образом, современная модель параллелизма, основанная на потоках, является слишком низкоуровневой и плохо подходит для программирования человеком.

Решить данные недостатки призвана модель акторов. Разработанная в 1973 году Карлом Хьюиттом, она в качестве главных «персонажей» программы предлагает акторы, подобно тому, как в объектно-ориентированном программировании это место занимают объекты. Актор – это вычислительная сущность, способная выполнять в ответ на полученное сообщение, одно из трёх действий (или несколько - параллельно):

- отправление конечного числа сообщений другим акторам;
- создание конечного числа новых акторов;
- изменение своего поведения для получения следующего сообщения.

Главная особенность и фундаментальное достижение модели – развязка отправителя и сообщений.

Возможный образ думать о традиционной и акторной моделях – представлять первую как о телефонной системе (позволяет дозвониться только одному человеку в данный момент времени), а вторую – как электронную почту (имеется очередь сообщений, на которые получатель реагирует).

В процессе выполнения работы с целью лучшего ознакомления с возможностями и особенностями модели акторов было создано простейшее многопоточное приложение с её использованием. В качестве языка программирования был выбран язык Scala , разработанный в 2003 году под руководством Мартина Одерски и сочетающий преимущества объектно-ориентированного и функционального программирования. Выбор был обусловлен широкими возможностями распределённого программирования на Scala.

Библиотека Akka ещё больше расширяет возможности языка, предлагает средства продвинутого параллельного программирования и на данный момент является стандартом де-факто реализации модели акторов.

Созданное приложение имитирует принцип работы кафе: выполнение барменом и официантами заказов посетителей. Используются базовые приёмы модели – создание акторов, отправление и обработка сообщений, возможность наличия у акторов внутреннего изменяемого состояния, а также более продвинутые средства, например, иерархии акторов и обработка ошибок времени выполнения.

На основании опыта программирования приложения и экспериментов с ним был сделан вывод: программы, разработанные с применением концепции акторов, просты в создании и сопровождении, а также принципиально не могут содержать ошибок, вызванных гонками и взаимными блокировками. Всё это делает модель акторов весьма перспективной для разработки распределённых систем с высокими требованиями к производительности. Среди основных недостатков концепции: отсутствие прямой связи с объектно-ориентированной моделью, сложность определения требований по памяти до запуска, а также проблемы, связанные с асинхронным выполнением. Тем не менее, преимущества модели акторов с избытком компенсируют её немногочисленные недостатки.

Список использованных источников:

1. Таненбаум Э. Современные операционные системы / пер. с англ. СПб.: Питер, 2015. – 1120 стр.: ил.
2. John C. Mitchell. Concepts in programming languages. — Cambridge University Press, 2003. — 529 p.
3. Руководство по работе с языком Scala для занятых разработчиков Java-приложений // IBM developerWorks [Электронный ресурс]. — 2015. Режим доступа : http://www.ibm.com/developerworks/ru/library/j-scala04109/index.html — Дата доступа : 15.03.2015.

ОБЗОР МЕТОДОВ ФАКТОРИЗАЦИИ БОЛЬШИХ ЧИСЕЛ

Белорусский государственный университет информатики и радиоэлектроники
г. Минск, Республика Беларусь

Аксамит М. В. Былинович В. Н.

Стройникова Е. Д. – ассистент

В математике до сих пор не найден эффективный способ факторизации больших чисел. Существующие алгоритмы обладают рядом недостатков, и ни один из них не может дать стопроцентный результат для любого числа за достаточно короткое время.

Факторизация больших чисел - вычислительно сложная задача, которая лежит в основе широко известного алгоритма шифрования RSA. Этот алгоритм имеет два ключа: открытый, которым информация зашифровывается, и закрытый, который может расшифровать. Один из ключей известен, а он в свою очередь состоит из двух частей, одна из которых – произведение больших простых чисел (обозначается n), с помощью которых генерировались оба ключа. Следовательно, если факторизовать число n, то шифр будет взломан.

В зависимости от сложности алгоритмы факторизации можно разделить на две группы: экспоненциальные и субэкспоненциальные алгоритмы.

