МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
ИЗВЕСТИЯ ВЫСШИХ УЧЕБНЫХ ЗАВЕДЕНИЙ
ПРИБОРОСТРОЕНИЕ
ИЗДАНИЕ САНКТ-ПЕТЕРБУРГСКОГО НАЦИОНАЛЬНОГО ИССЛЕДОВАТЕЛЬСКОГО УНИВЕРСИТЕТА ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ
Журнал издается с января 1958 г.

ТОМ 55 **ОКТЯБРЬ 2012** **№ 10**

СПЕЦИАЛЬНЫЙ ВЫПУСК

К 75-ЛЕТИЮ КАФЕДРЫ ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ НИУ ИТМО

Под редакцией доктора технических наук, профессора Т. И. Алиева и доктора технических наук, профессора А. А. Ожиганова

СОДЕРЖАНИЕ

ПРЕДИСЛОВИЕ 5

ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ

Бессмертный И. А. Управление контекстом в информационных системах 7

Косяков М. С., Шинкарук Д. Н., Торопов А. В., Шполянский Ю. А. Применение технологии CUDA для ускорения расчета цен опционов европейского типа сеточным методом 13

Шинкарук Д. Н., Шполянский Ю. А., Косяков М. С. Анализ эффективности применения технологии CUDA для решения систем линейных уравнений с трехдиагональными матрицами в задачах расчета цен опционов 20

Рубина И. С., Тропченко А. Ю. Исследование алгоритмов кодирования преобразованием в рамках задач сжатия опорных и остаточных кадров видеопоследовательности 26

Тропченко А. А., Тропченко А. Ю. Нейросетевые методы идентификации человека по изображению лица 31

Щеглов К. А., Щеглов А. Ю. Модель контроля доступа к создаваемым файловым объектам 37

Поляков В. И., Скорубский В. И. Преобразование моделей алгоритмов 41

КОМПЬЮТЕРНЫЕ СИСТЕМЫ И СЕТИ

Алиев Т. И., Соснин В. В., Шинкарук Д. Н., Тихонов М. Ю., Бурмакин Н. Г. САПР маршрутизируемой компьютерной сети на основе компонентов с открытыми исходными кодами 47

Богатырев В. А., Богатырев С. В., Богатырев А. В. Функциональная надежность вычислительных систем с перераспределением запросов 53

Алиев Т. И. Задачи синтеза систем с потерями 57

Муравьева-Витковская Л. А. Обеспечение качества обслуживания в мультисервисных компьютерных сетях за счет приоритетного управления 64

ПРОГРАММНО-АППАРАТНЫЕ СРЕДСТВА ИНФОРМАЦИОННО-УПРАВЛЯЮЩИХ СИСТЕМ

Ожиганов А. А. Композиционные кодовые шкалы для преобразователей линейных перемещений 69

Болгаров И. С., Маковецкая Н. А., Платунов А. Е., Постников Н. П. Проектирование приборных контроллеров 73

Николаенков А. В. Аспектные технологии в системах с преобладающей программной компонентой 78

Ожиганов А. А., Захаров И. Д. Система автоматизированного проектирования псевдорегулярных кодовых шкал 80

Чураев С. О. Метод реверсивной случайной выборки для измерения с пикосекундным разрешением времени задержки в элементах интегральных схем 84

SUMMARY (*перевод Ю. И. Копилевича*) 89

SPECIAL ISSUE

TO 75th ANNIVERSARY OF THE DEPARTMENT OF COMPUTER TECHNOLOGY NRU ITMO

By Edition of T. I. Aliev, Doctor of Technical Science, Professor,
A. A. Ozhiganov, Doctor of Technical Science, Professor

CONTENTS

PREFACE 5

INFORMATION TECGNOLOGIES

Bessmertny I. A. Context Management in Information Systems 7

Kosyakov M. S., Shinkaruk D. N., Toropov A. V., Shpolyanskiy Yu. A. Application of CUDA Technique to Quicken Calculation of European Option Prices by Finite-Difference Method 13

Shinkaruk D. N., Shpolyanskiy Yu. A., Kosyakov M. S. Analysis of CUDA Efficiency in Solving Linear Tridiagonal Systems for Theoretical Option Pricing 20

Rubina I. S., Tropchenko A. Yu. Study of Transform Coding Algorithms in Compression of Video Sequence Frames 26

Tropchenko A. A., Tropchenko A. Yu. Neural Network Methods of Person Identification by Face Image 31

Shcheglov K. A., Shcheglov A. Yu. Model of Control over Access to Newly Created File Objects 37

Polyakov V. I., Skorubsky V. I. Conversion of Algorithm Models 41

COMPUTER SYSTEMS AND NETWORKS

Aliev T. I., Sosnin V. V., Shinkaruk D. N., Tikhonov M. Yu., Burmakin N. G. Creating a CAD Simulator of a Routed Computer Network using Open Source Components 47

Bogatyrev V. A., Bogatyrev S. V., Bogatyrev A. V. Functional Reliability of Computing Systems with Redistribution of Inquiries 53

Aliev T. I. Problems of Synthesis of Systems with Losses 57

Muravyeva-Vitkovskaya L. A. Priority-Based Mechanisms of Service Quality Provision in Multiservice Computer Networks 64

SOFTWARE AND HARDWARE TOOLS OF INFORMATION MANAGEMENT SYSTEMS

Ozhiganov A. A. Composite Code Scales for Converters of Linear Movements 69

Bolgarov I. S., Makovetskaya N. A., Platunov A. E., Postnikov N. P. Design of Instrument Controllers 73

Nikolaenkov A. V. Aspect Technologies in Development of SW-Intensive Systems..... 78

Ozhiganov A. A., Zakharov I. D. A Computer-Aided System for Pseudo-Regular Code Scale Design..... 80

Churayev S. O. Method of Reverse Random Sampling for Measurement of Time Delay in Element of the Integrated Circuit with Picosecond Accuracy 84

SUMMARY..... 89

Editor-in-Chief E. B. Yakovlev

ПРЕДИСЛОВИЕ

Созданная в ноябре 1937 г. кафедра счетно-решающих приборов (с 1963 г. кафедра вычислительной техники) с самого начала была ориентирована на подготовку специалистов в области разработки и исследования математических и счетно-решающих приборов различного назначения. Первые научно-исследовательские работы, выполнявшиеся на кафедре, были связаны с созданием тренажеров для военно-морского флота и цифровых интеграторов, проектированием и изготовлением комплексов аналоговой вычислительной аппаратуры для определения и регистрации ходовых качеств судов.

В середине 1950-х гг. на кафедре началось проектирование первой в вузе цифровой вычислительной машины „ЛИТМО-1“ для оптических расчетов, которая была запущена в эксплуатацию в 1960 г. В 1964 году на кафедре была изготовлена электронная цифровая вычислительная машина „ЛИТМО-2“, предназначенная для выполнения теплофизических расчетов. В 1962 г. при кафедре была организована отраслевая лаборатория цифровых вычислительных управляющих машин, в которой были развернуты работы по созданию методов проектирования цифровых вычислительных устройств и разработке преобразователей аналоговых сигналов в цифровые. В лаборатории проводились исследования по алгоритмизации процессов проектирования с целью создания базы для использования универсальных ЭВМ при проектировании вычислительных машин и устройств с применением методов статистического и имитационного моделирования для анализа структур и алгоритмов на этапах логического и операционного проектирования.

В 1965 г. на базе ЭЦВМ „Минск-2“ при кафедре был организован вычислительный центр, в котором проводились работы по автоматизации программирования и алгоритмизации задач приборостроительного проектирования. В этот же период стали формироваться новые направления научных исследований, работы по которым ведутся на кафедре до настоящего времени: разработка моделей и методов исследования вычислительных систем и сетей различных классов; разработка методов преобразования, хранения и обработки информации с использованием оптических принципов; разработка принципов и методов построения преобразователей перемещений на основе рекурсивных кодовых шкал повышенной информационной надежности; разработка программно-алгоритмических и аппаратных средств для цифровой обработки изображений; работы в области микропроцессорной техники и информационно-управляющих систем, а также проектирования, разработки, сопровождения и реинжиниринга корпоративных информационных систем.

С середины 1990-х гг. на кафедре ведутся работы в области информационной безопасности и защиты информации, а также в сфере интеллектуальных информационных систем.

В настоящем сборнике представлены результаты научно-исследовательских работ, выполняемых на кафедре вычислительной техники НИУ ИТМО.

Заведующий кафедрой
вычислительной техники НИУ ИТМО,
Заслуженный работник высшей школы РФ,
доктор технических наук, профессор Т. И. АЛИЕВ

PREFACE

Department of Calculating Devices established in November 1937 (starting from 1963 — Department of Computer Technology) from the outset has been oriented to training of specialists in the field of development and study of mathematical and calculating devises of various purposes. The first scientific research works carried out at the Department were related to creation of training simulators for the Navy and digital integrators, development and manufacturing of analog calculating complexes for measurement and registration of running characteristics of vessels.

In the mid-1950s, development of LITMO-1 computer was started up; it was the first computer in the Institute designed for optical calculations. In 1964 the digital electronic computer LITMO-2 was manufactured at the Department for thermal physical calculations. In 1962 the applied-research laboratory of digital control computers was established at the Department, the laboratory carried out researches in development of methods for design of digital calculating devices and analog-to-digit converters. The laboratory was also involved in studies on algorithmization of design processes aimed at creation of the base for application of universal electronic computers in design of calculating machines and devices with the use of methods of statistical and imitation modeling for analysis of structures and algorithms at the stages of logical and operational projection.

In 1965, Computing Center was set up at the Department on the base of digital electronic computer Minsk-2. The Center carried out researches in programming automation and algorithmization of instrument-making design problems. Several new lines of investigation actual today were formulated in that period, including development of models and methods for analysis of computer systems and networks of various classes, development of methods of information conversion, storage, and processing with the use of optical approaches; development of principles and methods of conversion of movement conversion on the base of recursion cod scales of higher informational reliability; development of software and hardware tools for digital image processing; researches in micro-processing technique and information managing systems; development, design, and re-engineering of corporate information systems.

Starting from mid-1990, the Department has been carrying out researches in the field of information security and data protection, as well as in intellectual information systems.

This issue presents results of scientific works which are currently carried out at the Department of Computer Technology, NRU ITMO.

Taufik I. ALIEV,
Doctor of Technical Sciences, Professor,
Honored Worker of higher education of the Russian Federation,
Head of the Department of Computer Technology, NRU ITMO

ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ

УДК 004.89

И. А. Бессмертный

УПРАВЛЕНИЕ КОНТЕКСТОМ В ИНФОРМАЦИОННЫХ СИСТЕМАХ

Исследована проблема поиска данных в информационных системах, построенных на принципах Semantic Web. Обоснована необходимость использования данных контекста в поисковых запросах, предложена концептуальная модель интеллектуального агента, осуществляющего поиск с автоматическим управлением контекстом.

Ключевые слова: *информационная система, контекст, онтология, информационный поиск.*

Допущение открытого мира и поиск в базах знаний. Системы управления базами данных предоставляют возможность абстрагироваться от файловой системы, индексации и пр. при программировании приложений. При этом остается невозможным доступ на уровне языка SQL для конечного пользователя, поскольку для успешного поиска в базах данных (БД) при составлении SQL-запросов помимо владения языком SQL требуется знание структуры данных. Это означает, что запрос к БД всегда выполняется в заранее определенном и хорошо известном программисту контексте. Появление новых сущностей или отношений требует перепроектирования БД. Отсутствие в БД данных о факте автоматически означает его отрицание, т.е. в БД реализуется допущение замкнутого мира.

В отличие от БД в базах знаний (БЗ) отсутствуют заранее определенные модели данных, а разные фрагменты БЗ могут храниться на распределенных сетевых ресурсах, объединяемых в виде семантической сети во время выполнения поискового запроса. Кроме того, в БЗ обычно применяется допущение открытого мира, при котором отсутствие данных о факте не означает отрицания факта. Указанные свойства БЗ приводят к тому, что вычисление любых, даже очевидных, фактов может потребовать глобального поиска, сложность которого будет несоразмерна ценности полученного результата.

Из первых двух свойств вытекает третье, связанное с формированием запросов к БЗ. Аналогом инфологической модели БД для баз знаний является онтология предметной области. Поисковый запрос к БЗ должен базироваться на онтологии, а интерпретация онтологии представляет собой сложную задачу. Между тем допущение открытого мира означает необходимость неограниченного расширения пространства поиска, а значит и автоматической интерпретации онтологий. В рамках настоящего исследования предполагается, что поиск выполняется в документах, подготовленных на основе одной и той же онтологии.

Роль контекста в задаче поиска. Извлечение знаний из конкретного документа, созданного на основе известной онтологии, практически не отличается от обработки SQL-запросов к БД. Если документ не задан в запросе, возникает задача его поиска. Поиск с помощью серверов, ориентированных на использование человеком (Google, Yahoo, Yandex и др.), бесполезен,

поскольку теги HTML и любой другой разметки не индексируются, в то время как знания формализуются с применением разметки. В последнее время появился ряд сервисов, таких как SWOOGLE <http://swoogle.umbc.edu/>, SWSE <http://swse.org/> и др., позволяющих осуществлять поиск формализованных данных. Необходимость извлечения документов интеллектуальным агентом, в том числе с использованием таких поисковых сервисов, делает актуальной задачу автоматической оценки релевантности найденных документов запросу.

При написании документа всегда предполагается, что читатель знаком с его контекстом. Если название документа не полностью отражает содержание, то оно уточняется во вводной части. В лингвистике контекст есть фрагмент текста минус определяемая единица [1]. Таким образом, контекст $\xi(m)$ для сообщения m устанавливается как

$$\xi(m) = <C, L, P, I, V> - m,$$

где C — множество классов объектов, L — множество отношений или предикатов (связей) между объектами, P — множество свойств объектов, I — множество экземпляров объектов, V — множество значений. Следовательно, для каждого последующего m_{i+1} сообщения предыдущее m_i включается в состав контекста:

$$\xi(m_{i+1}) = \xi(m_{i-1}) + m_i.$$

Данное определение является антропоморфным, ориентировано на речевое взаимодействие и предполагает, что субъект и объект такого взаимодействия обладают интеллектом и могут идентифицировать (установить) контекст для каждого коммуникационного акта. В случае информационного поиска установление контекста может быть простым только для локальных БЗ, как, например, это делается с помощью микротеорий в продукте ResearchCyc компании Cycorp <www.cyc.com>. Автор запроса к БЗ должен знать, каким образом факты в базе знаний группируются в микротеории, и эксплицитно указывать идентификатор микротеории в запросе. Необходимость владения микротеориями, идентификаторами элементов БЗ, а также специальным языком запросов делает взаимодействие с такой БЗ подобным работе с БД, которая доступна только для программиста, а не конечного пользователя.

В настоящей работе под контекстом будем понимать множество понятий предметной области, которое должно быть общим для всех участников информационного обмена и позволять обмениваться короткими сообщениями. Другое назначение контекста — ограничение предметной области, позволяющее сократить размерность задачи поиска и избежать противоречивости фактов. Обычно контекст задается в начале коммуникационного акта и при необходимости может уточняться. Таким образом, для правильной интерпретации сообщений все участники обмена информацией должны владеть контекстом. Визуализация знаний также требует использования контекста, как это сделано, в частности, в разработанной автором программе Semantic [2, 3], предназначенной для создания БЗ, извлечения и визуализации знаний.

Концептуальная модель агента для поиска в базах знаний. В соответствии с концепцией Semantic Web [4] извлечение знаний осуществляется интеллектуальными агентами, которые самостоятельно отыскивают требуемую информацию, формулируя при необходимости запросы к другим агентам. В отличие от SQL-запросов к БД (SELECT … FROM … WHERE …) запросы к БЗ, в частности на языке SPARQL, не содержат конструкции FROM, поскольку запрос всегда выполняется в пределах целого документа. Поиск документа должен выполняться по контексту запроса. Это означает, что извлечение фактов из Semantic Web представляет собой двухэтапный процесс (рис. 1).

Поскольку точное совпадение контекста документа и контекста запроса является идеальным случаем, данный процесс не всегда завершается успешно и может итеративно повторяться.

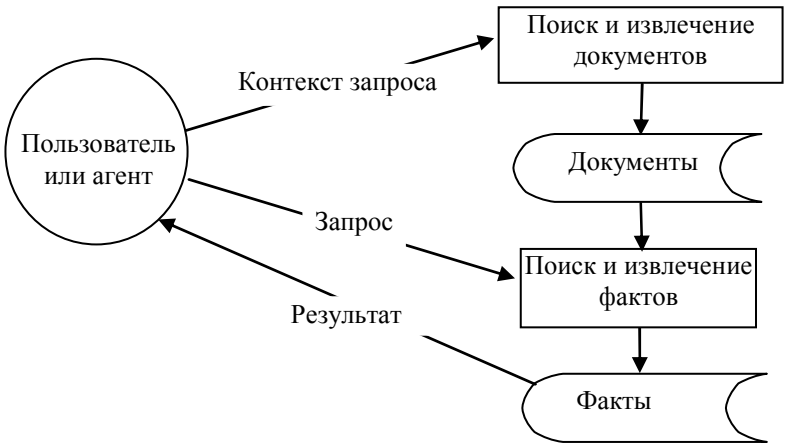


Рис. 1

На рис. 2 показаны все возможные сочетания контекста запроса Q и контекста предметной области (домена) D:

a) запрос породил набор фактов, ни один из которых не является релевантным (отсутствие решений);

b) найдены часть релевантных фактов и некоторое количество нерелевантных (решение есть, но оно неполное и противоречивое);

c) найдены все релевантные факты и часть нерелевантных (решение полное, но противоречивое);

d) найдена часть релевантных фактов и при этом нет нерелевантных (решение непротиворечивое, но неполное);

e) найдены все релевантные факты и ни одного нерелевантного (целевое состояние: решение полное и непротиворечивое).

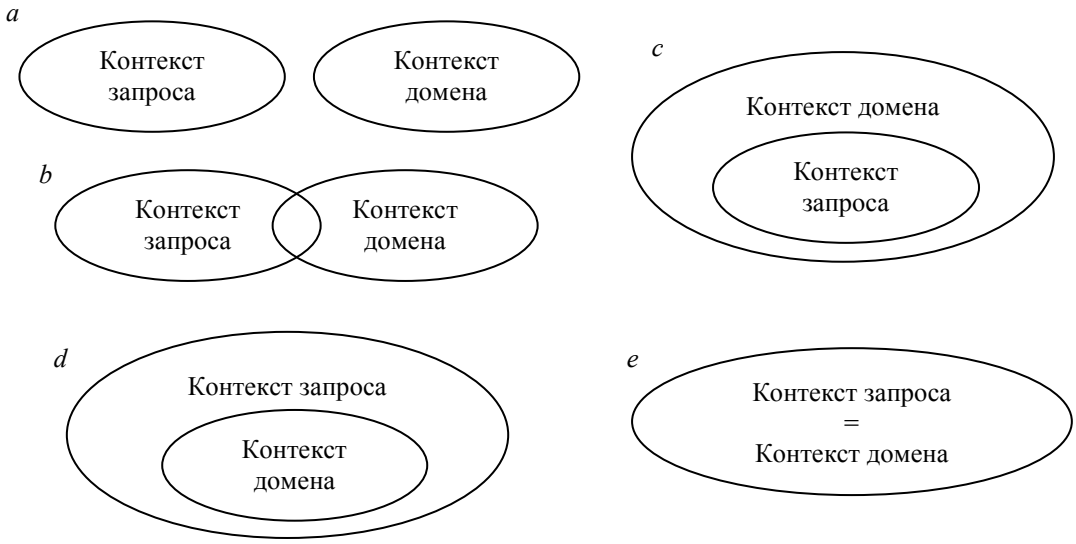


Рис. 2

В таблице приведены условия появления данных сочетаний и признаки, с помощью которых можно идентифицировать каждое из состояний.

Идентификатор	Решение			Условие
	наличие	полнота	противоречивость	
a	0	0	0	$Q \cap D = \varnothing$
b	1	0	1	$Q \cap D \neq \varnothing \,\&\, Q \cap D \neq Q \cup D$
c	1	1	1	$Q \subset D$
d	1	0	0	$Q \supset D$
e	1	1	0	$Q = D$

Если запрос не привел к состоянию *e*, следует изменить запрос. Ограничим модификации запросов двумя операциями: *x* — обобщение запроса (расширение контекста) и *y* — уточнение запроса (сужение контекста). Граф на рис. 3 отображает конечный автомат, в котором разрешенными являются только переходы, приближающие к целевому состоянию *e* (не отдаляющие от *e*).

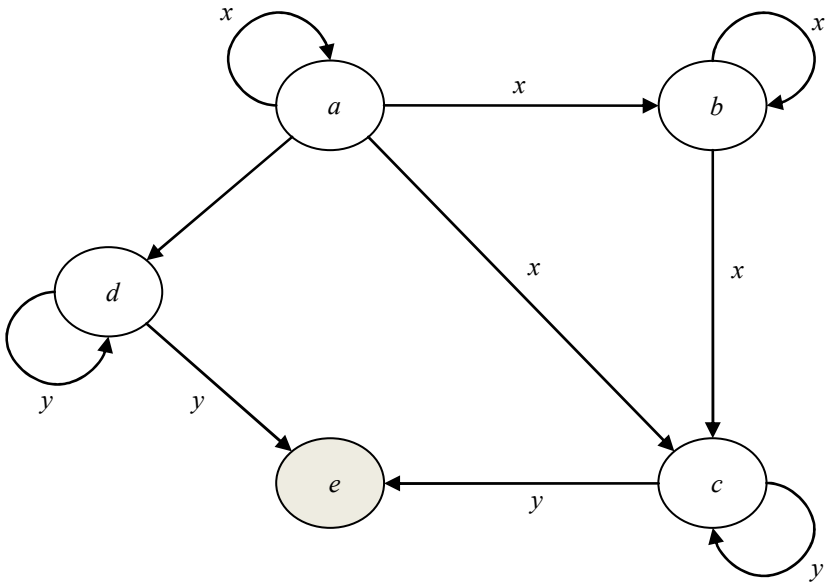


Рис. 3

Расширение пространства поиска может привести к увеличению времени извлечения фактов за счет применения правил, и это обстоятельство требует исследования.

Исследование производительности поиска в расширяющемся контексте. Пусть N — число фактов в базе знаний, τ — среднее время доступа к одному факту, A_i — число атрибутов или классов, релевантных i-му условию запроса, $i=\overline{1, n}$. При обработке запроса первое условие применяется в среднем к $A_1/2$ атрибутам (классам), второе — если первое выполнено, а вероятность данного события $p_1=1/A_1$ и т.д. Заметим, что речь идет не о ветвлении поиска на дереве решений, а о фильтрации фактов, поэтому здесь нет экспоненциального роста сложности поиска.

Таким образом, среднее время T обработки запроса на извлечение фактов для точно определенного контекста составит

$$T = \tau N\left(1+\frac{A_1}{2}+p_1\frac{A_2}{2}+p_1p_2\frac{A_3}{2}+\ldots+p_1p_2\ldots p_{n-1}\frac{A_n}{2}\right)=$$

$$=\tau N\left(1+\frac{A_1}{2}+\frac{A_2}{2A_1}+\frac{A_3}{2A_1A_2}\ldots+\frac{A_n}{2A_1A_2\ldots A_{n-1}}\right)=\tau N\left(1+\frac{1}{2}\sum_{i=1}^{n}\frac{A_i}{\prod_{k=1}^{i-1}A_k}\right).$$

Для $A=A_1=A_2=\ldots=A_n$ при $n\geq 2$

$$T = \tau N\left(1+\frac{A}{2}+\frac{1}{2}+\frac{1}{2A}+...+\frac{1}{2A^{n-1}}\right)\approx \tau N\frac{(A+3)}{2},$$

если пренебречь малыми значениями членов ряда, заключенных в скобки.

В случае поиска в расширенном контексте для каждого условия запроса из базы знаний должны извлекаться факты с близкими значениями атрибутов. Пусть δ_i — число допустимых значений i-го атрибута. При этом для каждого атрибута в условии запроса число извлекаемых фактов удваивается, поскольку требуются дополнительные факты, характеризующие близость значений атрибута. Тогда среднее время T_x обработки запроса на извлечение фактов для расширенного контекста будет

$$T_x = \tau N\left(1+A_1+\frac{\delta_1 A_2}{A_1}+\frac{\delta_1\delta_2 A_3}{A_1A_2}+...+\frac{\delta_1\delta_2...\delta_{n-1}A_n}{A_1A_2...A_{n-1}}\right)=\tau N\left(1+A_1+\sum_{i=1}^{n}A_i\prod_{j=1}^{n-1}\frac{\delta_j}{A_j}\right).$$

Для случая, когда $A=A_1=A_2=...=A_n$ и $\delta=\delta_1=\delta_2=\ldots=\delta_n$, при $n\geq 2$,

$$T_x = \tau N\left(1+A+\delta+\frac{\delta^2}{A}+...+\frac{\delta^{n-1}}{A^{n-2}}\right)\approx \tau N(1+A+\delta),$$

если также пренебречь малыми значениями членов ряда.

Если контекст расширяется до целого класса, то для i-го условия запроса атрибут может принадлежать одному из C_i классов. Первое условие запроса порождает в среднем A_1+C_1 обращений к базе знаний. Вероятность успешного выполнения первого условия $p_1=1/C_1$. Тогда среднее время T_c обработки запроса на извлечение фактов в пределах класса для каждого из условий запроса составит

$$T_c = \tau N\left(1+A_1+C_1+\frac{A_2+C_2}{C_1}+\frac{A_3+C_3}{C_1C_2}+...+\frac{A_n+C_n}{C_1C_2...C_{n-1}}\right)=\tau N\left(1+\sum_{i=1}^{n}\frac{A_i+C_i}{\prod_{j=1}^{i-1}C_j}\right).$$

Для случая, когда $A=A_1=A_2=\ldots=A_n$ и $C=C_1=C_2=\ldots=C_n$,

$$T_c = \tau N\left(1+A+C+\frac{A}{C}+1+\frac{A}{C^2}+\frac{1}{C}+...+\frac{A}{C^n}+\frac{1}{C^{n-1}}\right).$$

Экспериментальное исследование скорости извлечения фактов производилось в среде SWI-Prolog на синтетической базе фактов, фрагмент которой представлен ниже.

```
t(1, isa, matematician).          t(1,lives_in,denmark).
t(2, isa, player).                t(2,lives_in,korea).
t(3, isa, restorer).              t(3,lives_in,mexico).
t(4, isa, graver).                t(4,lives_in,spain).
t(5, isa, informatics).           t(5,lives_in,usa).
t(6, isa, conductor).             t(6,lives_in,switzerland).
…
t(painter, isa, artist).          t(painter, id, 1).
t(graphic_artist, isa,artist).    t(graphic_artist, id, 2).
…
t(matematician, isa,scientist).   t(matematician, id, 13).
…
t(conductor, isa, musician).      t(conductor, id, 23).
t(singer, isa, musician).         t(singer, id, 24).

t(korea, isa, asia).              t(korea, id, 1).
t(china, isa, asia).              t(china, id, 2).
…
t(england, isa, europe).          t(england, id, 12).
t(usa, isa, america).       t(usa, id, 13).
```

Базу составляют сгенерированные случайным образом факты о субъектах, идентифицируемых числами и имеющих атрибуты „профессия“ и „страна“. Экземпляры группируются в классы (континент, ученый, художник, инженер, …). Кроме того, атрибуты имеют численные идентификаторы, присвоенные таким образом, чтобы близкие значения идентификаторов соответствовали близким профессиям или соседним странам. В точном запросе выбирались все представители конкретной профессии, проживающие в конкретной стране, в расширенном запросе — профессии и страны, имеющие смежные идентификаторы, а в поиске по классам — субъекты, представляющие класс профессий и континент.

На рис. 4 представлены теоретические результаты (цифры без штриха) и данные, полученные с помощью экспериментов (цифры со штрихом) для поиска в точно заданном контексте *1*, а также поиска в ближайшем окружении заданного контекста *2* и в классе, объединяющем все контексты уровнем выше *3*. Приведенные результаты демонстрируют, во-первых, хорошее совпадение теоретических результатов с экспериментами, во-вторых, линейный рост сложности поиска при увеличении числа фактов в базе данных, в-третьих, заметно большее время поиска в ближайшем окружении заданного контекста по сравнению с поиском в пределах целого класса. Последнее обстоятельство объясняется тем, что при поиске в целом классе делается меньше проверок условий.

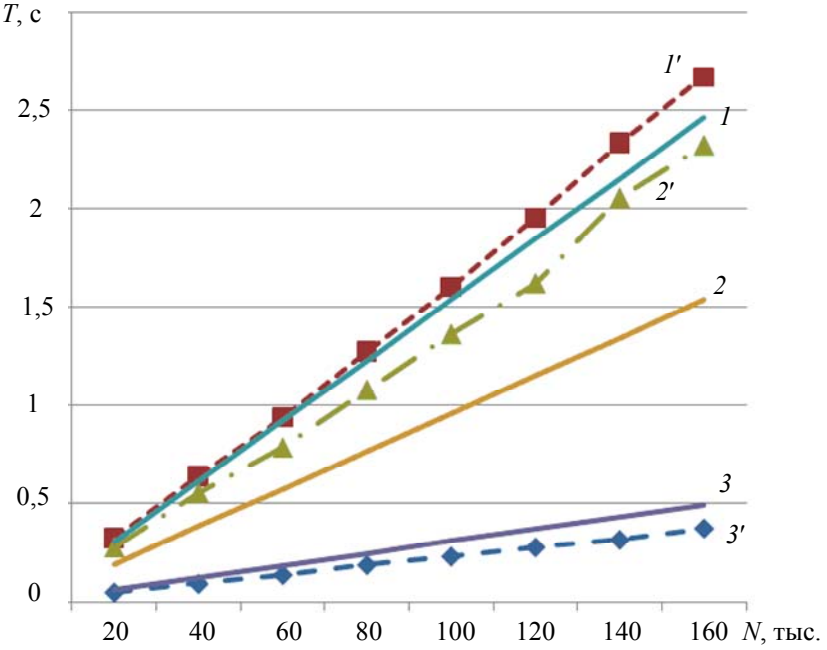


Рис. 4

Заключение. Представленные результаты исследования демонстрируют линейный рост сложности поиска в расширяющемся контексте, что позволяет использовать предложенную концептуальную модель интеллектуального агента для извлечения знаний из Semantic Web. Отдельного исследования заслуживает проблема идентификации контекста, решение которой может позволить автоматически оценивать степень доверия к результатам поиска.

