительно 3 единицы времени и $6n/t$ операций сложения t разрядных чисел. Итого $\log n + 3$ единиц времени.

Обозначим 2n разрядное полученное число $c = \sum_{i=0}^{2n-1} c_i 2^i$.

Будем предполагать, что в i-м куске оперативной памяти 2-й группы, $i=0,1, \dots, n-1$, содержится n-битное число --- наименьший положительный вычет $2^{n+i} \pmod M$, записанный в двоичной форме.

2-й этап: Суммирование

$$\sum_{i=0}^{n-1} c_{n+i}(2^{n+i} \pmod M) + \sum_{i=0}^{n-1} c_i 2^i \qquad (1)$$

осуществим описанным выше алгоритмом из первого этапа. Результат не будет превосходить $(n+1)2^n$. Результат запишем в c и повторим процедуру. Поскольку теперь в верхнем индексе первой суммы формулы (1) n заменено на $\log n$, получим результат, не превосходящий $2^n (1+\log n)$. После повторения этой процедуры в количестве, равном $(1+\lceil \log^* n \rceil)$, где $\log^* n$ - это итерационный логарифм n, то есть наименьшее такое k, что композиция k логарифмов, примененная к n, не превосходит ноль, получим либо ожидаемый результат, либо ожидаемый результат, увеличенный на M, так как $2^{n-1} \le M < 2^n$. Сравнивая с M и вычитая его, при необходимости, получим окончательный результат умножения по модулю M.

Общее время на системе из n^2/t процессоров оценивается величиной $(\log n+3)(\lceil \log^* n \rceil+2)$, а общая арифметическая сложность – величиной $(\lceil \log^* n \rceil+2)(2n^2+6n)/t$ сложений $t + \log n$ битных чисел. количество необходимых процессоров n^2/t.

Если алгоритм сдваивания осуществлять фиксированным n/K (а не $n/2$) числом процессоров, то, при оценке времени, полученной заменой $\log n$ на $K-2+\lceil \log(2n/K) \rceil \le K+[\log n/K]$, общее число используемых процессоров снижается до величины $2n^2/tK$. Здесь K слагаемых сдваивается до тех пор, пока не останется два слагаемых. Оставшиеся $2K$ слагаемых сдваиваются n/K процессорами параллельно как и выше. Итого $(K + [\log n/K]+ 3)(\lceil \log^* n \rceil+2)$ единиц времени.

Общая арифметическая сложность при этом не изменится.

В случае, когда b является l битным числом, $l < n$, не оптимизируя алгоритм сдваивания, аналогично получим, что общее время оценивается величиной

$$(\log l+3)(\lceil \log^* l \rceil+2),$$

а общая арифметическая сложность - величиной

$$(\lceil \log^* l \rceil+2)(2nl+6n)/t$$

Сложений $t+\log n$–значных чисел. Количество необходимых процессоров nl/t.

Для реализации конвеерной обработки на первом этапе, в каждой из $\log n$ единиц времени, выделенной на сдваивания, организуем вычисления на своей группе процессоров. В первую единицу времени сложение t-значных кусков осуществляют $n^2/2t$ процессоров, во вторую - $n^2/4t$ других процессоров, получивших результаты работы предыдущих и так далее. Переносы разрядов также осуществим на дополнительных вычислительных узлах. Количество процессоров при этом примерно удваивается по сравнению с реализацией, описанной выше. Второй этап преобразуем также с использованием удвоенного количества процессоров. Время работы всего алгоритма на системе из $2n^2/t$ независимых вычислителей, при потоковой подаче данных будет оцениваться временем на одно сложение t-битных чисел, то есть в принятых нами обозначениях --- единицей. Если уменьшить количество вычислителей до $n/2$ и обрабатывать на этой системе t-битные столбцы последовательно справа налево, то время будет оцениваться количеством столбцов, то есть $2n/t$.

III. Алгоритм умножения на константу.

Пусть теперь надо вычислить достаточно много произведений различных чисел на одно число:

$$ba_1, ba_2, \dots, ba_N. \qquad (2)$$

Сформируем массив n битных чисел $bc_i 2^{lj}$, $i=0,1, \dots, 2^l-1$, $j=0,1, \dots, \lceil n/l \rceil$, вычисленных по модулю M, где c_i пробегает все l разрядные числа (то есть меньшие 2^l). Это потребует $\lceil n/l \rceil 2^l$ единиц времени с учетом того, что каждое из них вычисляется за один такт на конвеерной системе, описанной выше, прибавляя предыдущее число или умножая его на 2^l.