В данной работе были рассмотрены три экспоненциальных алгоритма: перебор делителей, метод факторизации Ферма и *ρ*-алгоритм Полларда, а также была разработана программа для реализации *ρ*-алгорим Полларда.

Перебор делителей

Сложность: $O(N^{1/2})$.

Последовательное деление факторизуемого числа *N* на натуральные числа от 2 до $[N^{1/2}]$.

Метод факторизации Ферма

Сложность $O(\exp(N))$.

Поиск таких чисел *A* и *B*, что факторизуемое число *N* можно представить в виде: $N = A^2 – B^2 = (A + B)(A - B)$.

Алгоритм:

- 1. Находим квадратный корень числа *N*: $m = N^{1/2}$;
- 2. Для $x = \{1, 2, 3…\}$ вычисляем значение $q(x) = (m+x)^2 – N$ до тех пор, пока из очередного q(x) нельзя будет извлечь корень без остатка.
- 3. $A = (q(x))^{1/2}$, $B = m + x$, $n=(A+B)(A-B)$, $p=A+B$, $q=A-B$.

ρ-алгорим Полларда

Сложность: $O(N^{1/4})$.

Составляем рад $x_i = f(x_i\text{-}1) \pmod N$, (обычно за x_0 берут число 2), $i = \{1, 2, …\}$. На каждом шаге вычисляем d = НОД($|x_i – x_j| \pmod N$, N), $j = \{1,2,…\} < i$. Если $d > 1$, то d – делитель числа N.

Теорема: Пусть $\lambda > 0$. Для случайной выборки из l+1 элементов, каждый их которых меньше q, где $l = (2\lambda q)^{1/2}$, вероятность того, что два элемента окажутся одинаковыми $p > 1 – \exp(-\lambda)$.

Применение к алгоритму:

Имеем две последовательности: $\{u_n\}$, где $u_i = |x_i|$, и $\{z_n\}$, где $z_i = u_i \bmod q$, q – делитель числа N. Все члены $\{z_n\}$ меньше $N^{1/2}$. Если рассмотреть эту последовательность как случайную, то согласно теореме, вероятность того, что среди $l + 1$ её членов попадутся два одинаковых, превысит 50% при $\lambda \approx 0.69$, тогда l должно быть не меньше $(2\lambda q)^{1/2} \approx (1.4q)^{1/2} \approx 1.18 * (q)^{1/2}$.

Вот мы получили эти два одинаковых члена: $z_i = z_j$, тогда $x_i – x_j \equiv 0 \pmod q$. Следовательно, $|x_i – x_j|$ и N делятся на q. Остается только найти q как НОД $(N, |x_i – x_j|)$. Поскольку $(q)^{1/2} \le n^{1/4}$, то с вероятностью, превышающей 50%, делитель N будет найден за $1.18 * N^{1/4}$ итераций.

При оценке эффективности работы данных алгоритмов на практике можно сделать следующие выводы:

Перебор делителей: Эффективен для факторизации небольших чисел. Сразу дает простые множители. Работает очень медленно для больших чисел.

Метод Ферма: Эффективен для чисел, состоящих из больших множителей, но он может работать очень медленно, даже хуже, чем перебор делителей, если у числа есть маленькие делители. Именно поэтому использовать данный метод следует лишь после запуска остальных методов. Также полученные множители могут оказаться составными числами.

ρ-алгоритм Полларда: Является одним из наиболее эффективных алгоритмов факторизации. Полученные множители нужно дополнительно проверять на простоту, однако, данный алгоритм один из наиболее подходящих для факторизации больших чисел с множителями различной величины.

Список использованных источников:

1. Ишмухаметов Ш. Т. Методы факторизации натуральных чисел. / Ишмухаметов Ш. Т. // Уч. метод. пособие для студентов старших курсов факультета вычислительной математики и кибернетики. – Казань, 2011. – 190 с.
2. Василенко О. Н. Теоретико-числовые алгоритмы в криптографии. / Василенко О. Н. // Уч. Пособие для студентов старших курсов и аспирантов математических факультетов ВУЗ-ов, а также для специальстов, желающих ознакомиться с последними достижениями в данной области. – МЦНМО, 2003 г. – 328 с.