Работа выполнена при финансовой поддержке ФЦП „Научные и научно-педагогические кадры инновационной России на 2009—2013 годы“ (соглашение № 14.B37.21.0406).

СПИСОК ЛИТЕРАТУРЫ

1. *Торсуева И. Г.* Контекст // Лингвистический энциклопедический словарь. М.: СЭ, 1990. С. 238—239.

2. *Бессмертный И. А.* Методы поиска информации с использованием интеллектуального агента // Изв. вузов. Приборостроение. 2009. Т. 52, № 12. С. 26—31.

3. *Bessmertny I. A.* Knowledge Visualization Based on Semantic Networks // Programming and Computer Software. 2010. Vol. 36, N 4. P. 197—204.

4. *Berners-Lee T., Hendler J., Lassila O.* The Semantic Web // Scientific American Magazine. 2001. May.

Сведения об авторе

Игорь Александрович Бессмертный — канд. техн. наук, доцент; Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики, кафедра вычислительной техники;
E-mail: igor_bessmertny@hotmail.com

Рекомендована кафедрой вычислительной техники

Поступила в редакцию 08.02.12 г.

УДК 519.688

М. С. Косяков, Д. Н. Шинкарук, А. В. Торопов, Ю. А. Шполянский

ПРИМЕНЕНИЕ ТЕХНОЛОГИИ CUDA ДЛЯ УСКОРЕНИЯ РАСЧЕТА ЦЕН ОПЦИОНОВ ЕВРОПЕЙСКОГО ТИПА СЕТОЧНЫМ МЕТОДОМ

Дифференциальное уравнение Блэка—Шоулза решено численно по схеме Кранка—Николсона на графическом процессоре с применением технологии CUDA. Использование видеокарты NVIDIA GTX 580 позволило достичь ускорения более чем в 20 раз по сравнению с однопоточным расчетом на процессоре Intel Core i7 3,4 ГГц и в 2—3 раза по сравнению с наилучшими показателями многопоточной версии, основанной на технологии GCD на процессорах 2x Intel Xeon 3,06 ГГц с 24 ядрами. Результаты получены для нагрузочных параметров, представляющих практический интерес в системах алгоритмической торговли.

Ключевые слова: *CUDA, GPU общего назначения, параллельная циклическая редукция, алгоритмическая торговля, опцион, схема Кранка—Николсона.*

Введение. Одной из важнейших задач систем алгоритмической торговли, позволяющих совершать торговые операции на электронных финансовых рынках с помощью специализированных компьютерных систем, является расчет цен опционов в режиме реального времени с учетом постоянного изменения их параметров. Точные аналитические решения доступны не для всех типов опционов и вариантов постановки задач. Поэтому для теоретического ценообразования широко применяются численные методы [1]. В последние годы активно исследуются возможности применения современных видеокарт с мощными графическими процессорами (ГП) для ускорения расчетов [2—5]. В большинстве работ рассматриваются численные методы, обладающие естественным параллелизмом — метод Монте-Карло, биномиальная модель или явные разностные схемы [2, 3].

Для ускорения расчета цен опционов с использованием неявной схемы Кранка—Николсона второго порядка точности в настоящей работе применена технология CUDA-вычислений (*Compute Unified Device Architecture*) на ГП, разработанном компанией NVIDIA. Алгоритмы реализованы в виде CUDA-программы, исполняемой ГП видеокарты, а также в виде одно- и многопоточного приложения для выполнения центральным процессором (ЦП). Проведено сравнение времени вычислений и погрешности получаемых на современном оборудовании результатов в условиях, характерных для систем алгоритмической торговли. Выявлены условия и способы оптимальной организации вычислительного процесса.

Математическая постановка задачи. Модель Блэка—Шоулза — классическая для расчета цен опционов. Зависимость цены опциона $V(S,t)$ от цены базового актива S и времени t в этой модели описывается дифференциальным уравнением [1]:

$$\frac{\partial V}{\partial t}+\frac{1}{2}\sigma^2 S^2\frac{\partial^2 V}{\partial S^2}+rS\frac{\partial V}{\partial S}-rV=0, \qquad (1)$$

где r — безрисковая процентная ставка и σ — волатильность.

В настоящей работе уравнение (1) решается численно методом конечных разностей по схеме Кранка—Николсона. Рассматривается диапазон цен базового актива от нижней $S_{\min}$ до верхней $S_{\max}$ границы, для интервала времени от даты исполнения $t=T$ до текущего момента $t=0$. Введем равномерную сетку значений цены базового актива $S_i=S_{\min}+i\Delta S$, $i=0,\ldots,I$, с шагом $\Delta S=(S_{\max}-S_{\min})/I$, а также времени $t^k=T-k\Delta t$, $k=0,\ldots,K$, с шагом $\Delta t=T/K$, где I и K — количество шагов по S и t соответственно. В неявной по времени схеме Кранка—Николсона соотношение (1) аппроксимируется по S и t со вторым порядком точности [1]. На временных шагах решаются системы линейных алгебраических уравнений (СЛАУ) с трехдиагональной матрицей. Коэффициенты матрицы рассчитываются один раз, поскольку r и σ — константы.

Для полноты математической постановки задачи необходимо установить начальное распределение $V(S,T)=P(S)$, где $P(S)$ — функция выплаты опциона, а также граничные условия. В работе рассмотрение ведется на примере распространенных европейских put-опционов с ценой исполнения $X>0$:

$$P(S)=\max\{[X-S(T)],\,0\}.$$

Для put-опциона $S_{\min}=0$, $S_{\max}=\infty$. Граничные условия имеют вид:

$$V(S_{\min},t)=X\exp[-r(T-t)],\quad V(S_{\max},t)=0.$$

В разностной схеме используется значение $S_{\max}=3X$, так как $V(3X,t)\approx 0$ при $\sigma<0{,}8$.

СЛАУ с трехдиагональной матрицей на ЦП решалась традиционным методом прогонки [6]. Алгоритм этого метода не может быть эффективно распараллелен для архитектуры CUDA. Поэтому для ГП применен метод параллельной циклической редукции (ПЦР), также использован ряд идей по ускорению вычислений, предложенных нами ранее в работе [7].

В алгоритмической торговле опционы естественным образом объединяются в группы по общему базовому активу. Опционы внутри группы, как правило, имеют различные значения параметров X, T и σ. В результате решения задачи получают набор значений $V(S_0,0)$ для всей группы опционов при текущей цене актива $S=S_0$.

Архитектурные особенности реализации алгоритмов. Алгоритмы реализованы в виде CUDA-программы для выполнения на ГП видеокарты, а также в виде одно- и многопоточного приложения для выполнения на ЦП. Многопоточная версия построена с использованием технологии GCD (*Grand Central Dispatch*) [8]. Она позволяет назначать задачи в приложении, которые в зависимости от загруженности системы автоматически распределяются по потокам для параллельного выполнения. В настоящей работе под одной задачей GCD понимается расчет группы опционов с общим базовым активом.

Технология CUDA использует ГП в роли массово-параллельного сопроцессора к ЦП [9]. В задачи ЦП входит подготовка данных в оперативной памяти ЦП, их передача в память видеокарты для последующей обработки ГП и возвращение результатов. Код массово-параллельных вычислений — ядро CUDA-программы (*kernel*), выполняется ГП как набор многих одновременно работающих потоков (*threads* в терминах CUDA) [9].

В нашей реализации алгоритма ПЦР каждый поток занимается расчетом значения цены опциона в одном узле сетки S_i, взаимодействуя с другими потоками через разделяемую память [5]. Потоки, выполняющие расчет одного опциона, объединены нами в блок, размер которого совпадает с количеством узлов сетки, а количество блоков определяется числом опционов в одной группе. Расчет каждой группы опционов организован в специальные потоки команд (*streams*), которые выполняются одним ГП последовательно с использованием асинхронной передачи данных из оперативной памяти ЦП в память видеокарты и обратно. Доступ к значениям параметров опционов в оперативной памяти ЦП из ядра CUDA-программы осуществляется через механизм отображаемой памяти [10].

Операции с вещественными числами двойной точности (тип *double*) ГП исполняет заметно медленнее, чем операции с числами одинарной точности (тип *float*). Код ядра CUDA-программы разрабатывался с использованием типа *float* в отличие от реализованных нами для ЦП алгоритмов (тип *double*). Поэтому в работе изучено влияние использования типа *float* на погрешность итогового результата вычислений.

Для анализа реализованных в работе алгоритмов расчета использованы экспериментальные стенды (табл. 1).

Таблица 1

Характеристики экспериментальных стендов

Модель	SunFire x4170 M2	ПК Core i7
ЦП	2 x Xeon X5675, 3,06 ГГц, 24 ядра	Core i7-2600, 3,4 ГГц, 8 ядер
Объем ОЗУ	49 ГБ	4 ГБ
ОС	Solaris 10 x86_64 U10	Ubuntu 11.10 x86_64

Многопоточная версия приложения, использующая технологии GCD, запускалась на ЦП сервера SunFire x4170 M2. Персональный компьютер (ПК) Core i7 использовался для запуска однопоточного приложения на ЦП и программы с применением CUDA. Параметры видеокарты NVIDIA GTX 580, используемой в экспериментах: 16 мультипроцессоров, 32 ядра в мультипроцессоре, частота процессора — 1,564 ГГц, частота памяти — 2 ГГц, объем памяти — 1536 МБ, разрядность шины видеопамяти — 384 бит.

В работе измерялось общее время τ выполнения расчетов для всех M групп опционов. Для CUDA-программы значение τ включало также время передачи данных. Для многопоточного приложения измерялось время расчета в режиме холодного старта τ_c (*cold*) с учетом всех накладных расходов на запуск потоков и время расчета в режиме под нагрузкой τ_h (*hot*), определяемое как минимальное время расчета в серии последовательных экспериментов в работающем приложении.

Результаты были получены для европейских put-опционов с параметрами: $X = 100$ у.е., $\sigma = 25$ % в год, $r = 5$ % в год, $T = 0{,}3$ года. Если не оговорено отдельно, применялись следующие настройки сетки: шаг по времени $\Delta t = 0{,}02$ года, количество шагов по цене $I = 150$.

Оптимизация CUDA-программы. Объем регистровой и разделяемой памяти видеокарты ограничен, это влияет на число одновременно запущенных на ГП потоков выполнения, что определяет общую производительность системы.

Предложенный алгоритм требует относительно небольшого объема разделяемой памяти, поэтому основное внимание необходимо уделять работе с регистровой памятью. Был предпринят ряд мер для уменьшения количества регистров, используемых каждым потоком выполнения:

— объединение функций программы для уменьшения числа локальных переменных;

— применение спецификатора volatile, что в некоторых случаях позволяет уменьшить количество используемых регистров;

— использование числовых констант одинарной точности;

— использование библиотеки быстрой математики (use_fast_math).

Перечисленные меры по реорганизации ядра CUDA-программы, используемые далее, позволили сократить общее время выполнения расчетов τ более чем на 20 %.

Анализ времени вычислений. В табл. 2 приведены значения времени расчета одной группы опционов в разных реализациях. Для многопоточного GCD-приложения в режиме холодного старта $\tau_c \approx \tau_h$, поскольку параллелизм в GCD реализован на уровне групп опционов, т.е. расчет одной группы рассматривается как одна задача. Преимущество однопоточного приложения (τ_1), запущенного на ПК Core i7, по сравнению с GCD здесь обусловлено разницей тактовых частот процессоров экспериментальных стендов (3,4 ГГц у Core i7 против 3,06 ГГц у Xeon X5675).

Таблица 2

Сравнение времени расчета одной группы опционов

N	τ, мс	τ_1, мс	τ_c, мс	τ_h, мс	τ / τ_1	τ / τ_c	τ / τ_h
1	0,143	0,071	0,104	0,062	0,497	0,729	0,434
5	0,132	0,265	0,353	0,309	2,007	2,673	2,34
10	0,137	0,575	0,663	0,619	4,193	4,835	4,514
20	0,143	1,132	1,277	1,238	7,939	8,956	8,682
60	0,179	3,475	3,826	3,713	19,413	21,374	20,743
200	0,474	10,602	12,419	12,378	22,381	26,217	26,131
500	0,979	22,746	30,987	30,940	25,731	31,663	31,615
800	1,501	41,021	49,568	49,519	27,323	33,016	32,984
1000	1,871	51,822	62,053	61,88	27,686	33,151	33,059

Из табл. 2 видно, что видеокарта NVIDIA GTX 580 позволяет достичь более чем 20-кратного преимущества при одновременной обработке нескольких опционов в группе ($N \geq 60$ для данного случая).

Рассмотрим результаты исследований, полученных для нескольких групп опционов. На рис. 1 представлен график зависимости времени τ (*1*), τ_c (*2*), τ_h (*3*), τ_1 (*4*) обработки $M = 10$ групп опционов от числа N опционов в группе.

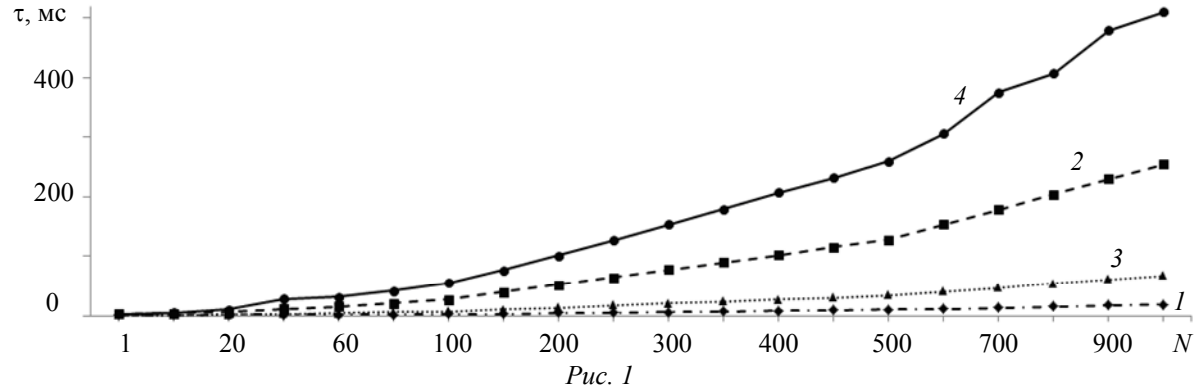


Рис. 1

Как видно из рис. 1, производительность многопоточного приложения GCD под нагрузкой значительно выше производительности однопоточного. Несмотря на это быстродействие CUDA-программы в 2,5—3 раза выше по сравнению с GCD в нагруженном состоянии и более чем в 9 раз по сравнению с GCD в режиме холодного старта (рассчитан важный для задач алгоритмической торговли случай $N \geq 100$). При увеличении N выигрыш в производительности CUDA-программы возрастает.

При увеличении количества групп M возрастает число операций передачи данных. Отсюда следует, что использование технологии CUDA должно давать большее преимущество при расчете малого числа групп большего размера. Для GCD, наоборот, с ростом M увеличивается количество используемых потоков, что повышает производительность.

На рис. 2 продемонстрировано преимущество CUDA-программы по сравнению с одно- и многопоточным приложением на ЦП. Точки кривой обозначают границы, ниже которых

использование соответствующей реализации для ЦП предпочтительней применения CUDA (GCD Hot (*1*), GCD Gold (*2*), Core i7 (*3*)).

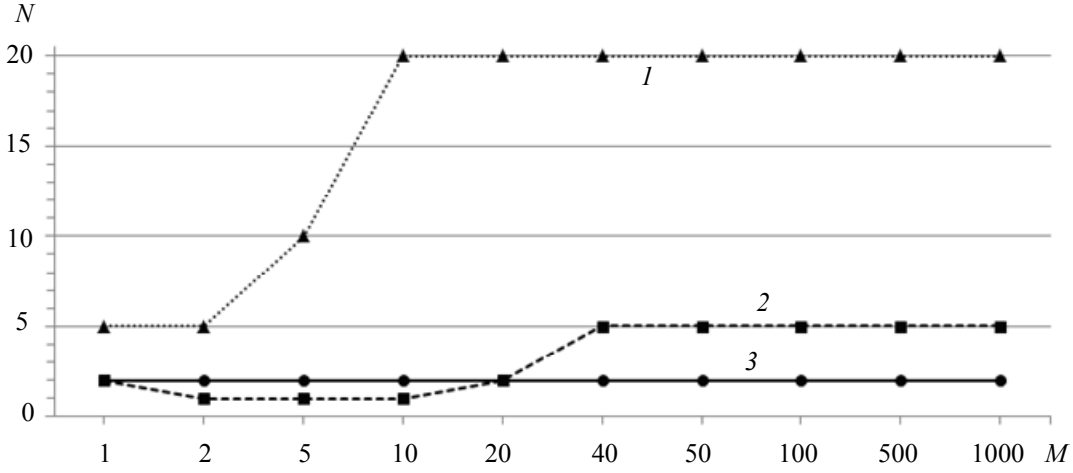


Рис. 2

Из рис. 2 видно, что при $N \geq 20$ CUDA-приложение выигрывает у GCD под нагрузкой для любого количества групп M.

Влияние особенностей работы графического процессора на время вычислений. В ходе экспериментов были отмечены некоторые особенности ГП и их влияние на время вычислений. Из-за архитектурно-функциональных особенностей CUDA потоки в блоке исполняются не отдельно, а группами по 32 потока (*warp* в терминах CUDA). Количество одновременно исполняемых блоков не может превышать числа мультипроцессоров видеокарты (16 для NVIDIA GTX 580), поэтому наблюдается скачкообразный рост времени вычислений при увеличении M и N. На рис. 3 представлено время расчета 10 (*1*), 50 (*2*) и 100 (*3*) групп опционов при различном размере групп. На рис. 4 приведена зависимости времени расчета одного опциона в от числа узлов сетки $I'=I+1$.

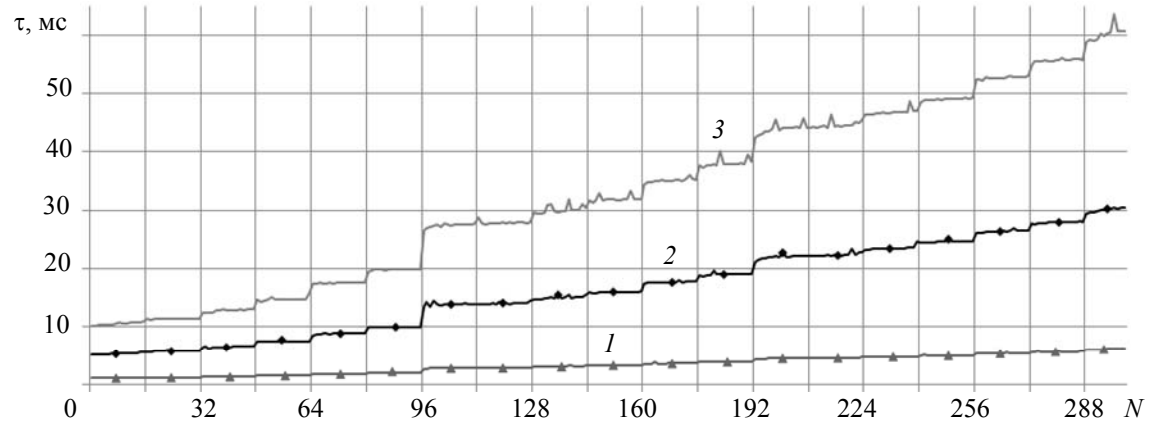


Рис. 3

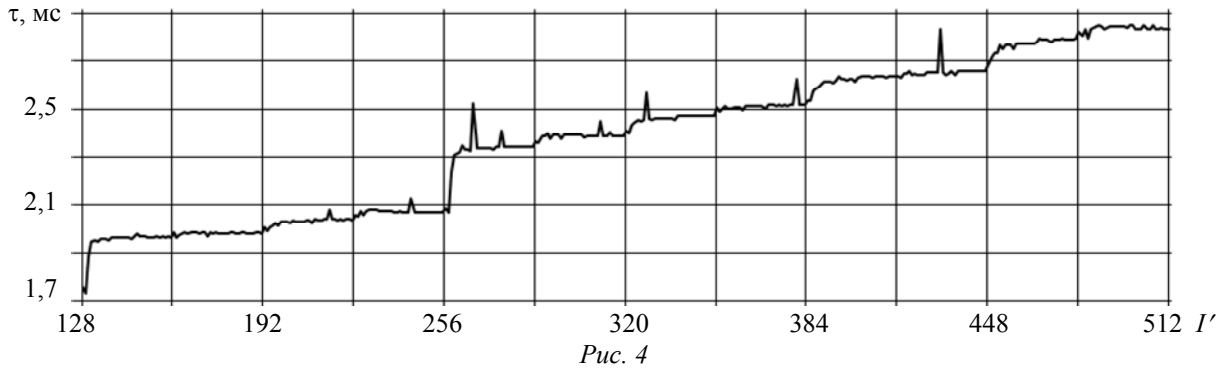


Рис. 4

Анализ погрешностей вычислений. Нами проанализированы погрешности численных алгоритмов, реализованных для разных архитектур. Анализ возможен благодаря наличию аналитического решения уравнения (1) для опционов европейского типа в отсутствие дискретных дивидендных выплат по базовому активу за время „жизни“ опциона. На рис. 5 приведены графики абсолютной погрешности (ЦП тип *double* (*1*), ЦП тип *float* (*2*), ГП тип *float*, ГП тип *use_fast_math* (*3*)), складывающейся из ошибки аппроксимации уравнения (1) схемой Кранка—Николсона и погрешности математических вычислений вещественных чисел с плавающей точкой. Отдельно рассматривался случай использования библиотеки быстрой математики, обладающей меньшей по сравнению со стандартной библиотекой точностью вычисления сложных функций для чисел одинарной точности (тип *float*) [10].

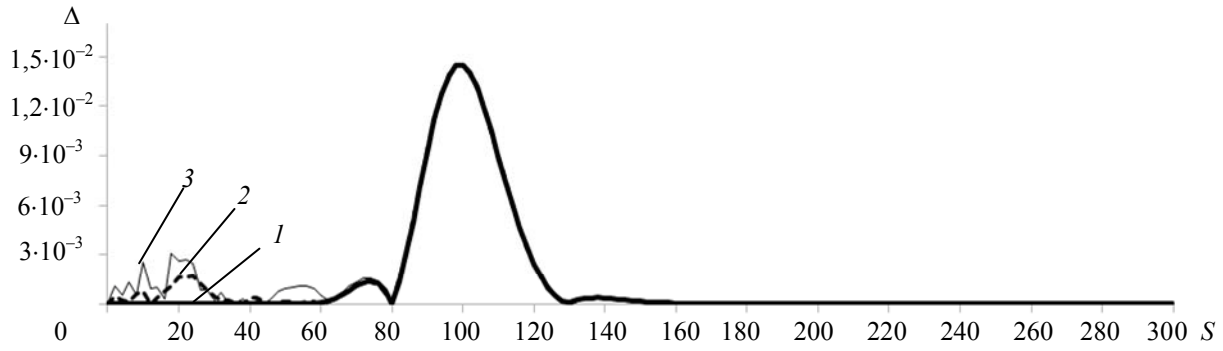


Рис. 5

Рис. 5 составлен для наиболее малого шага по времени (Δt = 0,001) и соответственно большого числа шагов (K = 300), в этом случае наблюдается максимальная погрешность вычислений. Значение погрешности, вносимой использованием *float* (область малых значений S) в 5 раз меньше максимальной погрешности схемы Кранка—Николсона в области $S \sim X = 100$ (относительная ошибка при этом порядка 0,3 %). Библиотека быстрой математики не снижает точность расчетов.

Выводы. В работе технология CUDA применена для ускорения расчета цен опционов европейского типа методом Кранка—Николсона. Соответствующие алгоритмы реализованы в виде CUDA-программы, исполняемой графическим процессором видеокарты, а также в виде одно- и многопоточного приложения, исполняемого ЦП. Многопоточная версия построена с использованием технологии GCD. Выявлены и сформулированы основные направления оптимизации ядра CUDA-программы, позволившие сократить время расчетов более чем на 20 %.

Отмечено влияние архитектурно-функциональных особенностей ГП на время вычислений. Сформулированы рекомендации по выбору параметров расчета цен опционов. Сделан вывод о допустимости использования вещественных чисел с одинарной точностью. Использование библиотеки быстрой математики ускоряет вычисления, но не приводит к увеличению погрешности.

Разработанная CUDA-программа, исполняемая на видеокарте NVIDIA GTX 580, позволяет достичь более чем 20-кратного преимущества перед однопоточной реализацией соответствующего алгоритма на современном ЦП Intel Core i7 3,4 ГГц. Быстродействие CUDA-программы по сравнению многопоточной реализацией выше в 2,5—3 раза с GCD в нагруженном состоянии и более чем 9 раз по сравнению с GCD в режиме холодного старта для практически важного для алгоритмической торговли количества опционов в группе. При увеличении числа опционов в группе выигрыш в производительности CUDA-программы возрастает.

СПИСОК ЛИТЕРАТУРЫ

1. *Wilmott P.* Paul Wilmott on Quantitative Finance. Wiley, 2006. 1380 p.
2. Monte Carlo Option Pricing [Электронный ресурс]: <http://developer.nvidia.com/cuda-cc-sdk-code-samples>.
3. Binomial Option Pricing [Электронный ресурс]: <http://developer.nvidia.com/cuda-cc-sdk-code-samples>.
4. *Lyuu Y. D., Wen K. W., Wu Y.* C. Performance of GPU for Pricing Financial Derivatives Convertible Bonds // To appear in J. of Information Science and Engineering [Электронный ресурс]: <http://www.iis.sinica.edu.tw/page/jise/FILE/AcceptedList/110/110444-POG.pdf>.
5. *Egloff D.* High Performance Finite Difference PDE Solvers on GPUs. Technical report. QuantAlea GmbH. February. 2010.
6. *Бахвалов Н. С., Жидков Н. П., Кобельков Г. М.* Численные методы. М.: Бином. Лаборатория знаний, 2003. 632 с.
7. *Шинкарук Д. Н., Шполянский Ю. А., Косяков М. С.* Анализ эффективности применения технологии CUDA для решения систем линейных уравнений с трехдиагональными матрицами в задачах расчета цен опционов // Наст. выпуск. С. 20—25.
8. Grand Central Dispatch (GCD) Reference. Apple Inc. 2011 [Электронный ресурс]: <http://developer.apple.com/library/ios/#documentation/Performance/Reference/GCD_libdispatch_Ref/Reference/reference.html>.
9. *Боресков А. В., Харламов А. А.* Основы работы с технологией CUDA. М.: ДМК Пресс, 2010. 232 с.
10. NVIDIA CUDA C Programming Guide. Version 4.2. 2011 [Электронный ресурс]: <http://developer.nvidia.com/nvidia-gpu-computing-documentation>.

Сведения об авторах

Михаил Сергеевич Косяков — канд. техн. наук; Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики, кафедра вычислительной техники; E-mail: mkosyakov@gmail.com

Дмитрий Николаевич Шинкарук — аспирант; Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики, кафедра вычислительной техники; E-mail: dimashink@gmail.com

Александр Владимирович Торопов — аспирант; Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики, кафедра компьютерных технологий; Тбрикс АБ; инженер-программист; E-mail: toropov@rain.ifmo.ru

Юрий Александрович Шполянский — д-р физ.-мат. наук, доцент; Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики, кафедра фотоники и оптоинформатики; Тбрикс АБ; ведущий математик; E-mail: shpolyan@mail.ru

Рекомендована кафедрой вычислительной техники

Поступила в редакцию 08.02.12 г.

УДК 519.688

Д. Н. Шинкарук, Ю. А. Шполянский, М. С. Косяков

АНАЛИЗ ЭФФЕКТИВНОСТИ ПРИМЕНЕНИЯ ТЕХНОЛОГИИ CUDA ДЛЯ РЕШЕНИЯ СИСТЕМ ЛИНЕЙНЫХ УРАВНЕНИЙ С ТРЕХДИАГОНАЛЬНЫМИ МАТРИЦАМИ В ЗАДАЧАХ РАСЧЕТА ЦЕН ОПЦИОНОВ

Исследованы возможности графического процессора при решении линейных систем с трехдиагональными матрицами. Показано, что целесообразно формировать матрицы непосредственно в глобальной памяти видеокарты, при этом достигается более чем 20-кратное ускорение вычислений по сравнению с однопоточным расчетом. С учетом обмена данными между оперативной памятью и памятью видеокарты достигнуто ускорение 5—8 раз при использовании механизма отображаемой памяти.

Ключевые слова: *CUDA, GPU общего назначения, система линейных уравнений, параллельная циклическая редукция, метод прогонки.*

Введение. В последние годы все более широко для высокопроизводительной обработки данных применяются дополнительные ускорители, принимающие на себя часть вычислительной нагрузки. В частности, к таким ускорителям относятся видеокарты с графическими процессорами (ГП), изначально предназначенные для использования в качестве устройств обработки графики. Производительность этих устройств растет с каждым годом, а реализации алгоритмов обработки данных на ГП позволяют значительно повысить скорость вычислений по сравнению с использованием центрального процессора (ЦП) [1].

Результаты многочисленных исследований технологии CUDA (*Compute Unified Device Architecture*), позволяющей производить массово-параллельные вычисления с использованием ГП компании NVIDIA, показали значительное увеличение скорости обработки данных в не связанных с графическими приложениями областях [2—5].

В настоящей работе рассматривается задача расчета теоретических цен опционов как численного решения дифференциального уравнения в частных производных (ДУЧП), которое часто сводят к последовательному решению систем линейных алгебраических уравнений (СЛАУ) с трехдиагональными матрицами [6]. Целью работы является анализ эффективности применения технологии CUDA для ускорения решения СЛАУ с трехдиагональными матрицами на ГП и выявление условий, при которых достигается наибольшее преимущество ГП перед ЦП. При сравнении производительности таких вычислений использованы современные ЦП Intel и ГП компании NVIDIA. Детально изучены особенности применения различных методов обмена данными между оперативной памятью компьютера (ОЗУ) и памятью видеокарты, сформулированы соответствующие рекомендации.

Особенности архитектуры CUDA. Технология CUDA основана на следующей концепции: ГП выступает в роли массово-параллельного сопроцессора к ЦП и обладает своей иерархией памяти [7]. Наиболее востребованы разделяемая память видеокарты, располагающаяся на кристалле ГП и обладающая малым временем доступа, но малым объемом, и сравнительно „медленная“ глобальная память, имеющая наибольший объем.

Программа, написанная с использованием технологии CUDA, состоит из двух частей: последовательного кода, выполняющегося на ЦП, и параллельного кода, который выполняется на ГП. Параллельная часть программы, называемая ядром (*kernel*), исполняется одновременно большим количеством потоков (*threads*), которые организуются в блоки. С использованием последовательного кода подготавливаются и пересылаются данные из ОЗУ компью-

тера в память видеокарты, после чего осуществляется запуск ядра. По окончании работы ядра результаты передаются обратно в оперативную память.