Требуемая память оценивается величиной $n\lceil n/l \rceil 2^l$ бит. Затем двоичная запись каждого из чисел a_i разрезается на отрезки длины l. Зануляя цифры во всех отрезках кроме одного, получаем число, умножение на которое по модулю M уже вычислено. Так что вычисление ba_i при фиксированном i сводится к сложению не более чем $\lceil n/l \rceil$ n-разрядных чисел, которое можно произвести на нашей конвеерной системе за один такт. Таким образом, общее число единиц времени оценивается величиной

$$N+\lceil n/l \rceil 2^l.$$

При $n = 2^9$ и $N = 10^6 \approx 2^{20}$, что соответствует необходимым параметрам при решении задачи дискретного логарифмирования, получаем при $l=15$ равенство полученных слагаемых и общую линейную оценку времени работы алгоритма в $2N$ единиц.

IV. Заключение.

Конвееризация позволяет осуществить операции модульной арифметики с большими числами за один такт, однако при этом необходимое число независимых вычислительных узлов для построения конвеера оценивается величиной $2n^2/t$.

БИБЛИОГРАФИЯ

[1] Карацуба А.А., Офман Ю.П. Умножение многозначных чисел на автоматах.// ДАН СССР, 1962, т.145(2), с.293-294
[2] Ахо А., Хопкрофт Дж., Ульман Дж. Построение и анализ вычислительных алгоритмов. М.: МИР, 1979.
[3] Василенко О.Н. Теоретико-числовые алгоритмы в криптографии.// М., МЦНМО, 2003, 328с.
[4] Montgomery P.L. Modular multiplication without trial divizion.// Math. Comp., 1987, v.48(177), p.243-264
[5] P.Giorgi, L.Imbert, T.Izard Paramodular multiplication on multi-core processors/ IEEE Symp. on Comp. Arithm. Apr. 2013, Austin TX, US 135-142 https:// hal.archives-ouvertes.fr/hal-00805242
[6] R.P.Brent and P.Zimmermann Moder Computer Arithmetic.// Cambridg Univ. Press, 2010
[7] K.Nitta, N.Katsuta, O.Matoba A method for Modulo Operation by use of Spatial parallelism.// LNCS v.5172(2008), p.98-103

Статья получена 7 июля 2020.
Работа поддержана грантом РФФИ **19-01-00294**.
М.А.Черепнев из Московского государственного университета имени М.В.Ломоносова, РФ (e-mail: cherepniov@gmail.com)

Parallel number multiplication algorithm

M.A. Cherepniov

***Abstract* - Modulo multiplication, where the modulus is an n-digit number, with n approximately equal to 2^9, is currently the main operation in the discrete logarithm algorithm. If we consider this process as multiplication of big numbers followed by taking modulo (division with remainder), then the latter is the most difficult. We suggest looking at this problem in a different way. Namely, in terms of minimizing time with an unlimited number of computing nodes. This also includes the problem of parallelization on a chip, where the nodes that perform calculations are numerous, but low-power, primarily in memory, and the connections between them are fast. We propose an algorithm for modulo multiplication taking with precalculations stored distributed. The same configuration will significantly reduce the time for multiplication with a fixed multiplier. For the multiplication itself, a parallelized column algorithm is used. The running time of the constructed algorithm on a distributed computing system is estimated by the value O (log *n*), and taking into account the conveyor calculation system is performed in one clock cycle.**

***Key words*--- multiplication, parallel computations, precomputations, conveyor calculation system**

REFERENCES

[1] Karacuba A.A., Ofman Ju.P. Umnozhenie mnogoznachnyh chisel na avtomatah.// DAN SSSR, 1962, t.145(2), s.293-294
[2] Aho A., Hopkroft Dzh., Ul'man Dzh. Postroenie i analiz vychislitel'nyh algoritmov. M.: MIR, 1979.
[3] Vasilenko O.N. Teoretiko-chislovye algoritmy v kriptografii.// M., MCNMO, 2003, 328s.
[4] Montgomery P.L. Modular multiplication without trial divizion.// Math. Comp., 1987, v.48(177), p.243-264
[5] P.Giorgi, L.Imbert, T.Izard Paramodular multiplication on multi-core processors/ IEEE Symp. on Comp. Arithm. Apr. 2013, Austin TX, US 135-142 https:// hal.archives-ouvertes.fr/hal-00805242
[6] R.P.Brent and P.Zimmermann Moder Computer Arithmetic.// Cambridg Univ. Press, 2010
[7] K.Nitta, N.Katsuta, O.Matoba A method for Modulo Operation by use of Spatial parallelism.// LNCS v.5172(2008), p.98-103