ПАРАЛЛЕЛЬНЫЙ АЛГОРИТМ ПРОВЕРКИ ЧИСЛА НА ПРОСТОТУ

А.В. Кузнецова

Томский государственный университет

Во многих алгоритмах шифрования современных криптосистем используются простые числа большого размера, генерация которых является нетривиальной задачей. Один из методов решения данной задачи заключается в следующем: генерируется в качестве кандидата случайное число n подходящего размера, с помощью специальных алгоритмов выполняется его проверка на простоту, и, если число оказывается составным, то генерируется следующее число, и т.д. Для проверки числа на простоту известны две основные группы тестов: детерминированные тесты, применяемые только для чисел специального вида; вероятностные тесты, которые с некоторой вероятностью утверждают, что число простое, и точно определяют, что число составное, или наоборот.

В данной работе рассматривается вероятностный тест Миллера–Рабина. Если в результате выполнения этого теста утверждается, что число n – составное, то это действительно так, а если n – простое, то это верно с некоторой вероятностью.

Тест Миллера–Рабина основан на теореме Ферма, которая утверждает, что если n – простое число, то $a^{n-1} = 1 \bmod n$, где a – любое целое число ($1 < a < n - 1$). Обратное утверждение теоремы неверно. В тесте Миллера–Рабина для набора чисел $a_1, a_2, ..., a_m$, называемых основаниями, проверяется выполнение тождества $a_i^{n-1} = 1 \bmod n$ ($i = 1, ..., m$) и еще ряд дополнительных условий. Если для всех оснований тождество и условия выполняются, то число n объявляется простым с вероятностью ошибки меньшей, чем $(1/4)^m$, в противном случае число объявляется составным. Отметим, что чем больше число проверяемых оснований, тем меньше вероятность ошибки. Проверка условий теста может выполняться одновременно и независимо для нескольких оснований, что позволяет эффективно использовать параллельные вычисления. Следующий параллельный алгоритм ориентирован на использование многопроцессорной системы с распределенной памятью.

Пусть s – число процессов, соответствующее числу процессоров, задействованных в вычислениях; m – требуемое число оснований, причем оно при необходимости увеличивается до числа, кратного s. Тогда каждый из процессов должен протестировать заданное число n по $r = m/s$ основа-

ниям. Тестируемое число n подается на вход управляющему процессу, который рассылает его и число r остальным. Затем каждый из процессов случайным образом выбирает r оснований и выполняет для них проверку условий теста Миллера–Рабина. Если хотя бы один из процессов сообщает, что число n – составное, то n объявляется составным и все процессы завершаются. Если же результаты всех процессов таковы, что n – простое, то n объявляется простым с вероятностью ошибки $(1/4)^m$.

Для достижения наибольшего ускорения число процессоров рекомендуется брать кратным числу оснований. Однако даже при равном числе оснований проверка условий теста для различных оснований может выполняться не за одинаковое время, что приводит к простаиванию процессоров и снижает эффективность алгоритма. Время работы теста для одного основания зависит как от выбранного основания, так и от длины числа в его двоичном представлении.

Эксперименты, проведенные на кластерной системе, показали, что для небольших чисел (порядка 100 бит) ускорение не велико и эффективность использования системы падает с ростом числа процессоров. Причина в том, что время, затрачиваемое на передачу числа и сбор результатов, составляет большую долю от общего времени работы, т.е. относительно велико по сравнению со временем, затрачиваемым непосредственно на вычисления. Для чисел большого размера (порядка 256, 512, 1024 бит) время, требующееся для вычислений, возрастает, процент параллельных вычислений увеличивается и, как следствие, возрастает эффективность использования системы.

Таким образом, предложенный метод позволяет ускорить проверку простоты очередного сгенерированного случайного числа и при отрицательном ответе перейти к следующему.

Далее нами предполагается разрабатывать параллельные алгоритмы, которые специальным образом строят заведомо простые большие числа.