Такие особенности архитектуры и программной организации работы ГП приводят к тому, что алгоритмы, предназначенные для работы на ЦП, не могут быть успешно применены для графического ускорителя. ГП требует применения специализированных параллельных алгоритмов обработки данных.

Методы решения СЛАУ. При организации вычислений на центральном процессоре СЛАУ обычно решают методом прогонки или LU-декомпозиции [8]. Алгоритм решения строго последователен и не может быть эффективно распараллелен. Поэтому при решении СЛАУ на ГП необходимы другие методы. В настоящей работе применен метод параллельной циклической редукции (ПЦР, *Parallel Cyclic Reduction*), который характеризуется большим объемом вычислений, но обладает высокой степенью параллелизма [9].

На рис. 1 представлена схема метода ПЦР. СЛАУ обрабатывается на ГП одним блоком, при этом расчетом каждого неизвестного x_i занимается отдельный поток P_i. Метод выполняется пошагово. На шаге j производится вычисление новых коэффициентов P_i^j для уравнений системы с использованием коэффициентов, рассчитанных на предыдущем шаге $j-1$. Взаимодействие потоков осуществляется через разделяемую память. На последнем этапе вычислений (на рис. 1 обозначен ромбами) производится итоговый расчет неизвестных x_i. Работа алгоритма требует $\log_2(N)$ шагов, где N — количество уравнений в СЛАУ.

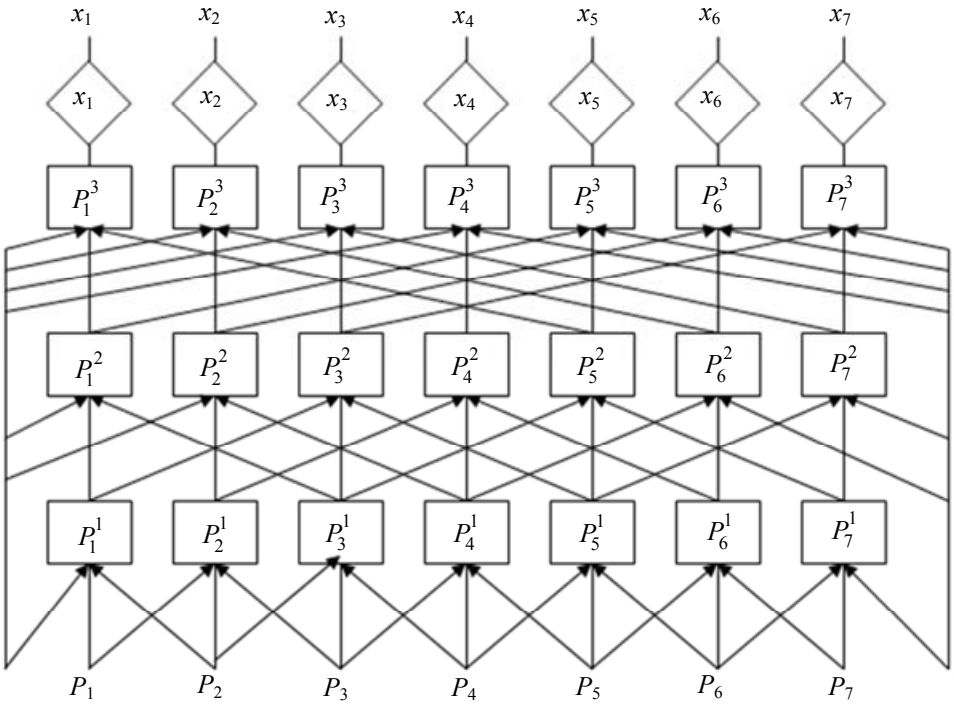


Рис. 1

Вычислительная сложность ПЦР пропорциональна $N\log_2(N)$, в то время как сложность метода прогонки линейно зависит от количества уравнений в системе. Но при массово-параллельном исполнении каждый отдельный поток выполняет лишь небольшую часть общего объема вычислений, пропорциональную $\log_2(N)$.

Решение одной системы. Расчеты цен опционов и измерения времени вычислений проводились на персональном компьютере (ПК) со следующими параметрами: ЦП Intel Core i5 3,8 ГГц; 4 ГБ ОЗУ; ГП NVIDIA GTX 580; ОС Ubuntu 11.04 (64 бит); версия драйвера CUDA 270.41.19 (64 бит).

При организации расчетов на ГП в ОЗУ компьютера формировались 4 массива данных, соответствующие трем диагоналям матрицы и правому столбцу СЛАУ. Для матрицы СЛАУ

обеспечивалось требование диагонального преобладания. Массивы передавались в память видеокарты, после чего запускалось ядро для обработки данных на ГП. Вектор результатов копировался из памяти устройства обратно в ОЗУ. Измеряемыми характеристиками были полное время T_f (*full*) от начала передачи данных на видеокарту до момента возвращения результатов в ОЗУ и чистое время вычислений на видеокарте, т.е. время выполнения ядра T_k (*kernel*). Согласно идеологии ПЦР, число запускаемых потоков выполнения совпадало с числом уравнений N. Для ускорения расчетов все потоки запускались в одном блоке, а требуемые данные располагались в разделяемой памяти.

Таблица 1

Время решения одной СЛАУ

N	2	4	8	16	32	64	128	256	512	1024
T_k, мс	4	4	5	6	7	7	8	9	12	23
T_f, мс	117	113	119	117	119	119	119	121	125	263
T_{CPU}, мс	1	1	2	2	2	3	5	11	19	39

Как видно из табл. 1, время работы ядра T_k на ГП сопоставимо со временем решения системы уравнений методом прогонки на ЦП T_{CPU}, однако для СЛАУ с $N \le 128$ превышает его. Связано это с тем, что при массово-параллельных вычислениях на ГП существуют затраты на создание и инициализацию потоков выполнения. Для CUDA время инициализации практически не зависит от количества создаваемых потоков, поэтому наблюдается более медленный рост T_k по сравнению с T_{CPU}. Важно отметить, что значение T_{CPU} линейно зависит от числа уравнений N в решаемой СЛАУ.

Значение полного времени T_f слабо изменяется с ростом размерности СЛАУ N, но в десятки раз превышает T_k и T_{CPU} (см. табл. 1). На рис. 2 представлена временная диаграмма работы видеокарты: передача данных из ОЗУ в память видеокарты (*1*) и обратно (*5*); работа ядра (*3*); простои (*2*, *4*). Время работы устройства составило 7,2 % от общего времени. Длительные простои (белые области) связаны с отработкой драйвера устройства, планированием процессов в системе, синхронизацией и пр.

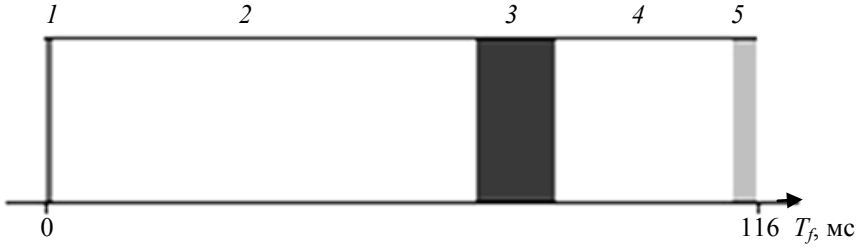


Рис. 2

Достичь преимущества ГП перед ЦП возможно, увеличив занятость устройства и эффективно используя его вычислительные ресурсы. Одним из способов достижения этой цели является параллельное решение большого числа СЛАУ.

При анализе погрешностей вычислений рассматривалась невязка решения СЛАУ. Наибольшее наблюдаемое значение невязки для метода ПЦР составило $\Delta = 8{,}5 \cdot 10^{-5}$, в то время как для метода прогонки (МП) $\Delta = 7{,}8 \cdot 10^{-6}$ при $N = 300$. Причина такого различия заключается в разном количестве арифметических операций, производимых над элементом данных. В реализации метода ПЦР осуществляется несколько больше операций умножения по сравнению с МП. Важно отметить, что для ПЦР количество арифметических операций определяется размерностью системы N: при увеличении числа уравнений в СЛАУ возрастает количество шагов решения. В наших экспериментах для $N = 8$ невязка составила $\Delta = 8 \cdot 10^{-8}$, а для $N = 64$ уже $\Delta = 1{,}2 \cdot 10^{-6}$. Эти значения не зависят того, где запускаются методы расчета: на ЦП или ГП, следовательно, использование ГП не приводит к дополнительным погрешностям.

Параллельное решение многих систем. На рис. 3, *а* представлена зависимость времени T_k и T_f расчетов на ГП от количества M одновременно обрабатываемых СЛАУ при $N = 256$. Как видно, при небольшой нагрузке значения T_k (*1*) и T_f (*2*) мало меняются, что связано с недостаточной загрузкой устройства. При $M = 64$ достигается предел одновременно запущенных потоков выполнения (16 384 потоков для видеокарты NVIDIA GTX 580), после чего время вычислений растет линейно. Стоит отметить, что время расчета на ЦП T_{CPU} (*3*) изменяется линейно во всей области рассматриваемой нагрузки.

На рис. 3, *б* представлена зависимость скорости решения СЛАУ V (определяемой как отношение числа M решаемых СЛАУ ко времени их расчета T) от нагрузки на систему. Рост скорости выполнения расчетов V_f (*1*) на ГП с увеличением M происходит за счет сокрытия задержек доступа к памяти при одновременной работе большого количества потоков выполнения, при этом скорость ЦП (кривая *2*) практически не изменяется.

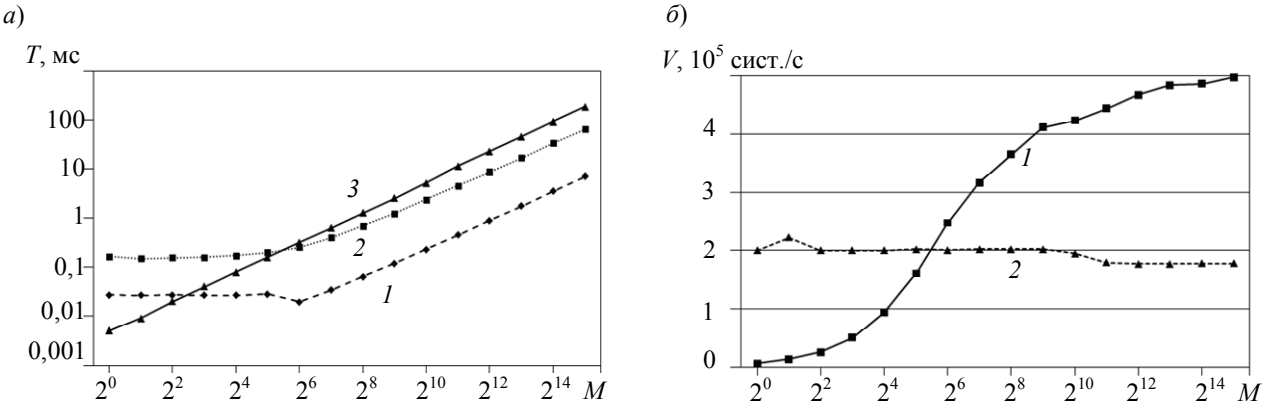


Рис. 3

При решении большого числа СЛАУ объем передаваемых данных между ОЗУ и памятью видеокарты значительно возрастает, что существенно сказывается на общем времени расчетов (табл. 2). Так, при увеличении числа решаемых систем M возможно сокращение чистого времени вычислений на ГП перед ЦП в 25 раз. Однако при пересылке данных между ОЗУ и памятью видеокарты преимущество в скорости расчетов с использованием ГП составляет 2,6 раза, 90 % общего времени работы устройства отводится на передачу данных. Один из возможных выходов — такая организация вычислительного процесса, при которой матрицы СЛАУ формируется непосредственно в глобальной памяти видеокарты.

Таблица 2

Время решения различного количества систем уравнений

M	T_k, мс	T_f, мс	T_{CPU}, мс	T_{CPU} / T_k	T_{CPU} / T_f
1	0,027	0,167	0,005	0,18	0,03
16	0,026	0,171	0,08	3,01	0,47
256	0,065	0,701	1,263	19,41	1,80
1024	0,231	2,418	5,255	22,70	2,17
4096	0,893	8,763	23,077	25,84	2,63

Видеокарты последних поколений позволяют осуществлять асинхронную передачу данных из ОЗУ в память видеокарты и обратно одновременно с исполнением кода ядра [10]. Таким образом достигается эффективное функционирование приложения при наличии большого объема входных данных. Одно из ограничений при этом — необходимость обозначения области ОЗУ, в которой хранятся передаваемые данные, как невыгружаемой области памяти. При реализации этого метода входные данные „вручную“ разбиваются на фрагменты таким образом, чтобы время передачи одного фрагмента приближалось ко времени его обработки.

Таблица 3

Полное время решения СЛАУ при синхронном и асинхронном обмене данными

M	T_f^s, мс	T_f^a, мс	T_{CPU}, мс	T_{CPU} / T_f^a
1	0,167	0,041	0,005	0,12
16	0,171	0,060	0,08	1,32
256	0,701	0,389	1,263	3,25
1024	2,418	1,374	5,255	3,82
4096	8,763	5,141	23,077	4,49

Применение асинхронной передачи данных позволило уменьшить общее время расчетов T_f^a более чем в 1,5 раза по сравнению со случаем синхронного обмена T_f^s (см. табл. 3). Для небольшого числа решаемых систем M наблюдается уменьшение времени расчетов в 3—4 раза, поскольку при асинхронной передаче данных исчезает значительная часть простоев (см. рис. 2). Однако необходимость „ручного“ разбиения входных данных на части весьма усложняет применение данного метода.

Нами исследована эффективность применения механизма отображаемой (*mapped*) памяти для прямого доступа к данным в ОЗУ из исполняемого на ГП ядра. В этом случае соответствующая область ОЗУ также должна быть обозначена как невыгружаемая. Результаты сравнения полного времени выполнения для синхронного T_f^s, асинхронного T_f^a обмена данными и T_f^m при использовании отображаемой памяти приведены в табл. 4.

Таблица 4

Сравнение времени расчета

M	T_f^s, мс	T_f^a, мс	T_f^m, мс	T_{CPU}, мс	T_{CPU} / T_f^m
1	0,167	0,041	0,026	0,005	0,19
16	0,171	0,060	0,038	0,08	2,10
256	0,701	0,389	0,269	1,263	4,69
1024	2,418	1,374	1,001	5,255	5,25
4096	8,763	5,141	3,101	23,077	7,44

Из таблицы видно, что использование отображаемой памяти дает значительное преимущество перед другими методами обмена данными. В этом случае достигается наилучшее перекрытие процессов передачи данных и решения СЛАУ во времени. Полное время выполнения на ГП по сравнению с ЦП может сокращаться в 5—8 раз для большого числа СЛАУ ($N > 1000$).

Выводы. В работе проанализирована эффективность применения технологии CUDA, позволяющей производить массово-параллельные вычисления с использованием ГП компании NVIDIA, для ускорения решения СЛАУ с трехдиагональными матрицами в задачах расчета цен опционов как численного решения ДУЧП. При организации вычислений на ЦП СЛАУ решались методом прогонки, который является строго последовательным, поэтому при расчетах на видеокарте использовался метод ПЦР.

Результаты показали допустимую для практических задач погрешность математических вычислений вещественных чисел с плавающей точкой для метода ПЦР. Использование ГП не вносит дополнительных погрешностей.

Сравнение времени выполнения расчетов показало неэффективность применения ГП для решения одной СЛАУ ввиду малой загрузки устройства. Для увеличения доли занятости ГП авторы рекомендуют решать одновременно большое число СЛАУ, что актуально для современных систем алгоритмической торговли. Матрицы СЛАУ целесообразно формировать непосредственно в глобальной памяти видеокарты.

Наша реализация алгоритма ПЦР на видеокарте NVIDIA GTX 580 позволила достичь более чем 20-кратного ускорения вычислений по сравнению с однопоточным расчетом на ЦП

Intel Core i5 3,8 ГГц при хранении данных в памяти видеокарты и одновременном решении более 1000 СЛАУ. Для задач, в которых формирование матриц СЛАУ непосредственно в памяти видеокарты невозможно, был проведен детальный анализ различных способов обмена данными между ОЗУ и памятью видеокарты. В этом случае для ГП удалось добиться ускорения в 5—8 раз при использовании механизма отображаемой памяти.

СПИСОК ЛИТЕРАТУРЫ

1. *Simoes B.* General-purpose computing on the GPU (GPGPU) [Электронный ресурс]: <http://www.think-techie.com/2009/09/general-purpose-computing-on-gpu-gpgpu.html>.
2. *Zhang Y., Cohen J., Owens J. D.* Fast Tridiagonal Solvers on the GPU // Proc. of the 15th ACM SIGPLAN Symp. on Principles and Practice of Parallel Programming (PPoPP 2010). 2010. January. P. 127—136.
3. *Egloff D.* High Performance Finite Difference PDE Solvers on GPUs. Technical report. QuantAlea GmbH. February. 2010.
4. *Lyuu Y. D., Wen K. W., Wu Y. C.* Performance of GPU for Pricing Financial Derivatives Convertible Bonds // To appear in J. of Information Science and Engineering [Электронный ресурс]: <http://www.iis.sinica.edu.tw/page/jise/FILE/AcceptedList/110/110444-POG.pdf>.
5. *Gaikwad A., Muni Toke I.* Parallel Iterative Linear Solvers on GPU: A Financial Engineering Case // Proc. of the 18th Intern. Conf. on Parallel, Distributed and Network-Based Computing. 2010. February. P. 607—614.
6. *Wilmott P.* Paul Wilmott on Quantitative Finance. Wiley, 2006. 1380 p.
7. *Боресков А. В., Харламов А. А.* Основы работы с технологией CUDA. М.: ДМК Пресс, 2010. 232 с.
8. *Бахвалов Н. С., Жидков Н. П., Кобельков Г. М.* Численные методы. М.: Бином. Лаборатория знаний, 2003. 632 с.
9. *Sweet R.* A parallel and vector variant of the cyclic reduction algorithm // SIAM J. Sci. Statits. Computing. 1988. Vol. 9, N 4. P. 761—765.
10. NVIDIA CUDA C Programming Guide. Version 4.2 2011 [Электронный ресурс]: <http://developer.nvidia.com/nvidia-gpu-computing-documentation>.

Сведения об авторах

Дмитрий Николаевич Шинкарук — аспирант; Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики, кафедра вычислительной техники; E-mail: dimashink@gmail.com

Юрий Александрович Шполянский — д-р физ.-мат. наук, доцент; Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики, кафедра фотоники и оптоинформатики; Тбрикс АБ; ведущий математик; E-mail: shpolyan@mail.ru

Михаил Сергеевич Косяков — канд. техн. наук; Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики, кафедра вычислительной техники; E-mail: mkosyakov@gmail.com

Рекомендована кафедрой вычислительной техники

Поступила в редакцию 08.02.12 г.

УДК 004.627

И. С. Рубина, А. Ю. Тропченко

ИССЛЕДОВАНИЕ АЛГОРИТМОВ КОДИРОВАНИЯ ПРЕОБРАЗОВАНИЕМ В ЗАДАЧАХ СЖАТИЯ КАДРОВ ВИДЕОПОСЛЕДОВАТЕЛЬНОСТИ

Предложен быстрый алгоритм кодирования видеоданных преобразованием при сжатии опорных и остаточных кадров, основанный на трехмерном преобразовании Хартли с фиксированным и переменным размером матрицы преобразования.

Ключевые слова: *кодирование преобразованием, преобразование Хартли, трехмерный алгоритм, матрица переменного размера, пространственная модель.*

Введение. Алгоритмы кодирования преобразованием широко используются для сжатия изображений, а также видеопоследовательностей. Этап кодирования преобразованием является частью пространственной модели видеокомпрессора [1]. Исходными данными для этого этапа являются отсчеты кадров видеопоследовательности. Обрабатываемые кадры подразделяются на:

1) опорные, или intra-кадры [1], не обрабатываемые блоком компенсации движения. Чем больше таких кадров выделяется в процессе сжатия видеопоследовательности, тем выше качество воспроизводимого видеоряда и меньше степень сжатия;

2) остаточные, представляющие собой разность исходного кадра и кадра-прогноза, полученного в процессе компенсации движения. Такие кадры определяют погрешность временной модели. В результате выполнения кодирования формируются коэффициенты разложения сигнала по базису, состоящему из функций, локализованных по частоте, а также в пространстве. Полученные коэффициенты преобразования поступают на вход квантователя и переупорядочиваются. Процесс сжатия завершается поэлементной обработкой потока энтропийным кодером.

Алгоритмы кодирования преобразованием подразделяются на блочные и глобальные, когда преобразованию подвергается все изображение [2]. В настоящей работе анализируется блочный тип преобразования.

Блочные преобразования выполняются над квадратными блоками опорного или остаточного кадра и после ряда операций формируют блоки коэффициентов таких же размеров. Элементами таких блоков служат сэмплы. Любой блок кадра можно восстановить с помощью линейной комбинации $N \times N$ базисных шаблонов [1], умножаемых на соответствующие весовые множители (коэффициенты преобразования). Достоинством блочных преобразований являются низкие требования к объему памяти. Среди недостатков можно выделить возможность возникновения артефактов на стыках блоков.

При кодировании на основе блоков используются ортогональные преобразования в различных функциональных базисах, например, дискретное косинусное преобразование (ДКП) и преобразование Хартли. Достоинством преобразования Хартли является то, что в отличие от преобразования Фурье оно вещественное. Преобразование Хартли определяется как

$$H(s) = \int_{-\infty}^{\infty} f(x)\,\mathrm{cas}2\pi sx dx, \qquad (1)$$

где $\mathrm{cas}2\pi sx = \sin 2\pi sx + \cos 2\pi sx$ [3].

Схемы алгоритмов. В ходе исследования был проведен анализ следующих алгоритмов преобразования.

1. Двумерное дискретное преобразование Хартли (2D-ДПХ), определяемое формулой:

$$H(u,v)=\frac{1}{NM}\sum_{i=0}^{N-1}\sum_{j=0}^{M-1}f(i,j)\operatorname{cas}\left(\frac{2\pi ui}{N}+\frac{2\pi vj}{M}\right), \tag{2}$$

где $N \times M$ — размеры матрицы преобразования, а $f(i,j)$ — значение яркости кадра в точке с координатами (i, j).

2. Двумерное дискретное косинусное преобразование (2D-ДКП), определяемое формулой:

$$Y(u,v)=C_uC_v\sum_{i=0}^{N-1}\sum_{j=0}^{M-1}f(i,j)\cos\frac{xu(2i+1)}{2N}\cos\frac{y\pi(2j+1)}{2M}, \tag{3}$$

где $C_u=\sqrt{\frac{1}{N}}$ (u=0), $C_u=\sqrt{\frac{2}{N}}$ (u>0), а $Y(i,j)$ — коэффициенты разложения.

3. Трехмерное дискретное преобразование Хартли (3D-ДПХ), определяемое в соответствии с выражением:

$$X(u,v,t)=\sum_{i=0}^{N-1}\sum_{j=0}^{M-1}\sum_{k=0}^{P-1}\operatorname{cas}\left(\frac{2\pi}{N}iu+\frac{2\pi}{M}jv+\frac{2\pi}{P}kt\right)f(i,j,k), \tag{4}$$

где P — число кадров, индекс t определяет номер двумерной страницы в трехмерном блоке разложения, а $f(i,j,k)$ — значение яркости отсчета в точке с координатами (i,j) k-го кадра.

Рассмотренные двумерные алгоритмы преобразования реализуются строчно-столбцовым методом. В матричном виде это может быть записано следующим образом:

$$F_N=\left[E_NX_N\right]E_N, \tag{5}$$

где E_N — матрица ядра дискретного ортогонального преобразования.

Дискретное многомерное преобразование Хартли реализовывалось в соответствии с алгоритмом, описанным в работе [4]. Такой подход позволяет существенно снизить вычислительные затраты.

4. Быстрое трехмерное дискретное преобразование Хартли с фиксированным размером матрицы преобразования (3D-БПХФ). Разработанный быстрый алгоритм основан на вычислении коэффициентов меньших 3D-блоков через коэффициенты более крупных.

Данный алгоритм быстрого преобразования получен в результате декомпозиции исходной суммы основного соотношения на восемь частичных сумм, для каждой из которых i, j, k либо четные, либо нечетные, и применения тригонометрических преобразований функций:

$$\left.\begin{aligned}
x_0&=X'_{000}(u,v,t)\\
x_1&=X'_{001}(u,v,t)\operatorname{cas}\frac{2\pi t}{P},\\
x_2&=X'_{010}(u,v,t)\operatorname{cas}\frac{2\pi v}{M},\\
x_3&=X'_{011}(u,v,t)\operatorname{cas}2\pi\left(\frac{v}{M}+\frac{t}{P}\right),\\
x_4&=X'_{100}(u,v,t)\operatorname{cas}\frac{2\pi u}{N},\\
x_5&=X'_{101}(u,v,t)\operatorname{cas}2\pi\left(\frac{u}{N}+\frac{t}{P}\right),\\
x_6&=X'_{110}(u,v,t)\operatorname{cas}2\pi\left(\frac{u}{N}+\frac{v}{M}\right),\\
x_7&=X'_{111}(u,v,t)\operatorname{cas}2\pi\left(\frac{u}{N}+\frac{v}{M}+\frac{t}{P}\right),
\end{aligned}\right\} \tag{6}$$

где

$$X'_{abc}(u,v,t) = \sum_{i=0}^{N/2-1}\sum_{j=0}^{M/2-1}\sum_{k=0}^{P/2-1} X(2i+a,2j+b,2k+c)\cos\left(2\pi\left(\frac{2iu}{N}+\frac{2jv}{M}+\frac{2kt}{P}\right)\right),$$

abc — трехбитный двоичный код, определяющий номер формируемой суммы.

Блок промежуточной размерности формируется на основе вычисляемых частичных сумм по формуле:

$$X(u',v',t') = \sum_{r=0}^{7}(\pm x_r), \tag{7}$$

где u', v', t' — размерность вычисляемого промежуточного блока, а r — номер частичной суммы. Промежуточный блок может иметь размерность, уменьшаемую вдвое хотя бы по одной из координат. Если размер блока уменьшается по одной или по всем координатам, то знак суммы считается отрицательным. При уменьшении размеров по двум координатам знак не изменяется.

5. Быстрое трехмерное дискретное преобразование Хартли с переменным размером матрицы преобразования (3D-БПХП). Выбор размера матрицы преобразования определяется на этапе компенсации движения по следующему алгоритму [5]:

— все изображение делится на блоки равного размера;

— для каждого блока выполняется проверка условия: если величина ошибки (соответствие) обрабатываемого блока с наиболее схожим блоком ссылочного кадра выше некоторой пороговой, то блок делится на четыре промежуточных фрагмента меньшего размера;

— если достигнуто заданное число промежуточных блоков или получена требуемая точность совпадения, процесс для исходного блока останавливается.

Экспериментальные результаты. Для анализа описанных ранее алгоритмов кодирования преобразованием были выбраны стандартные тестовые последовательности группы MPEG: „Теннис“, „Бригадир“ и „Береговая охрана“ [6].

Для количественного сравнения эффективности рассматриваемых преобразований были использованы следующие характеристики:

1) пиковое соотношение сигнал/шум (PSNR, peak signal to noise ratio) (рис. 1), вычисляемое в соответствии с формулой:

$$\mathrm{PSNR} = 10\log_{10}\frac{(2^q-1)^2}{\mathrm{MSE}}, \tag{8}$$

где q=8 — разрядность цветовой схемы, а MSE — среднеквадратичное отклонение яркости исходного изображения от восстановленного, определяемое формулой:

$$\mathrm{MSE} = \frac{1}{MN}\sum_{i=0}^{M-1}\sum_{j=0}^{N-1}\left\|f(i,j)-\hat{f}(i,j)\right\|^2, \tag{9}$$

где $f(i,j)$ и $\hat{f}(i,j)$ — яркость соответствующих пикселов исходного и восстановленного кадра;

2) характеристика, выражающая зависимость искажения сигнала (PSNR) от степени его сжатия $K_{сж}$ (рис. 2, *1* — 2D-ДКП; *2* — 2D-ДПХ; *3* — 3D-ДПХ, 3D-БПХФ; *4* — 3D-БПХП);

3) вычислительная сложность алгоритма, измеряемая числом операций умножения Q_1 и сложения Q_2 на пиксел (рис. 3, *а* и *б* соответственно). На рис. 4 представлены зависимости числа операций Q от размера блока V с учетом возможного приращения его размерности.

Все эксперименты проводились для 256 уровней квантования.

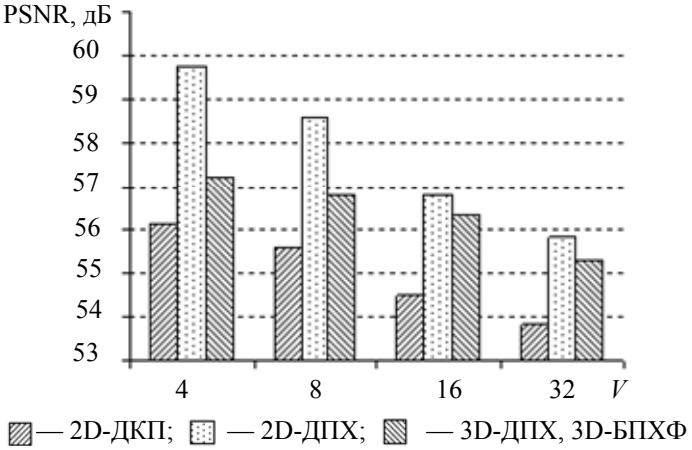


Рис. 1

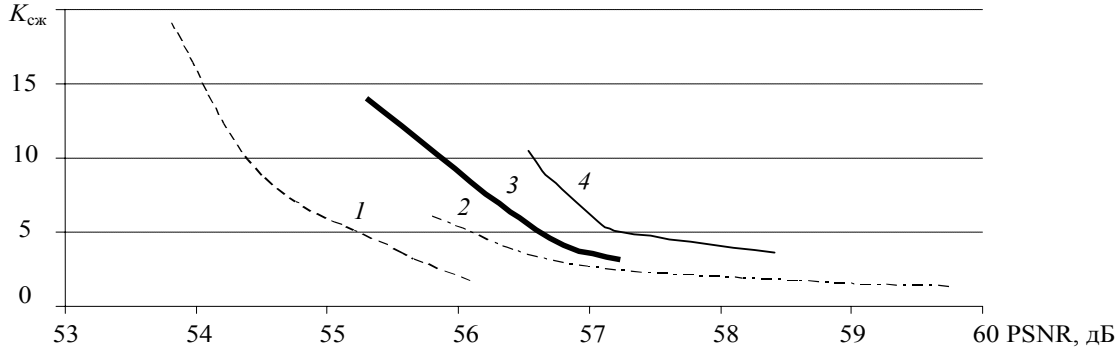


Рис. 2

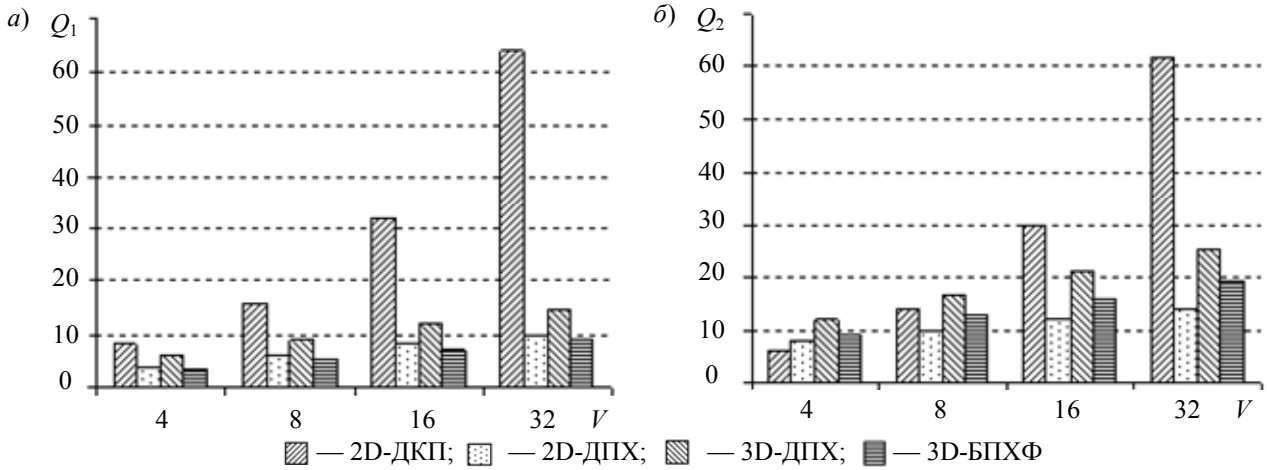


Рис. 3

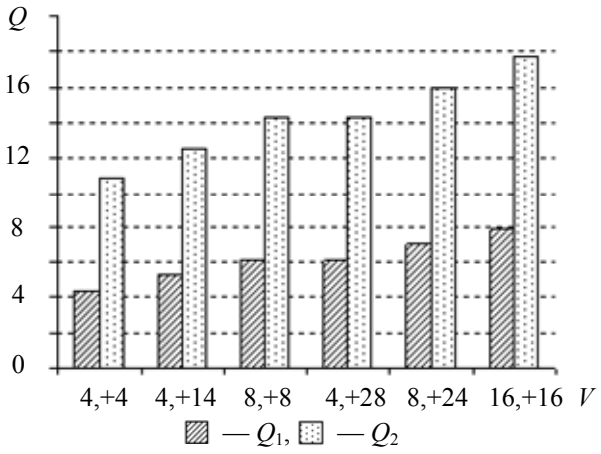


Рис. 4

Заключение. В ходе анализа было установлено следующее:

1) алгоритм 2D-ДПХ превзошел 2D-ДКП по качеству восстановленной видеопоследовательности на 5 %, но уступил ему в степени сжатия;

2) алгоритм 3D-ДПХ позволил вдвое увеличить степень сжатия видеопоследовательности при незначительном ухудшении ее качества при восстановлении. Это объясняется тем, что алгоритм выполняет преобразование не только в пространстве, но и во времени, устраняя соответствующую избыточность;

3) алгоритм 3D-БПХФ позволил на 30 % сократить число операций сложения/умножения на пиксел кадра видеопоследовательности за счет иерархического расчета коэффициентов преобразования;

4) алгоритм 3D-БПХП обеспечил повышение качества восстановленной видеопоследовательности на 2 %, а также степень ее сжатия на 1,5 %, что объясняется использованием адаптивно выбираемого размера матрицы преобразования для областей с мелкими деталями и для областей фона соответственно.

Таким образом, при использовании предложенных алгоритмов получено более высокое качество восстановленной видеопоследовательности и большее значение коэффициента сжатия при меньших вычислительных затратах за счет применения алгоритмов быстрого преобразования и переменного размера матрицы преобразования.

СПИСОК ЛИТЕРАТУРЫ

1. *Ричардсон Я.* Видеокодирование. H.264 и MPEG-4 — стандарты нового поколения. М.: Техносфера, 2005.

2. *Ватолин Д., Ратушняк А., Смирнов М., Юкин В.* Методы сжатия данных. Устройство архиваторов, сжатие изображений и видео. М.: ДИАЛОГ-МИФИ, 2003.

3. *Брейсуэлл Р.* Преобразование Хартли. М.: Мир, 1990.

4. *Yonghong Z., Guoan B., Abdul R. L.* New algorithms for multidimensional discrete Hartley transform // Signal Processing. 2002. Vol. 82. P. 1086—1095.

5. *Ribas-Corbera J., Neuhoff D. L.* On the optimal block size for block-based, motion compensated video coders // SPIE Proc. of Visual Communications and Image Processing. 1997. Vol. 3024. P. 1132—1143.

6. *Рубина И. С., Тропченко* А. Ю. Исследование алгоритмов выбора опорных пикселов в задачах выделения сегментов кадра видеопоследовательности // Изв. вузов. Приборостроение. 2012. Т. 55, № 1. С. 9—14.

Сведения об авторах

Ирина Семеновна Рубина — аспирант; Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики, кафедра вычислительной техники; E-mail: rubren@mail.ru

Александр Ювенальевич Тропченко — д-р техн. наук, профессор; Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики, кафедра вычислительной техники; E-mail: tau@d1.ifmo.ru

Рекомендована кафедрой вычислительной техники

Поступила в редакцию 08.02.12 г.

УДК 004.627

А. А. Тропченко, А. Ю. Тропченко

НЕЙРОСЕТЕВЫЕ МЕТОДЫ ИДЕНТИФИКАЦИИ ЧЕЛОВЕКА ПО ИЗОБРАЖЕНИЮ ЛИЦА

Рассмотрены нейросетевые методы распознавания человека по изображению лица, используемые в биометрических системах идентификации.

Ключевые слова: *нейронные сети, распознавание личности, биометрические системы.*

Введение. В настоящее время все более широкое распространение получают биометрические системы идентификации. Такие системы основываются на учете уникальных биологических характеристик человека, которые однозначно определяют его образ и являются трудно подделываемыми. К основным биометрическим характеристикам относятся отпечатки пальцев, форма ладони, узор радужной оболочки, изображение лица. Распознавание человека по изображению лица наиболее распространено в биометрических системах благодаря тому, что [1, 2]:

— не требуется специальное или дорогостоящее оборудование;

— не нужен физический контакт с какими-либо устройствами ввода данных.

Такие биометрические системы не обеспечивают высокой надежности идентификации, для этого требуется применять системы, использующие различные биометрические характеристики (так называемые мультимодальные системы).

При построении подобных биометрических систем целесообразно применять нейросетевые методы распознавания лица.

Нейронная сеть (НС) состоит из элементов, называемых формальными нейронами, каждый их которых элементарен по структуре и связан с другими нейронами. Каждый нейрон преобразует совокупность сигналов, поступающих к нему на вход, в выходной сигнал. Именно связи между нейронами, кодируемые весовыми коэффициентами, играют ключевую роль. Одно из основных преимуществ НС заключается в возможности параллельного функционирования ее элементов, что существенно повышает эффективность решения задачи. Обучение НС упрощает выбор ключевых признаков, их весовых коэффициентов и связей между ними. Рассмотрим особенности применения различных типов НС для распознавания человека.

Многослойные нейронные сети (МНС) состоят из последовательно соединенных слоев, нейрон каждого из которых своими входами связан со всеми нейронами предыдущего слоя, а выходами — последующего (рис. 1). Для активации таких нейронов служат разновидности линейных, пороговых и сигмоидных функций [3]. На рис. 1 представлена архитектура многослойной нейронной сети для распознавания изображений. Нейрон с максимальной активностью (цифра 1) указывает на принадлежность к распознанному классу.

НС с одним решающим слоем способна формировать линейные разделяющие поверхности, что значительно сужает круг решаемых задач, в частности, такая сеть не сможет решить задачу типа „исключающее или“. НС с нелинейной функцией активации и двумя решающими слоями позволяет формировать любые выпуклые области в пространстве решений, а с тремя решающими слоями — области любой сложности, в том числе и невыпуклой. Обучение МНС осуществляется с помощью алгоритма обратного распространения ошибки. Такой алгоритм является разновидностью градиентного спуска в пространстве весов и обеспечивает минимизацию суммарной ошибки сети:

$$\Delta W = -\alpha \frac{dE}{dW}, \quad E = \frac{1}{2}\sum_j (y_j - t_j)^2,$$

где y_j — выходное значение j-го нейрона сети, t_j — эталонное значение выходов сети. Скорректированные значения весов передаются от входов к выходам. Алгоритм обратного распространения является *NP*-трудным, поэтому время обучения сети увеличивается экспоненциально с ростом размерности данных.

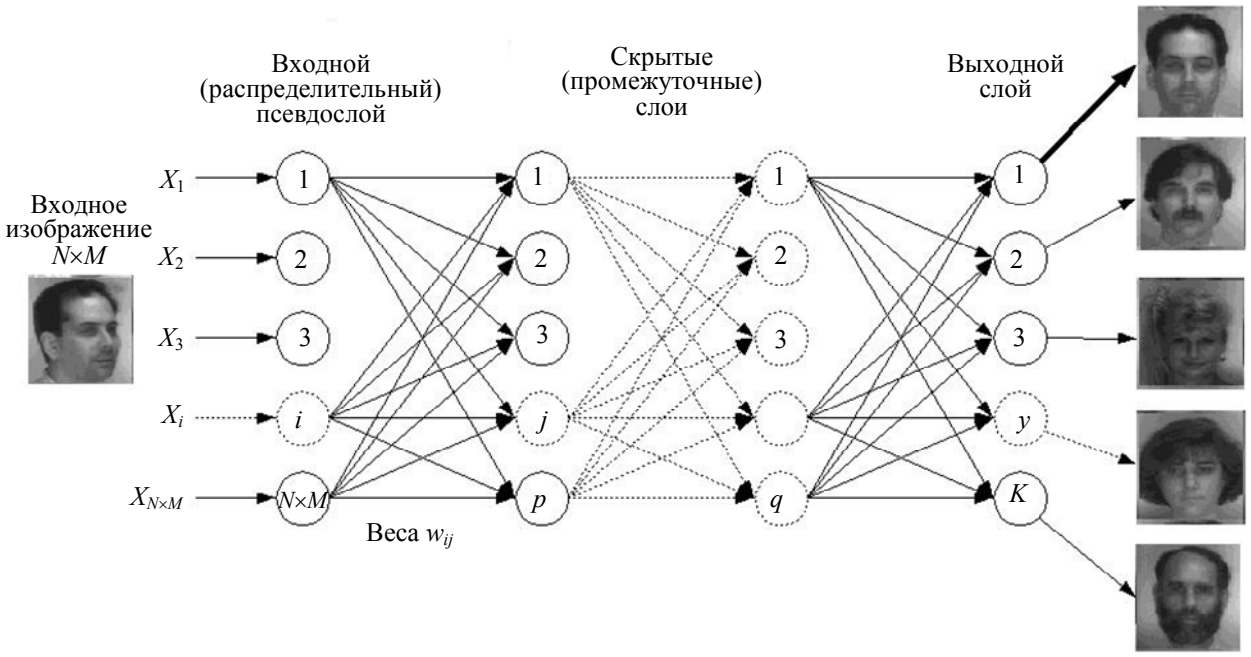


Рис. 1

Поскольку эталонные значения выходов известны, такой алгоритм относится к классу методов обучения с учителем. Применительно к извлечению ключевых признаков, когда происходит обучение сети реконструкции поданного на вход изображения, на скрытых нейронах сети формируется сжатое представление такого изображения, что может быть отнесено к классу методов самообучения.

Инициализация МНС перед началом обучения производится случайным выбором весовых коэффициентов. Поэтому две разные обученные НС, обеспечивающие одинаковые значения ошибки, часто могут быть представлены различными разделяющими поверхностями, не сводимыми друг к другу. На этом основан метод коллективов (ансамблей) нейронных сетей, часто применяемый при распознавании по изображению лица: создается набор (коллектив) сетей, обученных решать одну и ту же задачу различными способами. Обобщенное, полученное таким методом решение точнее и надежнее, чем решение единственной нейронной сети.

Нейронные сети высокого порядка (НСВП) отличаются от МНС наличием одного слоя, на входы нейронов поступают также совокупности сигналов, которые могут рассматриваться как термы высокого порядка, являющиеся произведением двух или более компонентов входного вектора. Например, для сетей второго порядка такой вектор обеспечивает функционирование в соответствии с выражением [4]:

$$S = \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij} x_i x_j - T.$$

Такие сети могут формировать сложные разделяющие поверхности, добавив компоненты входного вектора в произведение, получим класс полиномиальных разделяющих поверхностей. Сети также можно обучать по методу обратного распространения. Использование

МНС в общем случае эффективнее, но существует ряд приложений, в которых сети высокого порядка лучше.

Особенность НСВП заключается в том, что для распознавания некоторому классу достаточно предъявить обобщенный образ изображения без вариаций масштабов и поворотов. После обучения сеть будет распознавать известные классы инвариантно к масштабу и поворотам изображения. Такая сеть не является полносвязной, она характеризуется высокой обучаемостью и быстродействием. Точность классификации такой сетью различающихся масштабом и углом поворота изображений выше по сравнению с МНС [5, 6].

Радиально-базисные нейронные сети (РБНС) состоят из двух слоев (рис. 2). Первый слой описывается радиально-базисной активационной функцией:

$$y = \exp\left(\frac{-S^2}{2\sigma^2}\right),$$

где σ — среднеквадратичное отклонение, определяющее размер кластера, S — расстояние между вектором входных сигналов и сформированным вектором весовых коэффициентов W:

$$S^2 = | X - W |^2 = \sum_i (x_i - w_i)^2 .$$

Значение S определяет расстояние до центра кластера от исходного изображения на входе конкретного нейрона [7]. Второй (скрытый) слой представляет собой набор кластеров в пространстве образов и реализует первый этап кластеризации входного образа — значение активационной функции каждого нейрона быстро уменьшается с удалением от центра кластера. Последующий слой нейронов может быть описан линейной активационной функцией, он реализует второй этап кластеризации — распределяет кластеры по классам.

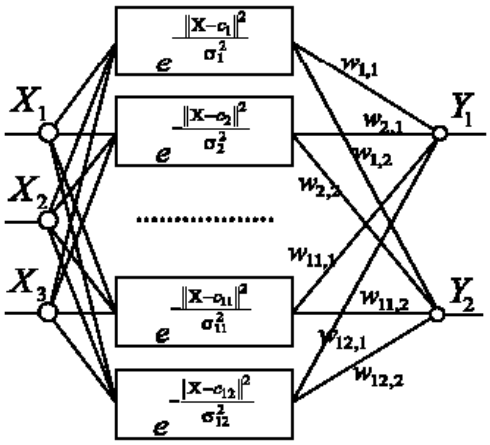


Рис. 2

РБНС позволяют строить плохо разделяющиеся области и аппроксимировать многомерные функции. По сравнению с многослойной нейронной сетью РБНС обучается на порядок быстрее, однако обладает намного худшей экстраполирующей способностью, т.е. не способна распознавать образы, значительно отличающиеся от образов-эталонов. Размерность РБНС больше, чем МНС, предназначенных для решения аналогичных задач, поэтому эффективность РБНС уменьшается с ростом размерности входных данных [1, 7].

Обучение такой сети происходит в два этапа: на первом — без учителя: первый слой выделяет компактно расположенные группы кластеров, при этом происходит корректировка центров кластеров. На втором этапе второй слой учится распределять по классам входные образы, пропущенные через первый слой. Если известны эталонные значения выходов, обучение обеспечивается матричными методами или алгоритмом обратного распространения ошибки. Рассмотренные типы нейронных сетей — МНС и РБНС — позволяют учесть топологию пространства изображения. Их принципы работы основываются на разбиении

изображения на локальные области и иерархическом сопоставлении как их взаимного расположения, так и содержания. Такие сети наиболее перспективны для распознавания изображений.

Когнитрон. В основу функционирования когнитрона (рис. 3) положена модель зрительной коры мозга [4]. Каждый слой мозга реализует различные уровни обобщения — входной слой чувствителен к простым образам, таким как линии различной ориентации в плоскости, в то время как другие слои позволяют формировать более сложные, абстрактные и не зависящие от положения образы.

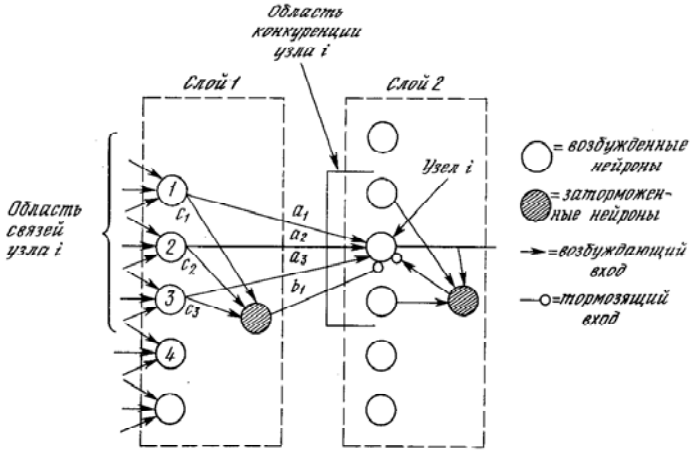


Рис. 3

Каждый нейрон когнитрона связан только с локальной областью предыдущего слоя, причем области могут взаимно перекрываться. Слоев в когнитроне обычно больше, чем в сетях других типов, таким образом достигается иерархическая организация. Высшие слои когнитрона реагируют на выделенные абстрактные признаки, поэтому на распознавание в меньшей степени влияют смещение и искажение исходного образа.

Неокогнитрон. В зрительной коре мозга человека были обнаружены группы нейронов (узлы), реагирующие на такие элементы, как линии и углы определенной ориентации. На более высоком уровне узлы реагируют на более сложные и абстрактные образы — окружности, треугольники и прямоугольники. С увеличением уровня степень абстракции возрастает до тех пор, пока не сформируются узлы, реагирующие на лица и другие сложные объекты. В общем случае узлы последующих уровней получают на вход результаты обработки группы низкоуровневых узлов и, следовательно, реагируют на более широкую область визуального поля. Реакции высокоуровневых узлов более устойчивы к искажениям исходного образа.

Неокогнитрон более точно, по сравнению с когнитроном, отражает строение зрительной коры и позволяет распознавать образы независимо от их преобразований: смещения, вращения, изменения масштаба и искажения [4]. Неокогнитрон может как самообучаться, так и обучаться с учителем. На вход неокогнитрона поступают двумерные образы, аналогичные изображениям, сформированным сетчатой оболочкой глаза, и обрабатываются аналогично зрительной коре мозга человека.

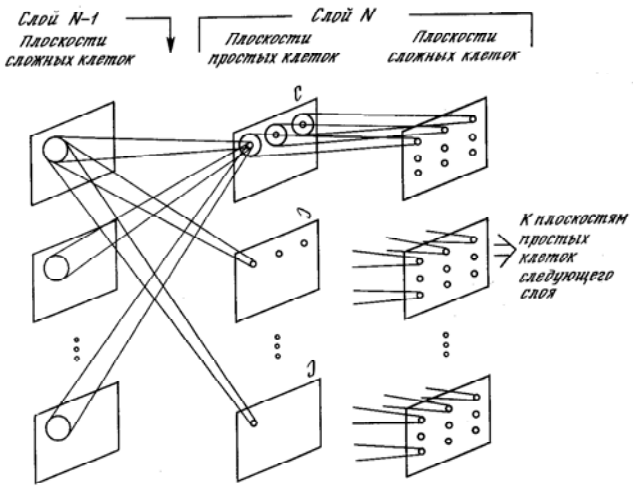


Рис. 4

Главное отличие неокогнитрона от когнитрона — двумерная организация локальных участков в виде иерархической структуры, состоящей из плоскостей (рис. 4).

Слои состоят из простых и сложных плоскостей. Каждый нейрон простой плоскости связан с локальным двумерным участком плоскостей предыдущего слоя, значения весовых коэффициентов всех нейронов в пределах одной плоскости одинаковы, и таким образом

плоскость реагирует на определенный образ, находящийся в участке изображения (на рис. 4 плоскости реагируют на букву „С“, вне зависимости от угла поворота). Местоположение активированного нейрона в простой плоскости определяет участок, в котором найден этот образ, независимо от его искажения.

Классический неокогнитрон является мощным средством распознавания изображений, однако требует больших, на сегодняшний день труднодостижимых, вычислительных затрат [4, 8, 9].

Сверточные нейронные сети (СНС). В классической многослойной нейронной сети межслойные нейронные соединения являются полносвязанными, изображение представлено в виде *n*-мерного вектора, не учитывающего ни двумерной локальной организации пикселов, ни возможностей деформации образа. Архитектура сверточной НС (рис. 5) позволяет преодолеть эти недостатки, в ней реализованы принципы архитектуры неокогнитрона, упрощенного и дополненного алгоритмом обучения с обратным распространением ошибки [8, 10].

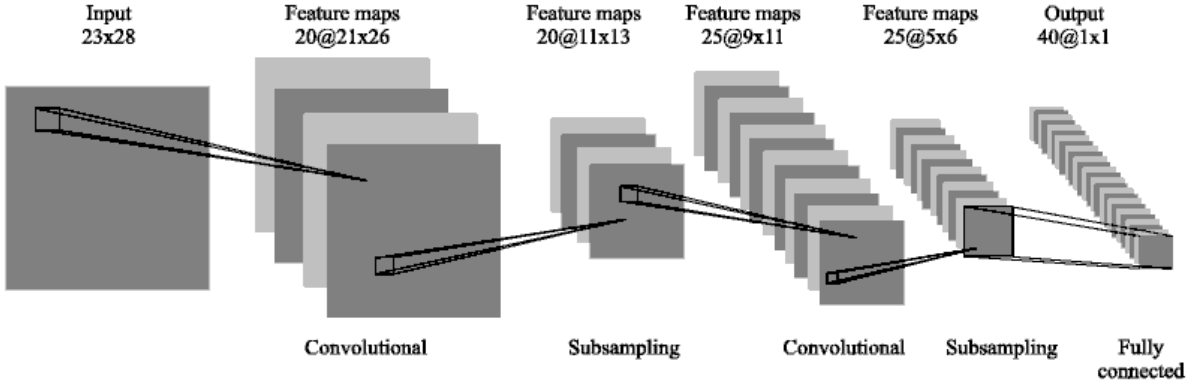


Рис. 5

В СНС используются локальные рецепторные поля (обеспечивают локальную двумерную связность нейронов), общие весовые коэффициенты (обеспечивают детектирование отдельных черт лица, находящихся в любом фрагменте изображения) и иерархическая организация с пространственными подвыборками (*Spatial subsampling*).

СНС обеспечивает частичную устойчивость к изменениям масштаба, смещениям, поворотам, смене ракурса и прочим искажениям. Архитектура СНС многослойна. Слои подразделяются на два типа: сверточные (*Convolutional*) и подвыборочные (*Subsampling*), чередующиеся друг с другом. В каждом слое имеется набор из нескольких плоскостей, причем нейроны одной плоскости имеют одинаковые весовые коэффициенты, поступающие ко всем локальным участкам предыдущего слоя (как в зрительной коре человека), изображение предыдущего слоя „сканируется“ небольшим окном и „взвешивается“ набором весовых коэффициентов, а результат отображается на соответствующий нейрон текущего слоя. Таким образом, плоскости называются картами характеристик (*feature maps*), каждая из них выделяет „свои“ участки изображения в любом месте предыдущего слоя. Следующий за сверточным подвыборочный слой уменьшает масштаб плоскостей за счет локального усреднения значений реакции слоя на выходах нейронов, таким образом достигается иерархическая организация СНС. Последующие слои извлекают более общие характеристики, меньше зависящие от искажений изображения [8].

Обучается СНС стандартным методом обратного распространения ошибки. Сравнение МНС и СНС показало существенные преимущества последней как по скорости, так и по надежности классификации. Полезным свойством СНС является и то, что характеристики, формируемые на выходах верхних слоев структуры, могут применяться для классификации по методу ближайшего соседа (например, при вычислении евклидова расстояния), причем СНС может успешно извлекать такие характеристики и для образов, отсутствующих в обучающем наборе. Для СНС характерны высокая скорость обучения и быстродействие.

Использование рассмотренных нейросетевых методов обеспечивает быстрое и надежное распознавание изображений. Однако применение этих методов к изображениям трехмерных объектов вызывает трудности, связанные с пространственными поворотами и изменением условий освещенности. Изображения при различных углах поворота объекта существенно различаются, при этом часть информации на изображении теряется и появляется новая информация, специфическая для данного угла.

Такая задача в общем виде для систем распознавания лиц еще не решена, но существуют методы, обеспечивающие решение отдельных ее аспектов (инвариантность к освещению, синтез повернутых в пространстве изображений лиц на основе обучения) [10—12].

СПИСОК ЛИТЕРАТУРЫ

1. *Панканти Ш., Болле Р. М., Джейн Э.* Биометрия: будущее идентификации // Открытые системы. 2000. № 3 [Электронный ресурс]: <http://www.osp.ru/os/2000/03/>.

2. *Foltyniewicz R.* Efficient High Order Neural Network for Rotation, Translation and Distance Invariant Recognition of Gray Scale Images // Lecture Notes in Computer Science - Computer Analysis of Images and Patterns. 1995. P. 424—431.

3. *Головко В. А.* Нейроинтеллект: Теория и применение. Кн. 1. Организация и обучение нейронных сетей с прямыми и обратными связями. Брест: БПИ, 1999. 260 с.

4. *Daughman J.* Face and Gesture Recognition: Overview // IEEE Transactions on Pattern Analysis and Machine Intelligence. 1997. Vol. 19. P. 675—676.

5. *Галушкин А. И., Томашевич Д. С., Томашевич Н. С.* Методы реализации инвариантности к аффинным преобразованиям двумерных изображений // Приложение к журналу „Информационные технологии“. 2001. № 1. С. 1—19.

6. *Giacinto G., Roli F.* Automatic Design of Multiple Classifier Systems by Unsupervised Learning // Lecture Notes in Artificial Intelligence - Machine Learning and Data Mining in Pattern Recognition. 1999. P. 131—143.

7. *Головко В. А.* Нейроинтеллект: Теория и применение. Кн. 2. Самоорганизация, отказоустойчивость и применение нейронных сетей. Брест: БПИ, 1999. 228 с.

8. *Lawrence S., Giles C. L., Tsoi A. C., Back A. D.* Face Recognition: A Convolutional Neural Network Approach // IEEE Transact. on Neural Networks, Special Issue on Neural Networks and Pattern Recognition. 1997. P. 1—24.

9. *Ranganath S. and Arun K.* Face recognition using transform features and neural networks // Pattern Recognition. 1997. Vol. 30. P. 1615—1622.

10. *Santaji G., Jayshree G., Shamla M., Dhanaji G.* Neural networks for facerecognition using SOM // IJCST. 2010. Vol. 1, Is. 2. P. 65—67.

11. *Thai Hoang Le.* Applying Artificial Neural Networks for Face Recognition // Hindawi Publishing Corporation, Advances in Artificial Neural Systems. 2011. P. 673 016.

12. *Saaidia M., Lelandais S., Vigneron V., El-Mouldi B.* Face detection by neural network trained with Zernike moments // Proc. of the 6th WSEAS Intern. Conf. on Signal Processing, Robotics and Automation. Corfu Island, Greece, 2007. P. 36—41.

Сведения об авторах

Андрей Александрович Тропченко — канд. техн. наук, доцент; Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики, кафедра вычислительной техники;
E-mail: zayka_98rus@mail.ru

Александр Ювенальевич Тропченко — д-р техн. наук, профессор; Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики, кафедра вычислительной техники;
E-mail: tau@d1.ifmo.ru

Рекомендована кафедрой вычислительной техники

Поступила в редакцию 08.02.12 г.

УДК 004.056.53

К. А. Щеглов, А. Ю. Щеглов

МОДЕЛЬ КОНТРОЛЯ ДОСТУПА К СОЗДАВАЕМЫМ ФАЙЛОВЫМ ОБЪЕКТАМ

Предложены метод контроля доступа к создаваемым файловым объектам, характеризующийся исключением сущности „объект доступа“ из разграничительной политики, и модель контроля доступа, с использованием которой разработаны требования к принудительному управлению потоками информации, обеспечивающие построение безопасной системы.

Ключевые слова: *защита информации, разграничение доступа, создаваемый файловый объект, контроль доступа, информационный поток.*

Введение. Основу защиты обрабатываемой компьютером информации от несанкционированного доступа составляет построение разграничительной политики доступа к файловым объектам, которую обеспечивает диспетчер доступа, перехватывающий и анализирующий все запросы от субъектов к объектам. Диспетчер однозначно выявляет в запросе доступа субъект и объект и на основании анализа заданного правила контроля принимает решение о предоставлении субъекту запрашиваемого доступа.

При назначении разграничений прав доступа первичен объект, поскольку именно он требует защиты от несанкционированного доступа. Однако применительно к решению задачи контроля доступа файловые объекты принципиально различаются — они могут быть подразделены на статичные (в первую очередь, системные) и создаваемые пользователями в процессе работы системы. Системные объекты присутствуют на момент настройки администратором прав доступа субъектов к объектам, а объекты второй группы еще не созданы. Возникает вопрос: как разграничивать доступ к еще не созданным объектам? А ведь эти объекты (прежде всего, файлы), особенно нуждаются в защите от несанкционированного доступа, поскольку содержат защищаемую конфиденциальную информацию.

Известные методы контроля доступа. Контроль доступа к статичным объектам осуществляется посредством назначения им атрибутов (прав доступа субъектов к объекту). Возможны два варианта назначения прав доступа к создаваемым объектам.

1. Права доступа задаются опосредованно, путем включения в систему на момент задания разграничительной политики доступа соответствующих статичных объектов — папок (своего рода „контейнеров“), доступ субъектов к которым, в том числе и по созданию в них новых объектов, разграничивается администратором. Вследствие разграничений пользователь может создавать новые файловые объекты (файлы) только в специально созданных папках. Создаваемые файловые объекты наследуют разграничения доступа от соответствующих включающих объектов (папок).

Как видим, собственно к создаваемым объектам контроль доступа не осуществляется, файл как сущность „исчезает“ из разграничительной политики доступа — используются субъект доступа и специально созданный контейнер — объект доступа, что определяет сложность задачи администрирования, многократно возрастающую, если в качестве субъекта доступа рассматривать не только учетную запись (пользователя), но и непосредственно процесс (приложение), без чего эффективную защиту в современных условиях не создать [1].

2. В разграничительной политике доступа используется сущность „Владения“, предоставляемая современными универсальными ОС: пользователь, создавший объект („Владелец“) наделяется полномочиями разграничивать права доступа к этому объекту для иных пользователей. В этом случае несмотря на то что права доступа в виде атрибутов назначаются именно

статичному объекту (владельцем, уже после создания им объекта), в определенном смысле можно говорить о контроле доступа к создаваемым файлам, поскольку права доступа на созданный файл назначаются пользователем-владельцем уже после задания администратором разграничительной политики доступа к объектам. Таким образом, изменяются собственно схема администрирования, способ задания разграничительной политики — реализуется непрерывное администрирование в процессе функционирования системы.

Однако функции администрирования при этом возлагаются уже не на администратора, а непосредственно на пользователя, что недопустимо в современных условиях.

Модель контроля доступа к статичным файловым объектам. Для оценки безопасности системы, основанной на контроле доступа к статичным файловым объектам, на практике используется модель Харрисона—Уззо—Ульмана [2]. Если считать, что $C = \{C_1,\ldots, C_l\}$ и $O = \{O_1,\ldots, O_k\}$ — соответственно линейно упорядоченные множества субъектов и объектов доступа, а $R = \{R_1,\ldots, R_m\}$ — конечное множество прав доступа (чтение, запись, удаление, исполнение и т.д.), то разграничительная политика доступа субъектов к объектам описывается матрицей доступа M, где M[C, O] — ячейка матрицы, которая содержит набор прав доступа субъекта из множества C к объекту из множества O. В любой момент времени система описывается текущим состоянием $Q = (C, O, \mathrm{M})$.

Требование к безопасности системы в рассматриваемом случае может быть сформулировано следующим образом: „Для заданной системы состояние $Q_0 = (C_0, O_0, \mathrm{M}_0)$ следует считать безопасным относительно некоторого права R, если не существует применимой к Q_0 последовательности действий, в результате выполнения которой субъектом C_0 приобретается право R доступа к объекту O_0, исходно отсутствующее в ячейке матрицы $\mathrm{M}_0[C_0, O_0]$“. Если право R, отсутствующее в ячейке матрицы $\mathrm{M}_0[C_0, O_0]$, приобретается субъектом C_0, то можно говорить, что произошла утечка права R, и относительно R его система небезопасна.

Поскольку последовательность действий генерирует субъект доступа, пользователь (действия администратора не имеет смысла рассматривать при анализе изменений состояния системы), то для рассматриваемой модели о безопасности системы можно говорить исключительно в предположении, что действия пользователя не приведут к утечке права R. Однако при реализации сущности „Владения“ именно пользователь, создавший объект, наделяет правом доступа R к этому объекту других пользователей. Как следствие, о безопасности системы в данном случае можно говорить исключительно в предположении, что субъект доступа (пользователь) не несет угрозы генерирования утечки права R с целью хищения, несанкционированной модификации или удаления обрабатываемой на предприятии информации. Однако в современных условиях, особенно применительно к корпоративным приложениям, где пользователь обрабатывает не собственную, а корпоративную информацию, и понятие „Владелец“ (не в технологическом смысле) к нему малоприменимо, нельзя сделать подобное предположение даже с большими оговорками. Например, в исследовании [3] сделан вывод о том, что наибольшую опасность для компаний сегодня представляет именно несанкционированный доступ собственных сотрудников к конфиденциальной информации предприятия.

Вывод. В безопасной системе контролем доступа должно осуществляться принудительное для пользователя управление потоками информации (или информационными потоками), сущность „Владения“ должна быть исключена из схемы контроля доступа.

Замечание. Перенос информации от C к O — информационный поток записи, от O к C — информационный поток чтения. Управление потоками — предоставление/изменение права R генерирования потока информации между C и O.

Правила управления потоками информации. Предлагаемый метод контроля доступа к создаваемым файловым объектам базируется на реализации принципов, изложенных в статье [4]. „Объект“ исключается из схемы реализации разграничительной политики — используются две сущности: идентификатор (учетная информация) субъекта, создавшего объект, и

идентификатор субъекта, запрашивающего доступ к созданному объекту. Новым файлом, созданным субъектом, наследуется учетная информация этого субъекта доступа. Учетная информация субъекта доступа, в общем случае используемая при реализации разграничительной политики, определяется следующими сущностями: исходный идентификатор пользователя (учетная запись, под которой осуществлен вход в систему); „полнопутевое“ имя исполняемого файла процесса, запрашивающего доступ к ресурсу; эффективный идентификатор пользователя (учетная запись, от которой осуществлен запрос доступа к ресурсу) [1].

При запросе доступа к любому файлу диспетчер доступа анализирует наличие, а при наличии — содержимое унаследованной файлом учетной информации создавшего его субъекта доступа. Матрица доступа здесь приобретает совершенно иной вид, поскольку из разграничительной политики доступа исключена сущность „Объект“. Если считать, что $C = \{C_1, \ldots, C_l\}$ — линейно упорядоченное множество субъектов доступа, а $R = \{R_1, \ldots, R_m\}$ — конечное множество прав доступа (r — чтение, w — запись, d — удаление, x — исполнение и т.д., 0 — отсутствие прав доступа) субъекта C_i к объекту, созданному субъектом C_j (i=1, …, l, j=1, …, l) то матрица доступа M, используемая для реализации разграничительной политики методом контроля доступа с принудительным управлением потоками информации имеет следующий вид (условимся в строках матрицы указывать учетную информацию субъектов, запрашивающих доступ к объектам, а в столбцах — учетную информацию субъектов, унаследованную созданными объектами):

$$\mathrm{M} = \begin{array}{c} \\ C_1 \\ C_2 \\ . \\ . \\ . \\ C_{l-1} \\ C_l \end{array} \begin{array}{c} \begin{array}{ccc} C_1 & C_2 & C_l \end{array} \\ \begin{bmatrix} r,w,d & w & 0 \\ r & r,w,d & 0 \\ & & \\ & ..... & \\ & ..... & \\ 0 & 0 & r \\ 0 & w & r,w,d \end{bmatrix} \end{array}.$$

В любой момент времени система описывается своим текущим состоянием $Q = (C,C,\mathrm{M})$, $\mathrm{M}[C,C]$ — ячейка матрицы, которая содержит набор прав доступа. Разрешение права доступа субъекта C_j к объектам, созданным субъектом C_i (i=1, …, l; j=1, …, l, где $R = \{x,w,R,d\}$), обозначим как $C_j(R)C_i$.

Вывод. Используя предложенный метод контроля доступа к создаваемым файловым объектам, можно построить безопасную систему, поскольку за счет контроля доступа осуществляется принудительное для пользователя управление потоками информации.

Сформулируем основные правила (требования) управления потоками, при реализации которых в системе отсутствует утечка права $R = \{x,w,r,d\}$, что не приводит к несанкционированному обмену информацией между субъектами, который будем обозначать $C_j[R]C_i$, $j \neq i$ (субъект C_j получает несанкционированный доступ к информации, обрабатываемой субъектом C_i):

1. Недопустимо разрешение пользователям права исполнения (x) файла, созданного в процессе функционирования системы. При выполнении данного требования система безопасна относительно x. Заметим, что выполнение именно этого требования обеспечивает эффективную защиту от вредоносных программ [5]. Как следствие, разрешенные права доступа к создаваемым файловым объектам: $R = \{w,r,d\}$.

2. При назначении разграничительной политики „по умолчанию“ должны быть установлены права доступа: $C_i(w,r,d)C_i$ (i=1, …, l). Данное правило обусловливает задание диаго-

нальной („канонической“ [1]) матрицы доступа, характеризуемой условием: $C_i(w,r,d)C_i$; $C_i(0)C_j$ ($i\neq j$, i=1, …, l, j=1, …, l). Реализующая каноническую модель доступа система безопасна относительно прав записи (w) и чтения (r).

3. При расширении канонической матрицы доступа правом чтения $C_i(r)C_j$ и при уже разрешенном в матрице праве чтения $C_k(r)C_i$, одновременно должно также разрешаться право чтения (r): $C_k(r)C_j$ ($i\neq j\neq k$, i=1, …, l; j=1, …, l; k=1, …, l), что предотвращает возможность несанкционированного обмена информацией между субъектами $C_k[R]C_j$, из-за возникающей при этом утечки права чтения $C_k(r)C_j$.

4. При расширении канонической матрицы доступа правом записи (w): $C_i(w)C_j$, при уже разрешенном в матрице праве записи (w): $C_j(w)C_k$, одновременно с этим должно также разрешаться право записи (w): $C_i(w)C_k$ ($i\neq j\neq k$, i=1, …, l; j=1, …, l; k=1, …, l), что предотвращает возможность несанкционированного обмена информацией между субъектами $C_i[R]C_k$, из-за возникающей при этом утечки права записи (w): $C_i(w)C_k$.

5. При расширении канонической матрицы доступа правом удаления (d) включающих объектов (папок, в которых создаются файлы), должно реализовываться следующее правило управления: любой включающий объект (папка) может быть удален любым субъектом при условии отсутствия включенных в него объектов (в первую очередь — файлов).

Замечание. Поскольку предлагаемый метод контроля доступа не предполагает реализации разграничительной политики доступа к статичным объектам — папкам — любой субъект может удалить любую папку, соответственно и все включенные в нее файлы, что приводит к утечке права удаления (d).

Таким образом, предложенный метод относится к дискреционным с принудительным управлением потоками информации. При реализации сформулированных требований метод позволяет построить безопасную систему.

Заключение. Использование предложенного метода контроля доступа к создаваемым файловым объектам позволяет пересмотреть принципы построения разграничительной политики доступа к защищаемым ресурсам, разделив задачи защиты статичных и создаваемых файловых объектов и способы их решения.

СПИСОК ЛИТЕРАТУРЫ

1. *Щеглов А. Ю.* Защита компьютерной информации от несанкционированного доступа. СПб: Наука и техника, 2004. 384 с.

2. *Harrison M., Ruzzo W., Ullman J.* Protection in operating systems // Communication of ACM. 1976. Vol. 19, N 8. P. 461—472.

3. [Электронный ресурс]: <http://www.securitycode.ru/company/news/SC-analytic-2011>.

4. *Щеглов К. А.* Принципы контроля доступа к создаваемым файловым объектам // Сб. тр. молодых ученых и сотрудников кафедры ВТ. СПб: НИУ ИТМО, 2012. Вып. 3. С. 85—86.

5. *Щеглов К. А., Щеглов А. Ю.* Защита от вредоносных программ методом контроля доступа к создаваемым файловым объектам // Вестн. компьютерных и информационных технологий. 2012. № 8. С. 46—51.

Сведения об авторах

Константин Андреевич Щеглов — студент; Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики, кафедра вычислительной техники; E-mail: schegl_70@mail.ru

Андрей Юрьевич Щеглов — д-р техн. наук, профессор; Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики, кафедра вычислительной техники; E-mail: info@npp-itb.spb.ru

Рекомендована кафедрой вычислительной техники

Поступила в редакцию 08.02.12 г.

УДК 510.5

В. И. Поляков, В. И. Скорубский

ПРЕОБРАЗОВАНИЕ МОДЕЛЕЙ АЛГОРИТМОВ

Рассматриваются преобразования регулярных выражений в конечные автоматы и обратные преобразования, относящиеся к теории алгоритмов и автоматов. Приведены преобразования этих моделей в блок-схемы и обратно.

Ключевые слова: *регулярные языки, регулярные выражения, конечные автоматы, модели алгоритмов, блок-схемы.*

Введение. Практика программирования предполагает решение задачи, а не получение доказательства того, что она разрешима. Вместе с тем в теории алгоритмов исследованы методы и модели, которые используются в программировании. К числу таких моделей относят регулярные языки и конечные автоматы. В практической реализации алгоритмов используются алгоритмические языки и блок-схемы. В настоящей работе рассматриваются преобразования моделей в блок-схемы и обратно.

Представить формальное описание алгоритма можно с помощью формального описания решаемой задачи, в то время как имеется неформальное (вербальное) описание задачи и соответственно переход к алгоритму будет также неформальным, и требуются верификация, тестирование и многократные итерации для приближения к допустимому решению.

Формализованное описание может быть получено с помощью регулярных языков [1, 2]. Ставится задача разработки алгоритма распознавания принадлежности фрагмента любого текста конкретному регулярному языку; доказывается, что регулярный язык может быть формально преобразован в *модель алгоритма* решения этой задачи за конечное число шагов. Такой моделью является *конечный автомат* (КА).

Расширения регулярных языков находят применение при описании лексики формальных алгоритмических языков, при описании алгоритмов редактирования, поиска по шаблону, в современном программировании (языки Perl, Java, Phyton, C#, PHP), в компиляторах и системах управления обработкой данных, реализованных в операционных системах. Следовательно, для алгоритмического решения таких задач может быть использован конечный автомат.

Преобразование регулярных выражений в конечные автоматы. Для любого регулярного языка, представленного регулярным выражением (РВ), можно построить КА — распознаватель слов, допустимых в языке.

Рассмотренная в работе [2] методика преобразования может быть упрощена с учетом алгебраических свойств операций регулярного выражения:

— последовательности символов, помещенных в скобки с операцией сложения, имеют общее начальное состояние перед открывающей скобкой и конечное после закрывающей;

— цикл может иметь произвольное число повторений. При пустом числе циклов (итераций) сохраняется начальное состояние;

— в конкатенации нескольких символов различимые состояния заменяют конкатенацией пары символов. Например, в следующем регулярном выражении определены состояния в правильных арифметических операторах и выбраны позиции — состояния КА [1]:

$$L(M)=(ab + b)((c + a)b)^*,$$
$$0\ \ 12\ 2\ \ 2\qquad 32$$

где $L(M)$ — язык, допускаемый автоматом M; a, b, c — символы входного языка КА; $((c + a)b)^*$ — множество всех цепочек, начинающихся с символов c или a и оканчивающихся символом b; 0 — начальное, 2 — конечное, 1 и 3 — промежуточные состояния.

Общие оценки числа состояний и переходов КА определяются следующим образом:

— число переходов (ребер графа) КА равно n+1, где n — число букв в регулярном выражении (в данном примере n=6);

— в полностью определенном КА нижнюю границу числа состояний m определяют из условия ms=n+1, где s — число символов входного алфавита (в данном случае s=3, $3m$=7 и m=]7/3[=3);

— если ms>n+1, то автомат является частично определенным и верхняя граница числа состояний $k \le n$+1 — количество позиций в регулярном выражении (в данном примере k=7).

Состояния размещаются в выбранные для них позиции и формируются тройки (q_i w q_j), обозначающие переходы из состояния q_i в q_j, w — символ входной цепочки символов предложения регулярного языка:

$$L(M)=\underset{0}{(}\underset{1}{ab}+\underset{2}{b}\underset{2}{)}((c+a)b)^* = ((0a1)(1b2)+(0b2))\,((2(c+a)3)(3b2))^* =$$

$$=0(a1b+b)2((c+a)3b2)^*.$$

Линейная помеченная строка символов заменяется графом КА. На рис. 1 приведен детерминированный КА, распознающий символы языка $L(M)$. Начальное состояние 0 указано направленной на него стрелкой, а конечное 2 обведено жирным кружком.

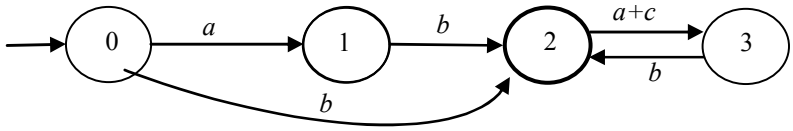


Рис. 1

Состояния q_i и q_j связаны дугой (q_i, q_j), если существует смежный символ в отмеченном регулярном выражении, и дуга помечается этим символом.

Преобразование недетерминированного конечного автомата в детерминированный. В общем случае регулярное выражение может быть преобразовано в недетерминированный конечный автомат (НДКА).

Пример. Рассмотрим следующее регулярное выражение:

$$L(M)=\underset{0}{a}\underset{1}{a}^*\underset{2}{b}d^* + ad^* = (aa^*b + a)d^* =(0a1(a1)^*b + 0a)2(d2)^*.$$

Недетерминированный КА, распознающий входные цепочки символов языка $L(M)$, представлен на рис. 2.

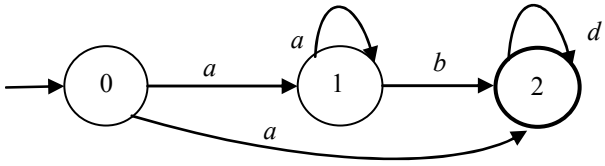


Рис. 2

Утверждение. Для НДКА можно построить эквивалентный ДКА.

Преобразование по методике, описанной в работе [2], можно представить следующей упрощенной процедурой:

1) на i-м шаге множество состояний ДКА обозначим Q_i. Q_0 обозначает множество состояний в НДКА;

2) находим различимые подмножества следующих состояний для всех условий $\sum$, применяемых к Q_i, и включаем их в множество Q_{i+1}. Итерации повторяются, пока $Q_i \ne Q_{i+1}$;

3) на каждом шаге недетерминированные переходы в состояния $\{q_i\}$ и $\{q_j\}$ становятся детерминированными при объединении их в одно состояние $\{\{1\},\{2\}\}$. Детерминированные выходы из $\{\{1\},\{2\}\}$ сохраняются.

Лемма. Число состояний в ДКА не превышает 2^m–1, где m — число состояний в НДКА.

Число 2^m–1 — количество собственных подмножеств множества, состоящего из m различимых элементов. Следовательно, процедура конечна.

Выполним преобразование НДКА в ДКА для автомата, представленного на рис. 2:

$$Q_0 = \{0, 1, 2\},$$
$$Q_1 = \{0, 1, 2, \{1, 2\}\}.$$

Эквивалентный детерминированный автомат содержит четыре состояния и определен тем же регулярным выражением $L(M')=L(M)$, состояния {2} и {1, 2} включают финальное состояние {2} из НДКА и становятся финальными в ДКА. Недетерминированные переходы в состояния {1} и {2} становятся детерминированными при объединении их в {1, 2}. Детерминированные выходы из {1, 2} сохраняются (рис. 3).

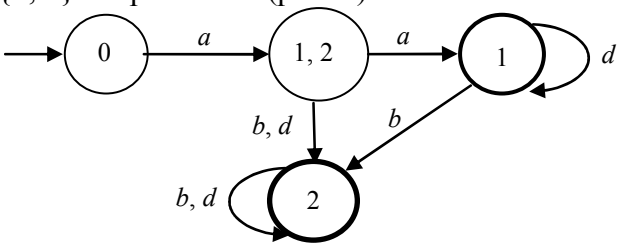


Рис. 3

Преобразование детерминированного конечного автомата в регулярное выражение может быть полезно, если КА является естественной формой описания алгоритма в задачах логического управления или получен преобразованием блок-схем.

Конечные автоматы могут применяться в качестве моделей алгоритмов для непосредственного программирования задач управления объектами [3].

К регулярному выражению применяются алгебраические преобразования и повторное обратное преобразование, позволяющее получить другие эквивалентные модели алгоритма (НДКА, минимальные ДКА и др.).

Пример. Дан КА (рис. 4).

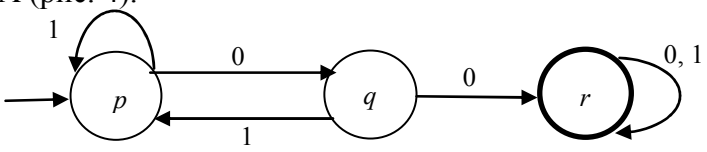


Рис. 4

Рассмотрим метод прямого преобразования КА в регулярное выражение:

1) выбрать циклы и для них последовательно представить дуги тройками вида $(q_i\, w\, g_j)$;

2) смежные дуги заменить последовательно тройками

$$L(M) = p(p1p)^*(p0q1p)^*p0q0r(r(0+1)r)^*;$$

3) пропустить первое состояние и сохранить полученное выражение

$$L(M) = p(1p)^*(0q1p)^*p0q0r((0+1)r)^*;$$

4) отметки-состояния рассматривать как позиции в регулярном выражении и выписать их

$$L(M) = p(1p)^*(0q1p)^*p0q0r((0+1)r)^* =$$
$$p\ \ p\quad\ q\ p\ \ p\ q\ r\quad\ \ r$$
$$= (1)^*(0\ 1)^*0\ 0(0+1)^* = (1^*+0\ 1)^*0\ 0(0+1)^*.$$
$$p\ \ p\ \ q\ p\ p q\ r\quad\ \ r\quad\ \ p\quad p\ q\quad p\ q\ r$$

Преобразование блок-схемы алгоритма в конечный автомат. Используя конструктивный метод преобразования блок-схем алгоритмов в КА [4], выполним преобразование на примере блок-схемы алгоритма умножения „в столбик“ $S=A*B$.

Содержательное описание алгоритма умножения:

— множимое $A = (a_{n-1}\, a_{n-2} \ldots a_1\, a_0)$ — n-разрядное десятичное число;

— множитель $B = (b_{n-1}\, b_{n-2} \ldots b_1\, b_0)$ — n-разрядное десятичное число;

— произведение S — $2n$-разрядное десятичное число, которое вычисляется суммированием частичных произведений со сдвигом влево на один десятичный разряд:

$$S = \begin{array}{r} A \bullet b_0 \\ A \bullet b_1 \\ \ldots\ldots\ldots\ldots\ldots \\ A \bullet b_n \\ \hline S_{2n-1}\, S_{2n-2} \ldots S_0. \end{array}$$

В результате получена рекуррентная формула вычисления частичных произведений $S_{i+1} = S_i + 10\,(A \bullet b_i)$. Предполагается, что вычисление частичных произведений, сдвиг и суммирование — эффективные операции. Блок-схема алгоритма умножения приведена на рис. 5.

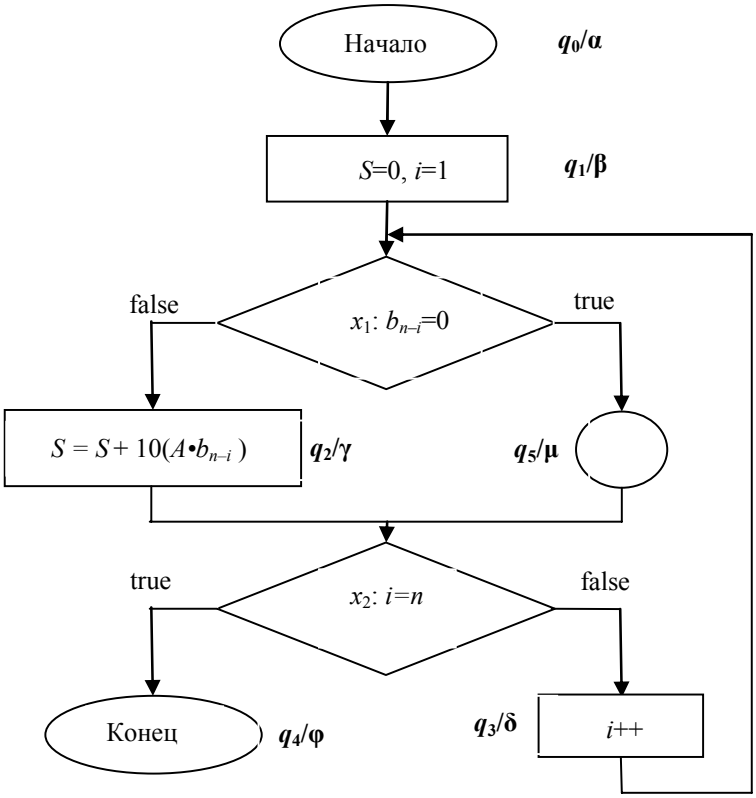


Рис. 5

Операторные вершины, а также начальная (ввод) и конечная (вывод) обозначаются как состояния КА, они обеспечивают *задержку*, необходимую для выполнения соответствующих операций. Задержки-состояния можно выбирать также для упрощения условий ветвления. В данном случае для этой цели можно включить промежуточное состояние q_5.

Таким образом, получено множество состояний $Q = \{q_0, q_1, ..., q_4, q_5\}$. Входной алфавит определяет множество символов, кодирующих обозначенные в условных вершинах предикаты и принимающие двоичные значения {true, false}. Переходы из одного состояния в другое могут быть представлены конъюнкциями значений входных переменных $\{x_1, x_2\}$.

Функции переходов КА определены в таблице.

q	q_1	q_2	q_3	q_4	q_5
q_0	T	—	—	—	—
q_1	—	$\neg x_1$	—	—	x_1
q_2	—	—	$\neg x_2$	x_2	—
q_3	—	$\neg x_1$	—	—	x_1
q_4	—	—	—	—	—
q_5	—	—	$\neg x_2$	x_2	—

Функции выходов, формируемых в состояниях, обозначим символами-командами выходного алфавита $W=\{\alpha, \beta, \delta, \gamma, \varphi, \mu\}$ и поставим им в соответствие состояния КА. Очевидно, что модель алгоритма в виде КА легко преобразуется в блок-схему, следовательно, всегда может быть выполнен переход к формальному описанию алгоритма на алгоритмическом языке и в виде программы.

При преобразованиях алгоритмов, представленных блок-схемами, могут быть использованы *частичные КА*.

Особенности использования частичных автоматов в программировании зависят от типа задачи, решаемой алгоритмическим методом:

— для распознавателей языка — ограничение на регулярный язык;

— для алгоритмов преобразования данных — ограничение на область значений данных и результатов выполнения преобразований;

— для алгоритмов управления — ограничение на входные данные.

В случае распознавателей предложений языка *при тестировании* признаки частичных автоматов могут быть использованы для контроля и выявления предложений, не принадлежащих языку.

Доопределение переходов можно использовать для контроля и тестирования программы в процессе исполнения. Для этого выполняется замена логических условий символами входного алфавита $\Sigma^*=\{T, a, b, c, d\}$, где $a= \neg x_1$, $b= x_1$, $c= \neg x_2$, $d= x_2$, q_0 — начальное и q_4 — финальное состояние:

$$L(M) =(0T1)\{[(1a2)(((2c3)(3a2))^*(2d4)+(2c3)((3b5)(5c3))^*(3b5)(5d4)] +$$

$$+(1b5)[((5c3)(3b5))^*(5d4)+(5c3)((3b5)(5c3))^*((3a2)(2c3))^*(3a2)(2d4)]\}=$$

$$=T[a((ca)^*d+c(bc)^*bd)+b((cb)^*d+c(bc)^*(ac)^*ad]=$$

$$=T[a((ca)^*+c(bc)^*b)+b((cb)^*+c(bc)^*(ac)^*a]d=$$

$$=T[a((ca)^* +c(bc)^*b) +b((cb)^* +c(bc)^*(ac)^*a]d=$$

$$= T[a((ca)^* +tb) +b((cb)^* +t(ac)^*a]d,$$

t — переводит КА в состояние, которое запускает КА, распознающий строку $c(bc)^*$.

Модель алгоритма умножения в символах входного алфавита приведена на рис. 6.

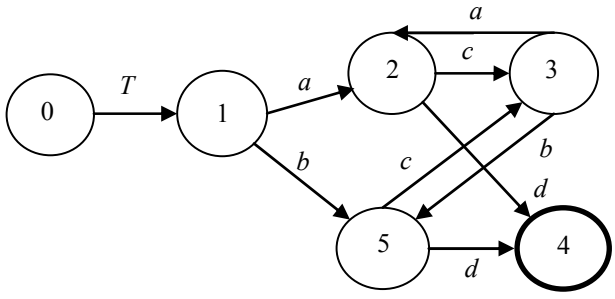


Рис. 6

Заключение. Алгоритм может быть представлен в формальной записи – конечными автоматами, синтаксическим разбором, рекурсивными функциями, регулярными выражениями. В работе рассмотрены методы преобразования моделей алгоритмов: регулярных выражений в конечные автоматы; недетерминированного конечного автомата в детерминированный; детерминированного конечного автомата в регулярное выражение; блок-схем в конечные автоматы и обратно. С помощью приведенных преобразований можно проектировать программы вычислительных машин. Скомпилированные по этим моделям программы не требуют верификации, так как получение этих моделей является доказательством существования алгоритма.

Работа выполнена при поддержке Российского фонда фундаментальных исследований (грант № 12-07-00376-а).

СПИСОК ЛИТЕРАТУРЫ

1. *Ахо А.., Хопкрофт Дж., Ульман Дж.* Построение и анализ вычислительных алгоритмов. М.: Мир, 1979. 536 с.

2. *Хопкрофт Дж., Мотвани Р., Ульман Дж.* Введение в теорию автоматов, языков и вычислений. М.: Изд. дом „Вильямс“, 2002. 528 с.

3. *Шалыто А. А.* Switch-технология. Алгоритмизация и программирование задач логического управления. СПб: Наука, 1998. 628 с.

4. *Майоров С. А., Новиков Г. И., Немолочнов О. Ф.* и др. Проектирование цифровых вычислительных машин. М.: Высш. школа, 1972. 344 с.

Сведения об авторах

Владимир Иванович Поляков — канд. техн. наук, доцент; Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики, кафедра вычислительной техники; E-mail: v_i_polyakov@mail.ru

Владимир Иванович Скорубский — канд. техн. наук, доцент; Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики, кафедра вычислительной техники; E-mail: vlis@km.ru

Рекомендована кафедрой вычислительной техники

Поступила в редакцию 08.02.12 г.

КОМПЬЮТЕРНЫЕ СИСТЕМЫ И СЕТИ

УДК 004.414.23

Т. И. Алиев, В. В. Соснин, Д. Н. Шинкарук, М. Ю. Тихонов, Н. Г. Бурмакин

САПР МАРШРУТИЗИРУЕМОЙ КОМПЬЮТЕРНОЙ СЕТИ НА ОСНОВЕ КОМПОНЕНТОВ С ОТКРЫТЫМИ ИСХОДНЫМИ КОДАМИ

Рассматриваются проблемы разработки САПР маршрутизируемых компьютерных сетей, обеспечивающих diffServ-сервисы QoS, с использованием компонентов с открытым исходным кодом на примере систем NS-3 и Qt Framework.

Ключевые слова: *NS-3, QoS, Qt, имитационное моделирование, компьютерная сеть, туннелирование, фильтрация, дисциплины обслуживания, WFQ, PQ, CQ, LLQ.*

Введение. Имитационные модели компьютерных сетей используются для обучения [1], проектирования сетей в составе САПР [2] и исследования сетевых технологий [3]. Необходимость разработки САПР маршрутизируемой компьютерной сети связана с отсутствием в настоящее время готовых программных решений, которые удовлетворяют следующим требованиям:

— открытость исходного кода всех компонентов, что позволяет модифицировать базовые возможности САПР, реализуя сколь угодно сложную логику обработки пакетов;

— компоненты САПР должны иметь лицензии GPL, LGPL или BPL и быть работоспособны в системе Linux, что снижает общую стоимость САПР;

— использование языка C++, что позволяет реализовать в САПР поддержку новых сетевых технологий за счет применения существующих программных кодов стеков протоколов Linux (например, BSD sockets);

— при обработке пакетов должны быть реализованы сложные функции OSI-модели, которые отсутствуют в стандартных пакетах моделирования сетей;

— наличие кроссплатформенного графического пользовательского интерфейса (GUI), содержащего средства для анализа результатов моделирования с детальными сведениями о динамике изменения исследуемых характеристик.

Постановка задачи. Целью работы являлось создание САПР маршрутизируемой компьютерной сети на основе компонентов с открытым исходным кодом. Разрабатываемая САПР должна удовлетворять перечисленным выше требованиям. Для достижения поставленной цели решались следующие задачи:

— выбор средства разработки ядра САПР и его GUI с учетом того, что ядро должно обеспечивать возможность применения статистического моделирования при имитации работы компьютерной сети;

— организация программного взаимодействия между ядром САПР и GUI;

— выбор состава характеристик функционирования компьютерной сети, необходимых для решения задач анализа и синтеза проектных решений, и метода их обработки;

— программная реализация разработанных методов с использованием выбранных средств программирования.

Выбор средства разработки ядра САПР. В работе [4] предлагается для моделирования компьютерных сетей использовать коллекцию библиотек NS-3 (основанную на C++), которая позволяет реализовать большинство базовых функций сети передачи данных. Однако в NS-3 *отсутствуют* штатные средства для программной реализации следующих функций, технологий и требований:

— моделирование отказов линий связи или маршрутизаторов с корректной реакцией узлов сети на отказы, включая реконфигурацию маршрутных таблиц;

— реализация корректного функционирования IP-IP-туннелей;

— реализация фрагментации TCP-трафика в узлах сети;

— полноценное функционирование протокола OSPF, включая рассылку служебных сообщений и реконфигурацию графа сети;

— организация приоритетных очередей с правилами обслуживания PQ, WFQ, LLQ, RPQ+ и CQ [1] с возможностью гибкой классификации трафика;

— расчет по результатам моделирования следующих характеристик: среднее значение и вариация задержки передачи пакетов на каждом маршрутизаторе сети и на всем пути следования пакетов; доля потерь пакетов из-за переполнения буферов, фильтрации или помех в канале связи; размер окна и количество повторных передач для пакетов TCP-потоков.

Для преодоления этой проблемы потребовалась модификация базовых возможностей системы NS-3.

Выбор средства разработки GUI в первую очередь определялся наличием в среде разработки поддержки языка C++, поскольку это позволило использовать в ядре САПР и GUI одинаковые структуры данных в разделяемых файлах с исходными кодами. В таблице приведены наиболее известные кроссплатформенные сетевые симуляторы, имеющие графический пользовательский интерфейс, анализ которых позволил выбрать среду разработки Qt Framework. Эта среда обладает следующими преимуществами:

— известен ряд успешных реализаций сетевых симуляторов в Qt;

— стремительно развивается в OpenSource-среде на протяжении многих лет;

— стандарты Qt поддерживаются крупными компаниями и сообществами;

— является хорошо документированной системой;

— в Qt существуют развитые средства разработки (отладчики, профайлеры);

— изначально проектировалась как кроссплатформенный продукт.

Сетевой симулятор	Язык программирования	Используемая библиотека для создания GUI
GNS-3	Python	Qt
OMNET++ / OMNEST	C++, Tcl	Tk
GloboSim / QualNet	C++	Qt
SSFNet	Java	Swing, JGraph
J-Sim	Java	Swing, JGraph
PacketTracerCisco	C++	Qt
CNet	C, Tcl	Tk
STPSim	Python	wxWidgets
OpenWNS	C++, Python	Qt
NetAnimator	ActionScript	AdobeFlash
NetSim	Java	Swing, JGraph

Кроме приведенных в таблице сред разработки были рассмотрены GTK+, .NET Framework, Ultimate++, Motif, FoxToolkit. Все они в той или иной мере не обеспечивают выполнения перечисленных требований.

Взаимодействия ядра модели и GUI. Разработанная модель, структура которой приведена на рис. 1, состоит из трех исполняемых файлов, два из которых (Конфигуратор и Плеер) разработаны с помощью Qt, а третий (Ядро) — с помощью NS-3. Конфигуратор используется в качестве GUI для изменения параметров модели, Плеер используется в качестве GUI для анализа результатов моделирования. Стрелками показано направление движения входных/выходных данных каждого из исполняемых файлов. Например, созданный с помощью Конфигуратора файл описания модели поступает на вход Ядра, а созданный Ядром файл с историей моделирования поступает на вход в Плеер, в котором можно отследить изменение исследуемых характеристик с течением времени.

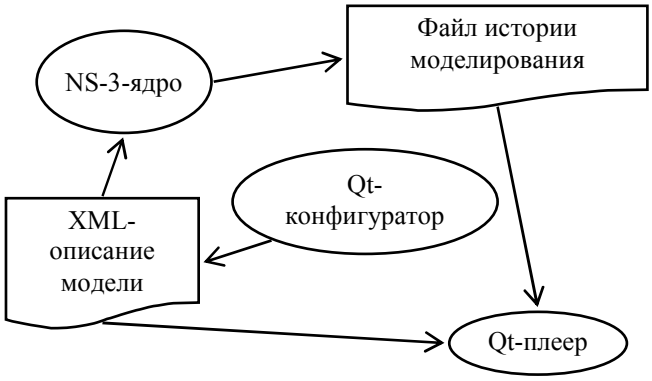


Рис. 1

В файл с историей моделирования записываются временные срезы состояния модели. Важной особенностью такого способа обмена данными между Ядром и Плеером является то, что данные передаются после их окончательного формирования, а не в режиме реального времени при работе имитационной модели. Такой подход обладает следующими достоинствами.

1. Модульность. Разработка Ядра, Конфигуратора и Плеера может выполняться независимо, с использованием описанных файлов как интерфейса между ними.

2. Обратная совместимость. Модель совместима с новыми версиями Qt и NS-3.

3. Возможность работы в демонстрационном режиме. Модель легко трансформировать для работы в этом режиме, для чего достаточно заготовить демонабор файлов с историей моделирования, которые предоставляются потенциальному покупателю со всеми компонентами модели, кроме Ядра.

4. Воспроизводимость. Историю моделирования можно неограниченное количество раз просматривать постфактум на компьютере любой производительности, что невозможно при анализе данных в процессе моделирования „на лету“ (как делается в большинстве имитационных пакетов), поскольку моделирование больших моделей является очень ресурсоемкой задачей.

5. Гибкость процесса анимации, т.е. возможность осуществлять его с любым разрешением по времени, что невыполнимо, если анимация производится одновременно с процессом моделирования.

Модификация базовых возможностей NS-3. Стандартные объекты NS-3 не позволяют реализовать все перечисленные высокоуровневые функции OSI-модели, поэтому при создании САПР на основе NS-3 требуется создавать собственные реализации сетевых устройств, интерфейсов и каналов связи. Из-за этого становится невозможным использование вспомогательных модулей NS-3 — Помощников (Helper), которые скрывают от программиста низкоуровневую работу с множеством объектов модели. Отказ от использования Помощников приводит к усложнению программирования, увеличению количества программного кода и снижению его удобочитаемости. Например, объем программы увеличивается более чем в 3 раза, если для конфигурирования канала связи не использовать Помощник.

Проблема фрагментации TCP-дейтаграмм. При реализации поддержки TCP-соединений между конечными узлами моделируемой сети было обнаружено, что в NS-3 все посылаемые через TCP-сокет пакеты дробятся на фрагменты размером не более 576 байт. Такое „несанкционированное“ дробление приводит к возникновению ошибок при дефрагментации пакетов в маршрутизаторах. Изучение исходных кодов реализации протокола TCP позволило понять, что разработчики NS-3 жестко придерживались стандарта RFC 791 (Internet Protocol [5]), где указана рекомендованная длина посылаемого узлом пакета. При фрагментации TCP-пакетов возникает и другая проблема. Если необходимость фрагментации связана с превышением MTU порта маршрутизатора, то дальнейшую обработку разбитых на фрагменты пакетов можно вести одним из следующих способов:

— присвоить пакетам виртуальные идентификаторы, которые будут хранить информацию о фрагментации;

— изменить алгоритм обработки пакета системой NS-3 в рамках стека протоколов TCP/IP, внеся правки в исходный программный код TCP/IP-стека;

— реализовать сборку фрагментированных пакетов в собственном модуле, который затем подключается в качестве промежуточного уровня TCP/IP-стека;

— использовать стандартные поля заголовка IP-уровня для сохранения информации о фрагментации.

Наилучшим из перечисленных способов является последний — хотя такая схема несколько усложняет фрагментацию и сборку пакетов, она совместима с протоколом [5] и может быть частично реализована встроенными методами NS-3, что упрощает поддержку и дальнейшее развитие модели.

Реализация TCP-сокетов стандартными средствами NS-3. При этом возможны два решения: устанавливать сокеты в рамках одного приложения ими рамках независимых приложений. Анализ показал, что для сбора детальной статистики временных задержек передачи пакетов лучшим является второе решение, так как в этом случае упрощается процедура анализа получаемых конечным узлом пакетов, а также возможно подключить несколько генераторов трафика к одному приемнику. Недостатком такого подхода является то, что в приемнике трафика потребуется установить большое количество приложений, что повысит затраты оперативной памяти при моделировании.

Реализация алгоритма поиска кратчайшего пути OSPF. В разработанной САПР компьютерных сетей маршрутизаторы связаны по схеме point-to-point (P2P), что существенно упрощает реализацию алгоритма маршрутизации. Поэтому возникла задача разработки упрощенной (модифицированной) версии протокола динамической маршрутизации OSPF, которая позволила бы получить те же качественные и количественные результаты, что и полная версия, описанная в стандарте RFC 2328 [6], но при меньших затратах вычислительных ресурсов. Для этого необходимо было выявить основные особенности работы OSPF в сетях point-to-point, найти в протоколе избыточные алгоритмы и неиспользуемые структуры данных в служебных пакетах. Работа протокола OSPF схематично представлена на рис. 2. Анализ показал, что без потери адекватности разрабатываемой версии протокола (по отношению к стандартной) могут быть опущены следующие части протокола OSPF v.2:

— все алгоритмы, связанные с выбором, проверкой работоспособности и функционированием „назначенного“ и „резервного назначенного“ маршрутизаторов (DesignatedRouter и BackupDesignatedRouter — DR и BDR);

— обмен hello-сообщениями и анонсами состояния связей (Link State Advertisement — LSA) по протоколу multicast.

Протокол OSPF содержит несколько типов служебных пакетов, наиболее длинным и сложным по структуре из которых является пакет обновления состояния связей типа 4 — Link State Update (LSU). Длина этого пакета примерно на порядок больше длины любого дру-

гого служебного пакета, поэтому при создании упрощенной версии OPSF имеет смысл модифицировать алгоритм обработки полей только пакетов LSU.

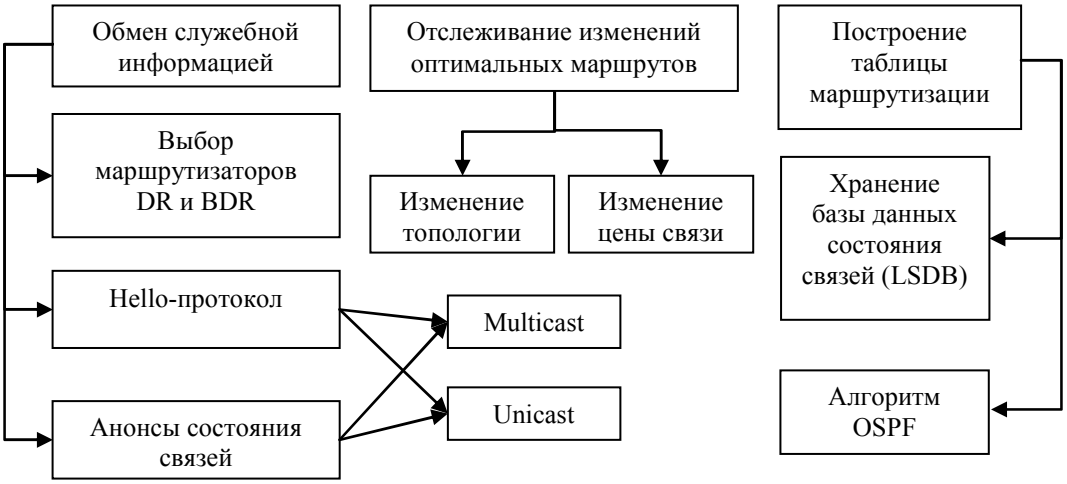


Рис. 2

Модификация протокола OSPF при обработке служебных пакетов позволяет увеличить скорость работы имитационной модели, поскольку дает возможность не рассматривать поля заголовка, которые не обязательны для корректного функционирования в условиях P2P-соединений между маршрутизаторами. Проведенный анализ стандарта [6] показал, что для получения кратчайших маршрутов достаточно следующих полей LSU-пакетов: тип сообщения (1 байт); длина сообщения (2 байта); идентификатор маршрутизатора (4 байта); номер последовательности (2 байта); количество анонсов (4 байта); количество портов (2 байта); адрес порта (4 байта); адрес соседа (4 байта); метрика (4 байта). Сокращение числа заполняемых и анализируемых полей с 27 до 9 позволило упростить протокол OSPF, удалив из него все алгоритмы, связанные с обработкой исключенных из рассмотрения полей. Для обеспечения адекватности модифицированного OSPF было решено пакет, отправляемый в канал, дополнять после перечисленных полей произвольной „набивкой“, равной по длине разнице в байтах между длиной пакета реализованного в модели протокола и стандартного. Таким образом, модифицированный OSPF, корректно рассчитывая маршрутную информацию, не влияет на временные задержки передачи LSU в каналах связи.

Анализ результатов моделирования. Важной особенностью модели является гибкая система сбора результатов ее работы, позволяющая пользователю индивидуально указывать классы трафика, характеристики которых требуется проанализировать. Это реализуется с помощью определения набора фильтров вида: „АИ, МИ, АН, МН, ПИ, ПН, Т, ВИП“, где АИ/МИ — IP-адрес и маска источника, АН/МН — IP-адрес и маска назначения, ПИ/ПН — порт источника/назначения, Т — тип сервиса в поле ToS (type of service) IP-пакета, ВИП — виртуальный идентификатор потока. Например, фильтр вида „192.168.1.2, 255.255.255.0, 192.168.2.3, 255.255.255.0, 25, 665, 3, 77“ из всего множества пакетов выделит только те, которые идут из подсети 192.168.1.0/24 в подсеть 192.168.2.0/24, при этом отправлены с порта 25 на порт 665 с ToS=3 и принадлежат виртуальному потоку № 77. В итоге возможно анализировать показатели качества обслуживания QoS только выбранного трафика. Смысл всех компонентов приведенного фильтра очевиден, кроме поля ВИП. Оно задает идентификатор потока, который присутствует в каждом пакете, но при этом не является частью ни одного из его физически передающихся полей. Использование этой метки позволяет отслеживать движение пакетов внутри туннелей, поскольку поле ВИП не подвергается изменению после любых операций туннелирования в отличие от всех остальных физически существующих полей пакета.

Заключение. Разработанная САПР компьютерной сети на основе компонентов с открытым исходным кодом позволяет решать задачи анализа и синтеза проектных решений при проектировании маршрутизируемых сетей, в которых обеспечивается функционирование QoS-сервисов. Предложенная модульная структурно-функциональная организация САПР позволяет обеспечить обратную совместимость модели с библиотеками Qt и NS-3, воспроизводимость результатов и высокое временно*е* разрешение процесса анимации, а также позволяет получать в качестве результатов характеристики любых классов трафика (включая туннелируемый трафик) на любых узлах моделируемой сети. В ходе создания САПР решен ряд задач моделирования протоколов TCP и OSPF в системе NS-3.

СПИСОК ЛИТЕРАТУРЫ

1. Инструменты для подготовки к сертификационным экзаменам Cisco [Электронный ресурс]: <http://www.cisco.com/en>.

2. Описание программы “NetCracker Network Management” [Электронный ресурс]: <http://netcracker.com/en/products/network_management>.

3. *Oliveira J., Vasseur J.P., Chen L., Scoglio C.* RFC4829. Label Switched Path (LSP) Preemption Policies for MPLS Traffic Engineering. San Diego: The IETF Trust, 2007 [Электронный ресурс]: <http://tools.ietf.org/rfc/rfc4829.txt>.

4. *Соснин В. В., Шинкарук Д. Н.* Моделирование маршрутизатора с поддержкой методов QoS в среде ns-3 // Сб. тр. молодых ученых и сотрудников кафедры ВТ. СПб: СПбГУ ИТМО, 2011. Вып. 2. С. 50—55.

5. *Postel J.* RFC791. Internet Protocol. California, 1981 [Электронный ресурс]: <http://tools.ietf.org/rfc/rfc0791.txt>.

6. *Moy J.* RFC2328. OSPF Version 2. Westford, 1998 [Электронный ресурс]: <http://tools.ietf.org/rfc/rfc2328.txt>.

Сведения об авторах

Тауфик Измайлович Алиев — д-р техн. наук, профессор; Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики, кафедра вычислительной техники; заведующий кафедрой; E-mail: aliev@d1.ifmo.ru

Владимир Валерьевич Соснин — Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики, кафедра вычислительной техники; ассистент; E-mail: vsosnin@mail.ru

Дмитрий Николаевич Шинкарук — аспирант; Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики, кафедра вычислительной техники; E-mail: dimashink@gmail.com

Михаил Юрьевич Тихонов — студент; Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики, кафедра вычислительной техники; E-mail: tihmihail@gmail.com

Никита Геннадьевич Бурмакин — аспирант; Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики, кафедра вычислительной техники; E-mail: nekit.mail@gmail.com

Рекомендована кафедрой вычислительной техники

Поступила в редакцию 08.02.12 г.

УДК 681.3

В. А. Богатырев, С. В. Богатырев, А. В. Богатырев

ФУНКЦИОНАЛЬНАЯ НАДЕЖНОСТЬ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ С ПЕРЕРАСПРЕДЕЛЕНИЕМ ЗАПРОСОВ

Предложен метод оценки функциональной надежности вычислительных систем по вероятности выполнения запроса в системе за время, не превышающее предельно допустимого значения. Предлагаемый подход к оценке позволяет учитывать возможность адаптации системы к отказам и изменениям потока запроса путем перераспределения запросов между вычислительными узлами через сеть.

Ключевые слова: *отказоустойчивость, распределенные вычислительные системы, перераспределение запросов, кластер, оптимизация, режим реального времени.*

Введение. Высокая функциональная надежность и отказоустойчивость распределенных вычислительных систем, объединяющих ресурсы нескольких кластеров, достигаются за счет перераспределения запросов между узлами кластеров [1—2]. Перераспределение запросов через сеть вносит дополнительную задержку, но увеличивает возможности адаптации системы к изменению потока запросов, к отказам и отключениям узлов, что определяет актуальность задачи анализа и оптимизации процесса распределения запросов, особенно для систем, работающих в режиме реального времени с жесткими ограничениями на время выполнения запросов.

Постановка задачи оптимизации распределения запросов. Рассмотрим распределенную компьютерную систему (рис. 1), содержащую локальные (отдельные) кластеры S_1, на каждый из которых поступает отдельный поток запросов, и доступный через сеть общий кластер S_3, обеспечивающий возможность адаптации системы к отказам и перегруженности серверов путем перераспределения запросов через сеть.

Для компьютерных систем, функционирующих в режиме реального времени, в качестве критерия оптимизации выберем максимум функциональной надежности, определяемой как вероятность выполнения заданий (функциональных запросов) за время, не превышающее предельно допустимого значения t_0. Указанный критерий имеет две составляющие:

— вероятность сохранения работоспособности системы при отказах;

— вероятность выполнения запроса в системе за время, не превышающее предельно допустимого значения, с учетом деградации вычислительных возможностей системы по мере накопления отказов.

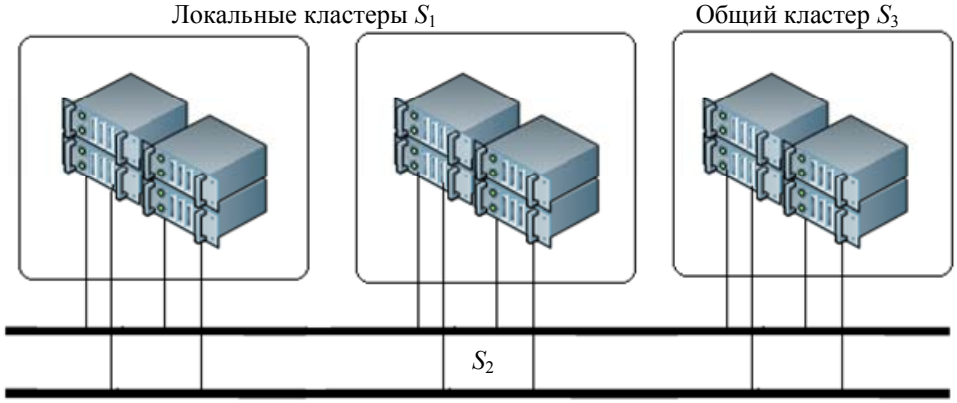


Рис. 1

Рассмотрим возможности увеличения функциональной надежности систем с разделением потока запросов к каждому локальному кластеру, достигаемого перераспределением

запросов (нагрузки) между кластерами с учетом отказов ресурсов кластеров и возможной неравномерности потока запросов к ним [2—5].

Для такой системы, предусматривающей перераспределение запросов через магистрали S_2 между кластерами, задача включает оптимизацию структуры системы (при проектировании) и процесса распределения запросов (при эксплуатации) в системе [6—10].

Задача оптимизации структуры является статической (выполняется до начала функционирования системы — изменение структуры функционирующей системы возможно только в результате отказов), а задача распределения запросов — динамической, решаемой в процессе функционирования системы на основе мониторинга текущей загруженности узлов системы и характеристик потока запросов.

При оптимизации структуры для заданных ограничений на стоимость реализации системы требуется найти число (кратность резервирования) магистралей R (узлов коммутации) и серверов в локальных кластерах N и в общем кластере M, обеспечивающих наибольшую вероятность выполнения запросов за время t_0.

Для локального кластера определяется доля потока запросов, перераспределяемых через сеть в общий кластер, при которой достигается максимум вероятности выполнения запросов за время t_0.

Оценка функциональной надежности. В исследуемой системе запросы, поступающие в локальный кластер $S_1(\alpha)$ с интенсивностью $\lambda=\alpha\Lambda$ (Λ — интенсивность входного потока запросов в системе), с вероятностью g обслуживаются внутри него и с вероятностью $(1–g)$ перераспределяются через сеть (резервированные магистрали) S_2 в общий кластер S_3. Число узлов в системах S_1, S_2, S_3 в исходном состоянии (без отказов) соответственно равно N, R, M.

Интенсивность поступления запросов в общий кластер S_3 через сеть S_2, равна $\Lambda_3=\Lambda((1–g)\alpha+1)$. Поскольку каждый запрос, направляемый в общий кластер, сопровождается возвратом результатов, интенсивность передачи через сеть S_2 запросов будем считать равной $\Lambda_2=2\Lambda((1–g)\alpha+1)$.

Представим каждый узел системы (серверы и магистрали) в виде одноканальной системы массового обслуживания типа $M/M/1$ [11].

Для локальных кластеров, не предусматривающих перераспределения запросов через сеть, вероятность своевременного выполнения запросов (непревышения значения t_0) [12], с учетом того что на каждый из N серверов кластера поступает часть $(1/N)$ потока запросов, определяется как

$$P = 1 - \left(\frac{\lambda v}{N}\right) e^{-\left(\frac{1}{v} - \frac{\lambda}{N}\right) t_0} = 1 - \left(\frac{\Lambda \alpha v}{N}\right) e^{-\left(\frac{1}{v} - \frac{\Lambda\alpha}{N}\right) t_0},$$

где v — среднее время выполнения запроса сервером.

При абсолютной надежности узлов кластера, если g-я доля запросов выполняется в соответствующем локальном кластере, а $(1–g)$-я их доля перераспределяется через сеть в общий кластер, то

$$P = g\left[1 - \left(\frac{g\Lambda\alpha v}{N}\right) e^{-\left(\frac{1}{v} - \frac{g\Lambda\alpha}{N}\right) t_0}\right] + (1-g)\left[1 - \left(\frac{\Lambda v[\alpha(1-g)+1]}{M}\right) e^{-\left(\frac{1}{v} - \frac{\Lambda[\alpha(1-g)+1]}{M}\right)(t_0 - T_2)}\right], \quad (1)$$

где T_2 — среднее время пребывания запросов в сети S_2:

$$T_2 = \frac{v_2}{1 - 2\Lambda v_2[\alpha(1-g)+1]/R},$$

здесь v_2 — среднее время передачи запросов и результатов их выполнения через сеть.

С учетом отказов вычислительной системы вероятность своевременного выполнения запросов определим как:

$$P = P_1P_2P_3\left\{g\left[1-\left(\frac{g\Lambda\alpha v}{n}\right)e^{-\left(\frac{1}{v}-\frac{g\Lambda\alpha}{n}\right)t_0}\right]+\right.$$

$$+(1-g)\left\{1-\left(\frac{\Lambda v[\alpha(1-g)+1]}{m}\right)e^{-\left(\frac{1}{v}-\frac{\Lambda[\alpha(1-g)+1]}{m}\right)\left(t_0-\frac{v_2}{1-\frac{2\Lambda v_2[\alpha(1-g)+1]}{r}}\right)}\right\}+$$

$$+P_1(1-P_2P_3)\left[1-\left(\frac{\Lambda\alpha v}{n}\right)e^{-\left(\frac{1}{v}-\frac{\Lambda\alpha}{n}\right)t_0}\right]+$$

$$+(1-P_1)P_2P_3\left[1-\left(\frac{\Lambda v[\alpha+1]}{m}\right)e^{-\left(\frac{1}{v}-\frac{\Lambda[\alpha+1]}{m}\right)\left(t_0-\frac{v_2}{1-\frac{2\Lambda v_2[\alpha+1]}{r}}\right)}\right], \qquad (2)$$

где $P_1P_2P_3$ — вероятность одновременной работоспособности рассматриваемого локального кластера, магистрали и общего кластера; $P_1(1-P_2P_3)$ — вероятность исправности рассматриваемого кластера при отказе остального оборудования системы; $(1-P_1)P_2P_3$ — вероятность отказа рассматриваемого кластера при сохранении функционирования системы за счет перераспределения запросов через сеть на общедоступный кластер,

$$P_1 = 1-(1-k_1)^N,\ P_2 = 1-(1-k_2)^R,\ P_3 = 1-(1-k_1)^M,$$

здесь k_1, k_2 — коэффициенты готовности сервера и магистрали;

$$k_1 = \mu_1/(\lambda_1+\mu_1),\ \ k_2 = \mu_2/(\lambda_2+\mu_2),$$

λ_1, λ_2, μ_1, μ_2 — интенсивность отказов и восстановлений серверов и магистралей.

Математические ожидания числа исправных серверов в рассматриваемом локальном кластере n, в общедоступном кластере m и числа r магистралей определим как:

$$n = \sum_{i=1}^{N} C_N^i k_1^i (1-k_1)^{N-i},\quad m = \sum_{i=1}^{M} C_M^i k_1^i (1-k_1)^{M-i},\quad r = \sum_{i=1}^{R} C_R^i k_2^i (1-k_2)^{R-i}.$$

Результаты расчетов. Расчеты значения вероятности P проведем при M=20, N=5, R=1, v=1, v_2 =0,01 с для средней интенсивности запросов Λ=10 1/с (рис. 2; *1* — t_0 =1,5, *2* — 2, *3* — 5, *4* — 10 с).

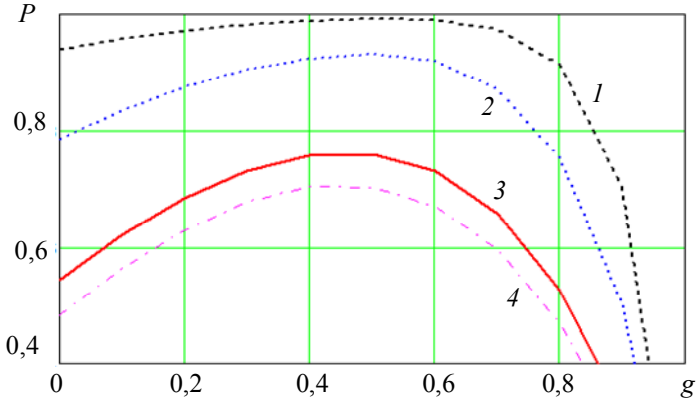


Рис. 2

Из представленных графиков видно, что существует оптимальное значение доли запросов, перераспределяемых через сеть для выполнения в общедоступном сервере, при котором

достигается максимальное значение вероятности (надежности) выполнения запросов за время, не превышающее предельно допустимого значения времени ожидания.

Для заданных параметров на основе целевой функции (1), (2) с использованием MathCad 15 определяется оптимальное значение доли запросов, перераспределяемых через сеть на выполнение в общедоступный кластер S_3. Так, при Λ=6 с$^{-1}$, t_0= 2 и 4 с оптимальному процессу перераспределения запросов соответствуют значения g=0,44 и 0,47. Результаты расчета вероятности P при t_0=2 и 4 представлены на рис 3, здесь кривые *1* и *2* соответствуют случаю перераспределения запросов через сеть при g=0,47; t_0=4 с и g=0,44; t_0=2 с; кривые *3* и *4* — системы без перераспределения запросов через сеть для t_0=4 и 2 с соответственно.

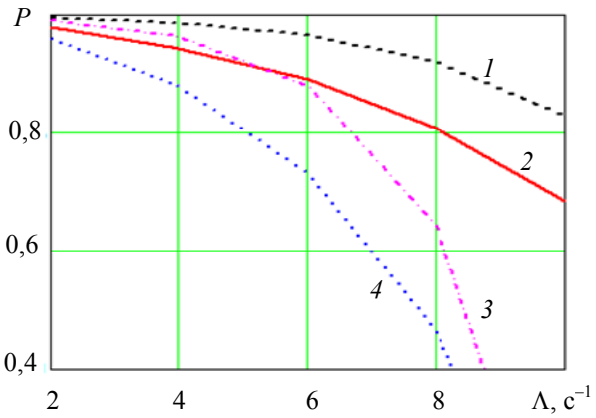


Рис. 2

Таким образом, продемонстрирована высокая эффективность перераспределения запросов между кластерами при оптимизации доли перераспределяемых через сеть запросов.

Заключение. В настоящей работе оценена функциональная надежность системы, предусматривающей перераспределение запросов через сеть между кластерами, определяемая как вероятность выполнения запроса в системе за время, не превышающее предельно допустимого значения.

Предлагаемый подход к оценке функциональной надежности учитывает возможности адаптации системы в результате перераспределения запросов через сеть к отказам серверов и изменениям потока запросов.

Показано, что перераспределение запросов между кластерами позволяет существенно повысить функциональную надежность распределенной системы, эффективность которой возрастает при оптимизации доли запросов, перераспределяемых через сеть.

СПИСОК ЛИТЕРАТУРЫ

1. *Таненбаум Э., Ван Стеен М.* Распределенные системы. Принципы и парадигмы. СПб: Питер, 2003. 877 с.

2. *Богатырев В. А.* К повышению надежности вычислительных систем на основе динамического распределения функций // Изв. вузов. Приборостроение. 1981. № 8. С. 62—65.

3. *Богатырев В. А.* Распределение заданий в многомашинных вычислительных системах // Изв. вузов. Приборостроение. 1986. № 5. С. 43—47.

4. *Богатырев В. А.* Интервально-сигнальный метод динамического распределения запросов с балансировкой нагрузки системы // Автоматика и вычислительная техника. 2000. № 6. С. 73—81.

5. *Богатырев В. А., Богатырев С. В.* К анализу и оптимизации серверных систем кластерной архитектуры с балансировкой нагрузки // Приборы и системы. Управление, контроль, диагностика. 2010. № 2. С. 4—9.

6. *Богатырев В. А., Богатырев С. В.* Объединение резервированных серверов в кластеры высоконадежной компьютерной системы // Информационные технологии. 2009. № 6. С. 41—47.

7. *Богатырев В. А., Богатырев С. В., Голубев И. Ю.* Оптимизация структуры и процесса перераспределения заданий между кластерами вычислительной системы // Автоматика и вычислительная техника. 2012. № 3. С. 73—81.

8. *Богатырев В. А.* Мультипроцессорные системы с динамическим перераспределением запросов через общую магистраль // Изв. вузов Приборостроение. 1985. № 3. С. 33—38.

9. *Богатырев В. А.* Надежность функционально-распределенных резервированных структур с иерархической конфигурацией узлов // Изв. вузов. Приборостроение. 2000. № 4. С. 67—70.

10. *Богатырев В. А.* Оптимальное резервирование системы разнородных серверов // Приборы и системы. Управление, контроль, диагностика. 2007. № 12. С. 30—36.

11. *Клейнрок Л.* Теория массового обслуживания. М.: Машиностроение, 1979.

12. *Кофман А., Крюон Р.* Массовое обслуживание. М.: Мир, 1965.

Сведения об авторах

Владимир Анатольевич Богатырев — д-р техн. наук, профессор; Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики, кафедра вычислительной техники; E-mail: Vladimir.bogatyrev@gmail.com

Станислав Владимирович Богатырев — „АЙТИ ХАУС“, Санкт-Петербург; главный инженер; E-mail: realloc@gmail.com

Анатолий Владимирович Богатырев — аспирант; Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики, кафедра вычислительной техники; E-mail: ganglion@gmail.com

Рекомендована кафедрой вычислительной техники

Поступила в редакцию 08.02.12 г.

УДК 681.2

Т. И. Алиев

ЗАДАЧИ СИНТЕЗА СИСТЕМ С ПОТЕРЯМИ

Формулируются задачи синтеза систем с потерями при наличии различных ограничений на характеристики их функционирования. В процессе синтеза определяются емкость накопителя и производительность устройства, обеспечивающие минимальную стоимость системы.

Ключевые слова: *система с потерями, накопитель ограниченной емкости, производительность, вероятность потери, время пребывания, стоимость системы.*

Введение. Проектирование дискретных систем со стохастическим характером функционирования, примерами которых могут служить информационно-управляющие системы, серверы локальных вычислительных сетей, сетевое оборудование (маршрутизаторы, коммутаторы) и т.п., осуществляется с использованием различных моделей и методов в зависимости от класса проектируемой системы, ее особенностей и требований, предъявляемых к качеству ее функционирования. При проектировании технических систем в качестве моделей целесообразно использовать системы и сети массового обслуживания с потерями, которые имеют накопители ограниченной емкости [1]. При некоторых предположениях могут быть получены аналитические зависимости для расчета характеристик дискретных систем со стохастическим характером функционирования, что позволяет эффективно решать задачи оптимального синтеза.

К основным характеристикам, определяющим эффективность функционирования систем с потерями, относятся: вероятность потери заявок π, среднее время пребывания заявок в системе u и стоимость системы S.

Требуемое качество функционирования системы, задаваемое в виде ограничений, налагаемых на характеристики, обеспечивается в процессе синтеза за счет выбора структурно-функциональной организации системы.

Описание системы. Положим, что исследуемая система содержит одно устройство, обрабатывающее заявки (запросы, команды, транзакции, детали и т.п.), поступающие в систему в случайные моменты времени из неограниченного источника заявок. Поступающие заявки образуют однородный поток и создают в системе однородную нагрузку. Средний интервал между заявками равен a. Средняя ресурсоемкость, измеряемая количеством работы, которое затрачивается на обработку одной заявки, равна θ. Скорость обработки одной заявки устройством (производительность, или быстродействие, устройства), измеряемая количеством работы, выполняемой устройством за единицу времени, равна V. В каждый момент времени устройство может обрабатывать только одну заявку. Заявки, поступившие в систему во время обработки устройством ранее поступившей заявки, располагаются в накопителе емкостью E, при этом обрабатываемая заявка также занимает одно место в накопителе. Заявка, поступившая в систему, теряется, если накопитель заполнен.

Расчет характеристик системы с потерями. Положим, что в систему поступает простейший поток заявок с интенсивностью $\lambda = 1/a$, длительность обработки которых распределена по экспоненциальному закону со средним значением b, причем нагрузка может принимать любые положительные значения:

$$y = \frac{\lambda\theta}{V} > 0. \tag{1}$$

В этом случае вероятность того, что в системе с накопителем емкостью E находится k заявок $k = 0, 1, 2, ..., E,$ рассчитывается в зависимости от значения нагрузки:

$$p_k = \begin{cases} \dfrac{y^k(1-y)}{1-y^{E+1}}, & y \neq 1, \\ \dfrac{1}{E+1}, & y = 1. \end{cases} \tag{2}$$

Тогда вероятность потери заявки из-за ограниченной емкости накопителя:

$$\pi = \begin{cases} \dfrac{y^E(1-y)}{1-y^{E+1}}, & y \neq 1, \\ \dfrac{1}{E+1}, & y = 1. \end{cases} \tag{3}$$

Соответственно загрузка системы составит

$$\rho = \lambda_0 b = \begin{cases} \left(1 - \dfrac{y^E(1-y)}{1-y^{E+1}}\right) y, & y \neq 1, \\ \dfrac{E}{E+1}, & y = 1, \end{cases} \tag{4}$$

где $\lambda_0 = (1-\pi)\lambda$ — интенсивность потока обслуженных заявок.

Среднее число заявок, находящихся в системе, и следовательно в накопителе, определяется через вероятности состояний (2):

$$m = \sum_{k=1}^{E} k\, p_k = \frac{1-y}{1-y^{E+1}} \sum_{k=1}^{E} k\, y^k ,$$

из этого соотношения после некоторых преобразований получим:

$$m = \begin{cases} \dfrac{y}{1-y^{E+1}} \left[\dfrac{1-y^E}{1-y} - E\, y^E \right], & y \neq 1, \\ \dfrac{E}{2}, & y = 1. \end{cases} \tag{5}$$

Тогда среднее время пребывания заявок в системе:

$$u = \frac{m}{\lambda_0} = \begin{cases} \dfrac{b}{(1-\pi)(1-y^{E+1})} \left[\dfrac{1-y^E}{1-y} - E\, y^E \right], & y \neq 1, \\ \dfrac{E}{2(1-\pi)\lambda}, & y = 1. \end{cases} \tag{6}$$

Анализ свойств системы. Одной из основных характеристик, определяющих эффективность функционирования систем с потерями, является вероятность потери заявок π вследствие переполнения накопителя. Очевидно, что значение этого параметра зависит от емкости накопителя E и производительности устройства V: $\pi = f(E, V)$, чем больше E и V, тем меньше π.

На рис. 1 показан характер зависимостей вероятности потери заявок от емкости накопителя (рис. 1, *а*) и производительности устройства (рис. 1, *б*) для системы с низкой (y_1<1) и высокой нагрузкой (y_2>1).

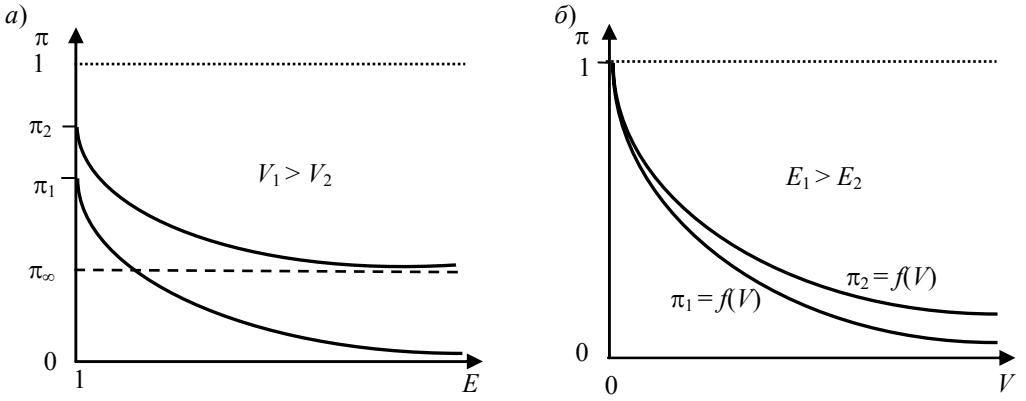


Рис. 1

Вероятность потери заявок в случае единичной емкости накопителя (E=1) определяется как $\pi_i = \dfrac{y_i}{1+y_i}$ $(i = 1, 2)$. Заметим, что для высоконагруженных систем ($y_2 > 1$) вероятность потери заявок с увеличением емкости накопителя $(E \to \infty)$ стремится к некоторому пределу π_∞. Этот предел обусловлен тем, что в случае $\lambda_2 > \mu_2$ даже при неограниченной емкости накопителя доля обработанных заявок за единицу времени, равная отношению интенсивности обработки $\mu_2 = 1/b_2$ к интенсивности поступления λ_2, составит $\pi_0 = \mu_2/\lambda_2 = 1/y_2$. Следовательно, доля необработанных заявок $\pi_\infty = 1 - y_2^{-1}$ может рассматриваться как доля потерянных заявок.

Уменьшение вероятности потери заявок в системе за счет увеличения производительности V устройства и емкости E накопителя приводит к увеличению стоимости системы, которая может быть рассчитана как

$$S = S_{\text{у}} + S_{\text{н}} = \alpha V^{\beta} + s_0 E, \qquad (7)$$

где $S_{\text{у}} = \alpha V^{\beta}$ — стоимость устройства (α и β — стоимостные коэффициенты пропорциональности и нелинейности соответственно); $S_{\text{н}} = s_0 E$ — стоимость накопителя (s_0 — стоимость единицы накопителя, предназначенной для одной заявки).

Постановка задачи синтеза. При отсутствии ограничений на характеристики функционирования системы в качестве обобщенного критерия эффективности может использоваться функция следующего вида:

$$F = \gamma_1 \pi + \gamma_2 u + \gamma_3 S, \qquad (8)$$

где $\gamma_1, \gamma_2, \gamma_3$ — весовые коэффициенты, определяющие степень ценности (важности) соответствующей характеристики для синтезируемой системы и удовлетворяющие условию $\gamma_1 + \gamma_2 + \gamma_3 = 1$, а π, u и S — определяются соответственно выражениями (3), (6) и (7).

В этом случае задача синтеза формулируется следующим образом: определить емкость накопителя и производительность устройства, при которых критерий эффективности (8) принимает минимальное значение.

На рис. 2 иллюстрируется задача определения оптимальных значений емкости накопителя и производительности устройства. Здесь представлены графики, показывающие зависимость вероятности потерь $\pi(E,V)$, среднего времени пребывания $u(E)$ и $u(V)$, стоимости системы $S = \alpha V$ (предполагается, что $\beta = 1$) и критерия эффективности (8) от вектора оптимизируемых параметров (E,V). Значения $(E,V)_{\text{opt}}$, соответствующие минимуму критерия эффективности F, являются оптимальными.

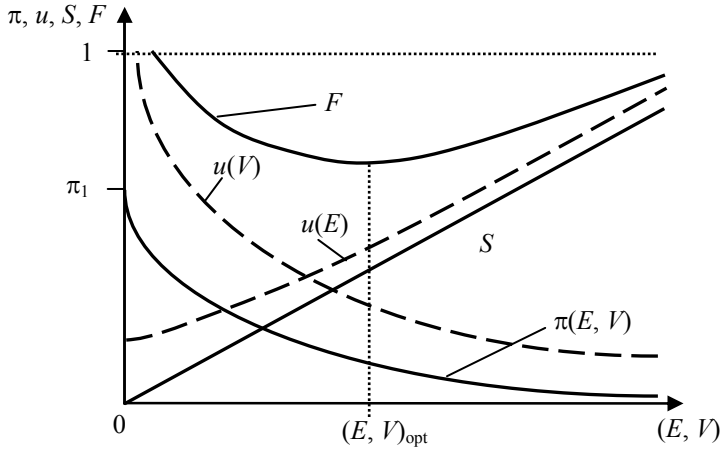


Рис. 2

Отметим, что зависимости среднего времени пребывания заявок в системе $u(E)$ и $u(V)$, представленные на рис. 2 пунктирными кривыми, ведут себя по-разному при увеличении соответственно емкости накопителя E и производительности устройства V: среднее время пребывания заявок в системе растет с увеличением емкости и снижается с увеличением производительности. При этом зависимость $u(E)$ может иметь разный характер — возрастающий, убывающий или любой другой.

При наличии ограничений на характеристики задача синтеза системы с одним устройством и накопителем ограниченной емкости может быть сформулирована в четырех постановках в зависимости от ограничений, а именно — определить производительность V устройства и емкость E накопителя, обеспечивающие выполнение ограничений (обозначено звездочкой):

1) на вероятность потери заявок $\pi \le \pi^*$ при минимальной стоимости системы S;

2) на вероятность потери заявок $\pi \le \pi^*$ и среднее время пребывания заявок в системе $u \le u^*$ при минимальной стоимости системы S;

3) на вероятность потери $\pi \le \pi^*$ и стоимость системы $S < S^*$ при минимальном значении среднего времени пребывания заявок в системе u;

4) на среднее время пребывания заявок в системе $u \le u^*$ и на стоимость системы $S \le S^*$ при минимальном значении вероятности потери заявок π.

Алгоритм синтеза. Рассмотрим задачу синтеза применительно к первой и второй постановкам.

Выражение (2) позволяет определить емкость накопителя для заданного ограничения π^* при известной нагрузке y, т.е. при известной производительности V устройства:

$$E \ge \begin{cases} \log_y \dfrac{\pi^*}{1 - y(1-\pi^*)}, & y \ne 1, \\ \dfrac{1-\pi^*}{\pi^*}, & y = 1. \end{cases}$$

В общей постановке задачи синтеза производительность устройства неизвестна и подлежит определению. Поскольку получить решение задачи оптимального синтеза в явном аналитическом виде с использованием зависимостей (1)—(6) не представляется возможным, один из подходов к решению задачи ввиду небольшого числа оптимизируемых параметров состоит в последовательном переборе значений емкости накопителя E. При этом для каждого значения E с использованием выражения (3) может быть определена минимальная производительность устройства V, при которой выполняется заданное ограничение на вероятность потери заявок:

$$\frac{y^E(1-y)}{1-y^{E+1}} \le \pi^*.$$

С учетом (1) преобразуем последнее неравенство к виду:

$$V^{E+1} - \frac{(\lambda\theta)^E}{\pi^*}V + (\lambda\theta)^{E+1}\left(\frac{1}{\pi^*} - 1\right) \ge 0.$$

Решив полученное неравенство относительно V при заданном значении емкости E, например, с использованием одного из численных методов [2], можно определить минимальное значение производительности устройства V, при котором обеспечивается ограничение π^*.

В частности, при $E = 1$ последнее выражение примет вид:

$$V^2 - \frac{\lambda\theta}{\pi^*}V + (\lambda\theta)^2\left(\frac{1}{\pi^*} - 1\right) \ge 0,$$

откуда получим:

$$V \ge \frac{1-\pi^*}{\pi^*}\lambda\theta.$$

Рассмотрим пример со следующими исходными данными: $\lambda = 1\,\text{с}^{-1}$; $\theta = 400$ единиц (ед.); $\alpha = 100$; $\beta = 0{,}5$; $s_0 = 50$; $\pi^* = 0{,}05$.

В таблице представлены результаты расчетов для разных значений емкости накопителя, из которой следует, что минимальная стоимость системы $S = 2534$ у.ед. достигается при $E = 7$ и $V = 477$ ед./с.

E	4	5	6	7	8	9	10	11	12
m	1,5	1,9	2,8	2,9	3,4	3,9	4,5	5,1	5,6
u, с	1,5	2,0	2,5	3,0	3,6	4,1	4,7	5,3	5,9
V, $с^{-1}$	594	537	501	477	459	446	436	428	422
S, у.ед.	2637	2567	2538	2534	2542	2562	2588	2619	2654

Во второй постановке задачи синтеза дополнительное ограничение налагается на среднее время пребывания заявок: $u \le u^*$. Если в результате синтеза на предыдущем этапе это ограничение не выполняется, необходимо увеличить производительность устройства, что приведет к повышению стоимости системы. При этом вероятность потери заявок уменьшится и, следовательно, можно попытаться снизить стоимость системы за счет уменьшения емкости накопителя.

Положим, что в нашем примере дополнительно задано ограничение $u^* = 2,5$ с. Для того чтобы при $E = 7$ выполнялось это условие, необходимо увеличить производительность устройства до $V = 518$ ед./с, что приведет к увеличению стоимости системы до $S = 2626$ у.ед. В результате предпочтительным оказывается вариант построения системы с накопителем емкостью $E = 6$ (см. таблицу) и стоимостью $S = 2538$ у.ед.

Синтез систем с двумя устройствами. Предложенный подход может быть распространен на системы с двумя последовательно расположенными устройствами, в которых заявки после обработки в первом устройстве направляются ко второму устройству. Производительность 1-го и 2-го устройства и емкость накопителей обозначим как V_1, V_2 и E_1, E_2. Тогда стоимость системы и вероятность потери заявок в системе:

$$S = \alpha_1 V_1^{\beta_1} + \alpha_2 V_2^{\beta_2} + s_0(E_1 + E_2) \text{ и } \pi = \pi_1 + \pi_2(1 - \pi_1).$$

С учетом ограничения на вероятность потери в системе π^* и последнего выражения получим: $\pi_2 \le (\pi^* - \pi_1)/(1 - \pi_1)$, где правая часть неравенства представляет собой ограничение на вероятность потери во втором устройстве при известном значении вероятности потери в первом устройстве. Таким образом, если известна вероятность потери заявок в первом устройстве, то задача синтеза для второго устройства при некоторых условиях может быть сведена к задаче синтеза системы с одним устройством.

Положим, что в первое устройство поступает простейший поток заявок с интенсивностью λ, а длительности обработки в устройствах распределены по экспоненциальному закону со средним значением $b_i = \theta_i / V_i \quad (i = 1; 2)$. При этих предположениях задача синтеза для первого устройства решается с использованием выражений (1)—(6) для $\pi_1^* = (0,25—0,75)\pi^*$. В качестве результирующего принимается значение π_1^*, при котором обеспечивается минимальная стоимость первого устройства. Тогда ограничение на вероятность потери для второго устройства: $\pi_2^* = (\pi^* - \pi_1^*)/(1 - \pi_1^*)$. Для того чтобы можно было воспользоваться выражениями (1)—(6) для второго устройства и решения задачи синтеза, необходимо выполнить анализ характера потока заявок, покидающих первое устройство и образующих входной поток заявок во второе устройство.

Многочисленные имитационные эксперименты показали, что коэффициент вариации ν выходящего потока из первого устройства зависит от нагрузки y_1 и емкости E_1 накопителя и

принимает значения в интервале ν = 0,7—1 при $E_1 = 1$ и ν = 0,95—1 при $E_1 = 4$ (рис. 3), причем наименьшие значения коэффициента вариации достигаются при нагрузке $y = 1$. Другими словами, чем больше E_1, тем ближе к единице коэффициент вариации выходящего потока, что при $E_1 \geq 4$ позволяет использовать выражения (1)—(6) с высокой степенью точности для расчета характеристик функционирования второго устройства.

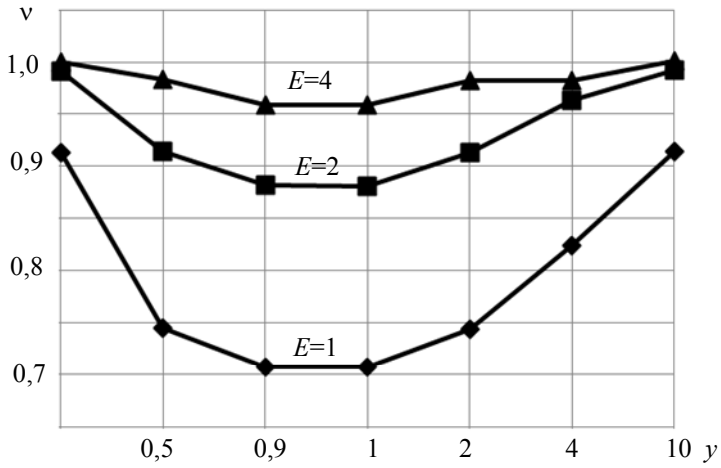


Рис. 3

Кроме того, предположение о простейшем потоке заявок ко второму устройству позволяет получить верхнюю оценку вероятности потери во втором устройстве.

Заключение. Рассмотренный подход к решению задач синтеза систем с потерями и полученные результаты позволяют определить не только емкость накопителей, но и оценить требования к производительности устройств, входящих в состав системы, с учетом совокупных затрат на построение системы для широкого диапазона параметров нагрузки.

СПИСОК ЛИТЕРАТУРЫ

1. *Алиев Т. И.* Основы моделирования дискретных систем. СПб: СПбГУ ИТМО, 2009. 363 с.

2. *Формалев В. Ф., Ревизников Д. Л.* Численные методы. М.: Физматлит, 2006. 398 с.

Сведения об авторе

Тауфик Измайлович Алиев — д-р техн. наук, профессор; Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики, кафедра вычислительной техники; заведующий кафедрой;
E-mail: aliev@d1.ifmo.ru

Рекомендована кафедрой
вычислительной техники

Поступила в редакцию
08.02.12 г.

УДК 004.89: 002.53

Л. А. Муравьева-Витковская

ОБЕСПЕЧЕНИЕ КАЧЕСТВА ОБСЛУЖИВАНИЯ В МУЛЬТИСЕРВИСНЫХ КОМПЬЮТЕРНЫХ СЕТЯХ ЗА СЧЕТ ПРИОРИТЕТНОГО УПРАВЛЕНИЯ

Рассматриваются модели и методы оценки характеристик функционирования мультисервисных компьютерных сетей. Показана возможность обеспечения качества обслуживания за счет применения приоритетного управления неоднородным трафиком.

Ключевые слова: *мультисервисные компьютерные сети, приоритетное управление, качество обслуживания, неоднородный трафик, модель функционирования.*

Введение. Мультисервисные компьютерные сети (КС) — специфические системы, предоставляющие услуги как компьютерных, так и телекоммуникационных сетей. Специфичность мультисервисных КС обусловлена следующими факторами: многообразием сетевых технологий и архитектур; разнообразием требований к качеству передачи данных различного типа (например, для текстовых файлов наиболее важным показателем является надежность доставки, т.е. отсутствие потерь и искажений данных в файлах, а для аудио- и видеоданных — вариация (джиттер) задержки пакетов данных относительно требуемого момента поступления); неоднородностью трафика, для управления которым могут использоваться различные механизмы (методы доступа в LAN, алгоритмы маршрутизации, способы установления соединений и т.п.) [1—7], позволяющие предотвращать перегрузки и блокировки и обеспечивающие требуемое качество обслуживания (QoS) данных каждого типа. В рекомендациях МСЭ-Т Y.1541 [1] определены 5 классов QoS и сформулированы требования к характеристикам обслуживания пакетов каждого класса, выполнение которых гарантирует качественную передачу трафика. В качестве основных характеристик рассматриваются среднее время задержки пакета, вариация задержки и вероятность потери пакетов.

Системный подход к исследованию мультисервисных КС. Современные мультисервисные КС характеризуются разнообразием предоставляемых услуг, увеличением числа пользователей и объема передаваемых данных; повышением уровня требований к качеству обслуживания пользователей. Выполнение требований, предъявляемых к мультисервисным КС, возможно за счет структурно-функциональной организации сети, включая выбор конкретной технологии передачи и обработки данных, определение рациональной топологии коммуникационной сети, выбор сетевого оборудования, механизмов управления трафиком и т.д. Одним из предпочтительных способов распределения сетевых ресурсов является распределение в соответствии с существующими на данный момент приоритетами.

В качестве основных показателей эффективности мультисервисных КС используются производительность, оперативность и надежность. Для оценки эффективности мультисервисных КС могут использоваться модели массового обслуживания, отображающие временные задержки при обработке и передаче данных.

В основе исследований мультисервисных КС, как и других сложных технических систем, лежит системный подход [5], в рамках которого выполняется системотехническое проектирование, направленное на построение системы с заданным качеством. Для системотехнического проектирования необходимо располагать знаниями о том, как влияют различные способы структурной и функциональной организации на характеристики функционирования мультисервисных КС.

Отметим две характерные особенности системного подхода. Первая состоит в том, что все элементы и составляющие процесса проектирования рассматриваются в их взаимосвязи, взаимообусловленности, взаимозависимости и взаимном влиянии для оптимального достижения целей создания мультисервисной КС.

Вторая особенность системного подхода состоит в том, что он, являясь методологической основой, предполагает обязательную предпосылку о необходимости анализа процессов проектирования в их взаимосвязи на базе широкого применения современных количественных методов исследования, что позволяет при проектировании вырабатывать и принимать решения в условиях неопределенности и неполноты информации.

Модели и методы исследования мультисервисных КС. Одной из важных задач системотехнического проектирования мультисервисных КС является разработка методов исследования характеристик КС применительно к процессам приоритетного управления неоднородным трафиком на этапах разработки и эксплуатации.

Указанная цель достигается решением следующих задач исследования.

1. Анализ принципов организации мультисервисных КС различных классов с учетом существенной неоднородности трафика, наличия приоритетов, многообразия механизмов управления трафиком и способов распределения функций между устройствами системы, наличия непроизводительных затрат на обеспечение качества обслуживания (QoS), многообразия структурных способов построения систем.

2. Разработка на основе принципа иерархического многоуровневого моделирования структурно-функциональных моделей мультисервисных КС с приоритетным управлением неоднородным трафиком.

3. Разработка моделей и методов исследования процессов управления неоднородным трафиком в статическом и динамическом режимах с учетом сбоев в работе устройств мультисервисных КС.

Методы исследования базируются на аппарате теории вероятностей, теории массового обслуживания, теории случайных процессов, методах численного анализа и имитационного моделирования.

При исследовании способов управления неоднородным трафиком в мультисервисных КС одной из важных задач является определение законов распределения времени доставки пакетов и времени ожидания освобождения канала связи и их числовых характеристик (первых и вторых начальных моментов, коэффициентов вариации и т.п.) при заданном механизме управления трафиком.

Для решения указанной задачи воспользуемся базовой моделью массового обслуживания с неограниченной очередью, в которую с интенсивностью $\lambda_1,...,\lambda_H$ поступают H потоков пакетов, интервалы между которыми распределены по экспоненциальному закону. Длительность обслуживания τ_{b_h} пакетов класса h (h = 1, …, H) распределена по произвольному закону с плотностью распределения $b_h(\tau)$ и средним значением b_h^{-1}. Выбор пакета из очереди на обслуживание осуществляется в соответствии со смешанными приоритетами, которые задаются в виде матрицы приоритетов.

Для описанной модели методом введения дополнительного события получены соответствующие начальные моменты времени пребывания пакета класса h в модели, в частности, два первых начальных момента:

$$u_h^{(1)} = \frac{\sum_{i=1}^{H} r_6(i,h)\lambda_i b_i^{(2)}}{2(1-R_h^{(4)})(1-R_h^{(5)})} + \frac{b_h^{(1)}}{1-R_h^{(3)}};$$

$$u_h^{(2)} = \frac{\sum_{i=1}^{H} r_6(i,h)\lambda_i b_i^{(3)}}{3(1-R_h^{(4)})^2(1-R_h^{(5)})} + \frac{b_h^{(2)}}{(1-R_h^{(3)})^2} + \frac{\sum_{i=1}^{H} r_5(i,h)\lambda_i b_i^{(2)} \sum_{i=1}^{H} r_6(i,h)\lambda_i b_i^{(2)}}{2(1-R_h^{(4)})^2(1-R_h^{(5)})^2} +$$

$$+ \frac{\sum_{i=1}^{H} r_4(i,h)\lambda_i b_i^{(2)} \sum_{i=1}^{H} r_6(i,h)\lambda_i b_i^{(2)}}{2(1-R_h^{(4)})^3(1-R_h^{(5)})} + \frac{b_h^{(1)} \sum_{i=1}^{H} r_6(i,h)\lambda_i b_i^{(2)}}{(1-R_h^{(3)})(1-R_h^{(4)})(1-R_h^{(5)})} + \frac{b_h^{(1)} \sum_{i=1}^{H} r_3(i,h)\lambda_i b_i^{(2)}}{(1-R_h^{(3)})^3},$$

где $r_g(i,h)$, g =1, …, 6 — коэффициенты приоритетности, принимающие значения 0 или 1 в зависимости от значений элементов q_{ih} и q_{hi} матрицы приоритетов (q_{ih} = 0, если класс i не имеет приоритета к классу h; q_{ih} = 1, если класс i имеет относительный приоритет к классу h; q_{ih} = 2, если класс i имеет абсолютный приоритет к классу h) и позволяющие выделить классы пакетов i и h, имеющих между собой определенный вид приоритета: $r_1(i,h) = 0{,}5(1 - q_{ih} - q_{hi})(2 - q_{ih} - q_{hi})$; $r_2(i,h) = q_{ih}(2 - q_{ih})$; $r_3(i,h) = 0{,}5\ q_{ih}(q_{ih} - 1)$; $r_4(i,h) = r_2(i,h) + r_3(i,h)$; $r_5(i,h) = r_1(i,h) + r_4(i,h)$; $r_6(i,h) = r_2(h,i) + r_5(i,h)$;

$$\Lambda_h^{(g)} = \sum_{i=1}^{h} r_g(i,h)\lambda_i;\ R_h^{(g)} = \sum_{i=1}^{h} r_g(i,h)\lambda_i b_i^{(1)};\ b_i^{(l)} = \int_0^{\infty} \tau^l b_i(\tau) d\tau \quad (l = 1,2,...).$$

На основании приведенных формул можно определить характеристики функционирования мультисервисной КС как единой системы, так и ее основных элементов или подсистем.

Предложена локальная модель маршрутизатора, позволяющая определить время задержки пакетов разных классов в маршрутизаторе с приоритетным управлением, основанном на дисциплине обслуживания со смешанными приоритетами (ДО СП), динамически изменяющимися по нелинейной функции. Модель позволяет учитывать ненадежность работы канала связи.

Получено выражение для первого начального момента времени ожидания пакета класса h в модели:

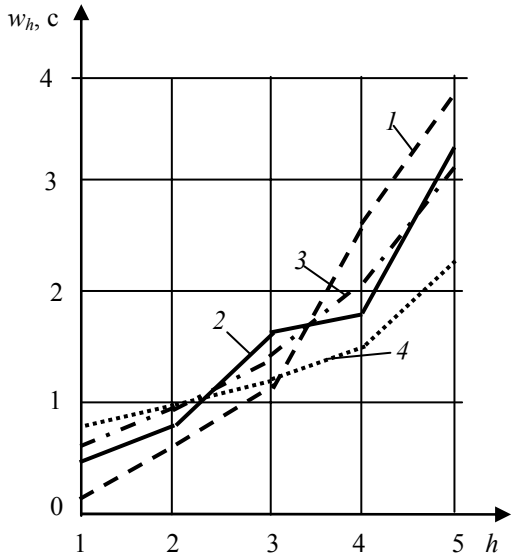
$$w_h = [1 - \lambda_0 b_0 - \sum_{i=1}^{H} r_4(i,h)\rho_i(1-\gamma_{hi})]^{-1} \{(1-R)^{-1} \sum_{i=1}^{H} \rho_i b_i + [\lambda_0 b_0 +$$

$$+ \sum_{i=1}^{H} r_3(i,h)\rho_i(1-\gamma_{hi})] b_h - \sum_{i=1}^{H} r_4(h,i)\rho_i w_i(1-\gamma_{ih}) - \sum_{i=1}^{H} r_3(h,i)\rho_i b_i(1-\gamma_{ih})\},$$

где $\rho_i = \lambda_i b_i$ — коэффициент загрузки, создаваемой пакетами класса i; $R = \sum_{i=1}^{H} \rho_i$ — суммарный коэффициент загрузки системы, причем $R < 1$; $\gamma_{ih} = \beta_i / \beta_h$ — отношение коэффициентов пропорциональности; λ_0 — интенсивность, с которой моменты выхода канала связи из строя образуют пуассоновский поток; b_0 — среднее значение времени восстановления канала связи, распределенного по экспоненциальному закону.

На рисунке приведена зависимость среднего времени ожидания пакетом освобождения канала связи от статических и динамических смешанных приоритетов (дисциплины обслуживания: *1* — с абсолютными, *2* — с относительными, *3* — со статическими смешанными, *4* — с динамическими смешанными приоритетами).

Анализ полученных результатов показывает, что введение динамических приоритетов позволяет уменьшить разброс между средними значениями времени задержки пакетов разных классов по сравнению со статическими.



Преимущество дисциплины обслуживания с динамическими приоритетами заключается в том, что при переходе от одной ДО СП к другой может быть обеспечено плавное (непрерывное) изменение характеристик пакетов разных классов путем соответствующего выбора значений коэффициентов пропорциональности β_h, в то время как в дисциплинах обслуживания со статическими приоритетами это изменение происходит скачкообразно.

Заключение. Предложенные аналитические методы расчета базовых и локальных моделей могут служить основой для построения более сложных глобальных моделей мультисервисных КС. С помощью глобальных моделей могут решаться задачи определения структурных и функциональных параметров мультисервисных КС, обеспечивающих заданное качество обслуживания данных разных типов. Аналитические методы расчета позволяют лишь оценить значения параметров, при этом результаты могут иметь значительную погрешность. Для повышения достоверности результатов следует применять комбинированный метод моделирования, основанный на сочетании аналитических и имитационных методов [2, 6, 7].

СПИСОК ЛИТЕРАТУРЫ

1. ITU-T Recommendation Y.1541. Network Performance Objectives for IP-Based Services. Int'l Telecommuncation Union, 2006. 50 p.

2. *Муравьева-Витковская Л. А.* Оценка характеристик приоритетной модели звена передачи данных мультисервисной компьютерной сети // Сб. докл. „Имитационное моделирование. Теория и практика. ИММОД—2011“. СПб: ОАО „Центр технологии судостроения и судоремонта“, 2011. Т. I. С. 207—213.

3. *Алиев Т. И.* Характеристики дисциплин обслуживания заявок с несколькими классами приоритетов // Изв. АН СССР. Техническая кибернетика. 1987. № 6. С. 188—191.

4. *Алиев Т. И., Муравьева Л. А.* Система с динамически изменяющимися смешанными приоритетами и ненадежным прибором // Автоматика и телемеханика. 1988. № 7. С. 99—106.

5. *Алиев Т. И.* Стохастические модели информационно-вычислительных систем // Сб. науч. статей. „Современные технологии“. СПб: СПбГУ ИТМО, 2003. С. 6—17.

6. *Муравьева-Витковская Л. А.* Определение структурно-функциональных параметров коммутатора телекоммуникационной сети при приоритетной обработке кадров // Там же. С. 27—33.

7. *Алиев Т. И., Никульский И. Е., Пяттаев В. О.* Моделирование ядра мультисервисной сети с относительной приоритезацией неоднородного трафика // Науч.-техн. вестн. СПбГУ ИТМО. 2009. Вып. 4(62). С. 88—96.

Сведения об авторе

Людмила Александровна Муравьева-Витковская — канд. техн. наук, доцент; Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики, кафедра вычислительной техники; E-mail: mur-lada@yandex.ru

Рекомендована кафедрой вычислительной техники

Поступила в редакцию 08.02.12 г.

ПРОГРАММНО-АППАРАТНЫЕ СРЕДСТВА ИНФОРМАЦИОННО-УПРАВЛЯЮЩИХ СИСТЕМ

УДК 621.3.085.42

А. А. Ожиганов

КОМПОЗИЦИОННЫЕ КОДОВЫЕ ШКАЛЫ ДЛЯ ПРЕОБРАЗОВАТЕЛЕЙ ЛИНЕЙНЫХ ПЕРЕМЕЩЕНИЙ

Рассматривается метод построения линейных композиционных кодовых шкал с одной информационной кодовой дорожкой. Приведен пример построения шкалы на основе предложенного метода.

Ключевые слова: *композиционная последовательность, кодовая шкала, линейная композиционная кодовая шкала, считывающие элементы.*

Введение. Цифровые преобразователи линейного перемещения (ЦПЛП) используются для обеспечения информационной связи по положению между позиционируемым объектом и устройством числового программного управления или устройством цифровой индикации. Наиболее перспективны ЦПЛП с непосредственным преобразованием перемещения в код на основе пространственного кодирования, основным элементом которых является кодовая шкала. В настоящее время в ЦПЛП широко применяются шкалы, кодовая маска которых выполнена в обыкновенном двоичном коде или коде Грея. Такие кодовые шкалы сложны в изготовлении, так как число их информационных дорожек обычно равно разрядности преобразователей. Поэтому масса и габариты преобразователей с увеличением их разрядности также возрастают [1].

В работах [2—4] рассмотрены псевдослучайные кодовые шкалы, используемые в качестве кодированного элемента ЦПЛП и имеющие всего одну или несколько (2—4) информационных кодовых дорожек (КД). Рассмотрим теоретические основы и метод построения однодорожечных линейных композиционных кодовых шкал (ЛККШ), обладающих всеми достоинствами псевдослучайных, но с более широким спектром разрешающей способности.

Теоретические основы построения линейных композиционных кодовых шкал. В цифровых преобразователях угла (ЦПУ), построенных по методу считывания, в качестве кодированного элемента используются [5] однодорожечные композиционные кодовые шкалы (ККШ). Особенностью таких шкал является то, что кодовая маска дорожки выполняется в соответствии с символами композиционной двоичной последовательности p-го порядка (K_p-последовательности).

Для получения K_p-последовательности $\{A_i\}$ используется полином

$$H(x) = \prod_{k=1}^{p} h_k(x)$$

степени $N = \sum_{k=1}^{p} m_k$ с коэффициентами поля Галуа $GF(2)$, где

$$h_k(x) = \sum_{j=0}^{m_k} h_j(x^j)$$

— примитивный полином степени m_k с параметрами $h_0 = h_{m_k} = 1$, h_j={0,1} при $0 < j < m_k$ [6].

Символы K_p-последовательности A_{N+i} генерируются в соответствии с рекуррентным выражением:

$$A_{N+i} = \bigoplus_{j=0}^{N-1} A_{i+j} H_j, \quad i = 0, 1, 2, ..., R - N - 1, \tag{1}$$

где знак $\oplus$ означает суммирование по модулю два. Начальные значения символов K_p-последовательности $A_0A_1...A_{N-1}$ выбираются с учетом того, что наибольший общий делитель (НОД) $[t_i(x),H(x)]=1$,

$$t_i(x) = \sum_{j=0}^{N-1} A_{i+j} x^j, i = 0,1,2,...,R-1.$$

Период R K_p-последовательности зависит от степеней полиномов $h_k(x)$ и от полинома начальных значений символов K_p-последовательности $t_i(x)$. Если все m_k (справедливо для $m_k <$ 34 [7]) представляют собой взаимно простые числа, а НОД $[t_i(x),H(x)]=1$, то

$$R = \prod_{k=1}^{p} L_k, \tag{2}$$

где $L_k = 2^{m_k} - 1$.

При построении ККШ символы K_p-последовательности отображаются на КД по ходу часовой стрелки в последовательности $A_0A_1...A_{R-1}$, что позволяет получить разрешающую способность ЦПУ на основе таких шкал $\delta = 360^o/R$.

K_p-последовательности относятся к классу циклических кодов, следовательно, можно через циклические сдвиги последовательности задать порядок размещения на шкале N считывающих элементов (СЭ). Иными словами, n-му СЭ (n = 1, 2, ..., N) ставится в соответствие I_n-й циклический сдвиг K_p-последовательности. Тогда полином, определяющий порядок размещения N СЭ на ККШ, имеет вид

$$r(x) = \sum_{n=1}^{N} x^{I_n}, \tag{3}$$

где $I_n \in \{0, 1, ..., R-1\}$. При I_1=0, согласно (3), второй, третий,..., N-й СЭ будут смещены (по ходу часовой стрелки) относительно первого СЭ на I_2, I_3,...,I_N квантов шкалы соответственно.

Для заданной разрешающей способности необходимо получить с N СЭ при полном обороте шкалы R различных N-разрядных кодовых комбинаций. Это обеспечивается путем решения задачи размещения на ККШ СЭ, которая сводится к нахождению подходящего линейно независимого множества из N циклических сдвигов K_p-последовательности [8].

Рассмотрим построение N-разрядной линейной ККШ с разрешающей способностью $\delta_л = D/R$, где D — контролируемое перемещение.

Для синтеза ЛККШ получим последовательность $\{A_i^*\}$, воспользовавшись рекуррентным соотношением (1) и предположив, что размещение СЭ на ЛККШ корректно и задается полиномом (3).

Очевидно, что последовательность $\{A_i^*\}$ включает в себя последовательность $\{A_i\}$, а также некоторые дополнительные символы, число которых зависит от особенностей размещения на ЛККШ СЭ.

Определим разность номеров циклических сдвигов K_p-последовательности, соответствующих случаю размещения на шкале двух соседних СЭ, как $d_i = j_m - j_{m-1}$ (i=1, 2, ..., N–1, m=2, 3, ..., N).

Тогда для получения последовательности $\{A_i^*\}$ рекуррентное соотношение (1), при заданных начальных значениях символов, необходимо применить q раз:

$$q = R - N + \sum_{i=1}^{N-1} d_i \ . \tag{4}$$

С учетом того, что

$$\sum_{i=1}^{N-1} d_i = d_1 + \cdots + d_i + \cdots + d_{N-1} = (j_2 - j_1) + \cdots + (j_m - j_{m-1}) + \cdots + (j_N - j_{N-1}) = j_N \ \ (j_1 = 0),$$

соотношение (4) в итоге принимает вид

$$q = R - N + j_N . \tag{5}$$

Общее число символов последовательности $\{A_i^*\}$ с учетом N задаваемых начальных значений может быть найдено из соотношения

$$Q = R + j_N . \tag{6}$$

Метод построения линейных композиционных кодовых шкал для преобразователей линейных перемещений включает следующие этапы.

1. В зависимости от требуемой разрядности N и разрешающей способности $\delta_л$ шкалы с использованием примитивных полиномов $h(x)$ формируется полином $H(x)$.

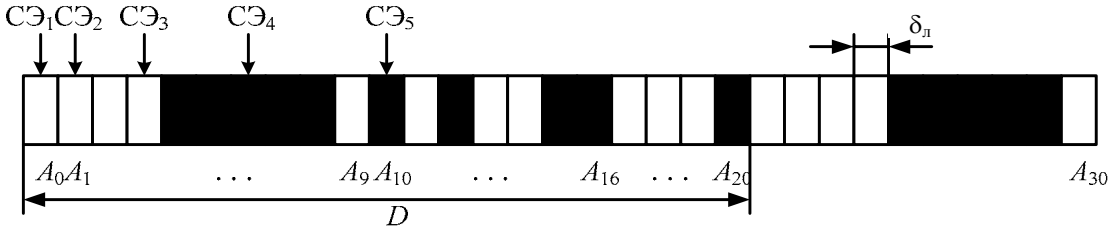
2. С учетом требований к размещению на шкале СЭ формируется полином размещения $r(x)$.

3. С использованием рекуррентного соотношения (1), с учетом (5) и (6), получается последовательность $\{A_i^*\}$, i=0, 1, …, Q–1.

4. Элементарные участки (кванты) шкалы выполняются в соответствии с двоичной последовательностью $\{A_i^*\}$. Символы последовательности $\{A_i^*\}$ отображаются на информационной кодовой дорожке слева направо: $A_0\, A_1\, \ldots\, A_{Q-1}$.

5. Считывающие элементы размещаются вдоль КД линейной ККШ в соответствии с полиномом $r(x)$.

Пример линейной композиционной кодовой шкалы. На рисунке приведена пятиразрядная ЛККШ, СЭ размещены в соответствии с полиномом $r(x) = 1 + x + x^3 + x^6 + x^{10}$.



Информационная дорожка шкалы длиной Q=31 выполнена в соответствии с последовательностью $\{A_i^*\}$=$A_0A_1...A_{30}$=00001111101010011000010000111110. При построении использован полином $H(x) = h_1(x)h_2(x) = (x^2 + x + 1)(x^3 + x + 1) = x^5 + x^4 + 1$, а символы A_{5+i} последовательности $\{A_i^*\}$ при начальных значениях A_0=A_1=A_2=A_3=0, A_4=1 удовлетворяют рекуррентному выражению $A_{5+i} = A_{4+i} \oplus A_i$ (i = 0, 1, ..., 25).

При перемещении шкалы на один элементарный участок, например справа налево, на выходах считывающих элементов СЭ$_1$, СЭ$_2$, СЭ$_3$, СЭ$_4$ и СЭ$_5$ формируются пятиразрядные кодовые комбинации, соответствующие двадцати одному варианту перемещений ЛККШ (см. таблицу).

Последовательность кодовых комбинаций ЛККШ

№ положения ЛККШ	СЭ$_1$	СЭ$_2$	СЭ$_3$	СЭ$_4$	СЭ$_5$	Десятичный эквивалент кода
0	0	0	0	1	1	3
1	0	0	1	1	0	6
2	0	0	1	1	1	7
3	0	1	1	0	0	12
4	1	1	1	1	0	30
5	1	1	1	0	1	29
6	1	1	0	1	1	27
7	1	1	1	0	0	28
8	1	0	0	0	0	16
9	0	1	1	1	0	14
10	1	0	0	1	1	19
11	0	1	0	0	0	8
12	1	0	1	0	0	20
13	0	0	1	0	0	4
14	0	1	0	1	0	10
15	1	1	0	0	1	25
16	1	0	0	0	1	17
17	0	0	1	0	1	5
18	0	0	0	0	1	1
19	0	1	0	1	1	11
20	1	0	0	1	0	18

Заключение. Рассмотренные однодорожечные ЛККШ могут использоваться в качестве кодированного элемента в преобразователях линейного перемещения, построенных по методу считывания. При одинаковой разрядности разрешающая способность ЛККШ ниже разрешающей способности псевдослучайных и классических кодовых шкал, маска которых выполнена в обыкновенном двоичном коде или в коде Грея. Однако K_p-последовательности, используемые при получении кодовой маски шкалы, позволяют в пределах одной разрядности реализовать ЛККШ с более широким диапазоном разрешающей способности.

СПИСОК ЛИТЕРАТУРЫ

1. *Преснухин Л. Н., Майоров С. А., Меськин И. В., Шаньгин В. Ф.* Фотоэлектрические преобразователи информации. М.: Машиностроение, 1974. 375 с.

2. *Ожиганов А. А.* Псевдослучайные кодовые шкалы для преобразователей линейных перемещений // Изв. вузов. Приборостроение. 1995. Т. 38, № 11—12. С. 37—39.

3. *Ожиганов А. А., Жуань Чжипэн.* Использование псевдослучайных последовательностей при построении кодовых шкал для преобразователей линейных перемещений // Изв. вузов. Приборостроение. 2008. Т. 51, № 7. С. 28—33.

4. *Ожиганов А. А., Жуань Чжипэн.* Критерий выбора длины линейной псевдослучайной кодовой шкалы // Изв. вузов. Приборостроение. 2010. Т. 53, № 5. С. 30—35.

5. *Ожиганов А. А., Тарасюк М. В.* Композиционные кодовые шкалы // Изв. вузов. Приборостроение. 1994. Т. 37, № 5—6. С. 26—29.

6. *Макуильямс Ф. Д., Слоан Н. Д.* Псевдослучайные последовательности и таблицы // ТИИЭР. 1976. Т. 64, № 12. С. 80—95.

7. *Яковлев В. В., Федоров Р. Ф.* Стохастические вычислительные машины. Л.: Машиностроение, 1974. 344 с.

8. *Ожиганов А. А., Тарасюк М. В.* Размещение считывающих элементов на композиционной кодовой шкале // Изв. вузов. Приборостроение. 1997. Т. 40, № 1. С. 42—47.

Сведения об авторе

Александр Аркадьевич Ожиганов — д-р техн. наук, профессор; Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики, кафедра вычислительной техники;
E-mail: ojiganov@mail.ifmo.ru

Рекомендована кафедрой вычислительной техники

Поступила в редакцию 08.02.12 г.

УДК 004.2, 681.3

И. С. Болгаров, Н. А. Маковецкая, А. Е. Платунов, Н. П. Постников

ПРОЕКТИРОВАНИЕ ПРИБОРНЫХ КОНТРОЛЛЕРОВ

Рассмотрены особенности встроенных вычислительных систем (приборных контроллеров), используемых в современных научных измерительных приборах и комплексах. Показана сложность разработки подобного рода систем, описаны тенденции и актуальные проблемы процесса проектирования. Предложены перспективные пути развития методов и технологий высокоуровневого проектирования посредством формального расширения пространства поиска решений.

Ключевые слова: *встроенная вычислительная система, приборный контроллер, автоматизация эксперимента, сканирующий зондовый микроскоп, глубоко программируемые системы, заказное проектирование, процесс проектирования, высокоуровневое проектирование.*

Введение. В современных научных измерительных и аналитических приборах и комплексах широко используются встроенные вычислительные системы (ВсС), выполняющие комплекс функций от сбора данных и управления экспериментом до финишной обработки, представления и архивирования полученной информации. Такие ВсС называются приборными контроллерами (ПрК).

ПрК относятся к „глубоко программируемым“ (SW-intensive) системам и обычно реализуют несколько уровней обработки информации эксперимента, сервисные и вспомогательные функции. При создании ПрК используются различные вычислительные архитектуры и технологии, что делает разработку этой категории ВсС сложной научно-технической задачей.

Востребованным направлением создания сложных ПрК с высокими инструментальными, метрологическими и пользовательскими характеристиками является заказное проектирование, охватывающее все основные уровни вычислительной иерархии ВсС. Такого рода разработки выполняются в ограниченных временных и финансовых рамках при условии высокой надежности проектирования, что определяет необходимость решения ряда сложных методологических проблем проектирования на базе широкого спектра информационно-коммуникационных и микроэлектронных технологий [1].

Методология заказного проектирования ПрК. Перечислим основные тенденции в развитии технологий и средств проектирования ВсС, которые непосредственно связаны с усилением значения этапов архитектурного, функционального, логического проектирования:

— повышение уровня абстракции проектирования;

— широкое применение моделирования, методов формального анализа и верификации моделей;

— введение уровня абстрактного представления вычислительного процесса;

— усовершенствование технологий создания встраиваемого программного обеспечения.

Данные этапы, относящиеся к области высокоуровневого проектирования (High Level Design, HLD), сегодня составляют до 70 % общего объема процесса разработки ВсС. По-прежнему широко используются традиционные технологии, относящиеся к низкоуровневому проектированию, когда архитектурное проектирование осуществляется на последующих фазах разработки. Выбор маршрута проектирования ПрК зависит, прежде всего, от сложности решаемой задачи и квалификации потенциальных разработчиков. Инструментарий сегодня достаточно развит и доступен, он не является ограничивающим фактором [2, 3].

На рис. 1 представлен упрощенный маршрут проектирования ВсС. От того, насколько проектировщики могут расширять зону работы с абстрактными моделями вычислительного процесса (ВП) и вычислительной системы в рамках всего маршрута проектирования, кардинально зависит качество проектирования ВсС.

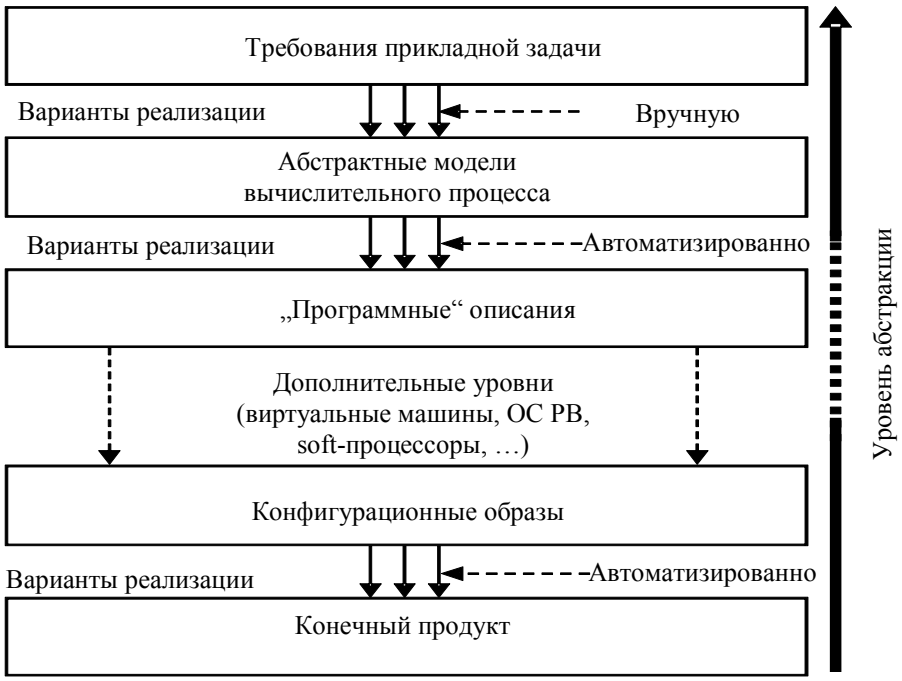


Рис. 1

Проектирование архитектуры ПрК. ПрК в зависимости от своей структуры, организации ВП, используемых вычислительных и языковых платформ могут быть разделены на группы: автономные, с хост-машиной (например, персональный компьютер), пространственно или/и функционально распределенные (многопроцессорные и сетевые структуры), с многоуровневой системой прикладного пользовательского программирования (СППП) и др.

На рис. 2 приведена структура вычислительных средств сканирующего зондового микроскопа (СЗМ), которые представляют собой ПрК, состоящий из ряда стандартных и специализированных аппаратных и программных блоков. ПрК обеспечивает многоуровневую организацию ВП в СЗМ, что позволяет гибко балансировать параметры целевого физического эксперимента и подстраиваться под требования различных категорий пользователей прибора (от студентов до экспертов в прикладной области).

Оценить сложность задач, решаемых разработчиком ПрК, и разнообразие вариантов их решения позволяют представленные ниже примеры. Первый из них относится к организации системы управления ПрК, второй иллюстрирует будни проектировщика, который решает задачу архитектурного проектирования значительной по сложности подсистемы ПрК, выбирая и осваивая инфокоммуникационные компоненты и технологии, с одной стороны, навязанные

техническим заданием, с другой полезные из соображений требуемой функциональности и повторного использования.

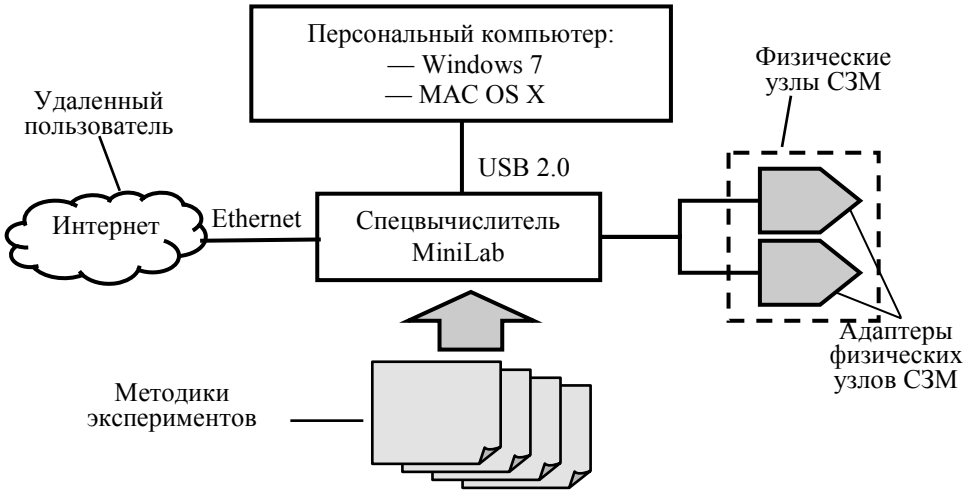


Рис. 2

Пример 1. Автоматизация эксперимента — основное назначение ПрК. Разработчик ПрК устанавливает общую направленность вычислительного процесса и задает базовые средства выражения алгоритма проведения эксперимента. Конечный пользователь описывает конкретный алгоритм эксперимента, используя средства прикладного программирования (конфигурирования) ПрК.

В случае СЗМ заказчиком озвучивались лишь общие пожелания по выполнению функций генерации сигналов, синхродетектирования, цифровой фильтрации и т.д. В связи с отсутствием четких требований к структуре вычислительного процесса было принято решение о разработке способа описания вычислительной схемы на основе функциональных блоков. Такое архитектурное решение обеспечило возможность эффективно реконфигурировать вычислительную схему для решения задач и достижения новых свойств. Был создан спецвычислитель (СВ) NL3 [4] на базе микросхемы программируемой логики Xilinx Spartan3E, а также разработано сопутствующее инструментальное ПО (компилятор NL3 и др.). Пользователю предоставлены система прикладного программирования эксперимента, отвечающая требованиям режима реального времени, а также средства прикладного программирования на языке Java. Таким образом, ПрК СЗМ предоставляет возможность гибкой декомпозиции алгоритма и реализации его в подходящей системе программирования.

Актуальность разработанной архитектуры ПрК подтверждается направлением исследований ведущих фирм в области вычислительной техники, однако на текущий момент существуют серьезные ограничения в использовании стандартных инструментов. Поэтому применение различного рода спецвычислителей оправдано, с одной стороны, ориентированностью на определенный класс задач, а с другой — возможностью оптимизации конечного продукта.

Способ распределения алгоритма по средам программирования влияет на качество ПрК как конечного изделия. Использование среды программирования „реального времени“ более трудоемко по сравнению с использованием среды программирования общего назначения. Для ускорения процесса разработки алгоритм может быть реализован на процессоре общего назначения с „потерей качества“. Дальнейшее снижение сложности и сроков программирования возможно при переносе алгоритмов эксперимента на хост-машину, что обычно применяется при создании прототипов.

Пример 2. В число подсистем ПрК СЗМ входит виртуальный осциллограф (ВО), предназначенный для визуализации внутренних процессов прибора на ПК под управлением ОС Windows, подключенном к СЗМ по каналу USB. ВО служит примером системы реального

времени с широкими возможностями конфигурирования пользователем. Ключевое требование ТЗ — работа в режиме реального времени со сравнительно большими объемами данных (порядка 100 000 срезов данных в секунду) — существенно повлияло на реализацию ВО.

Для обеспечения требуемой скорости передачи данных было принято решение нестандартным образом использовать штатные средства операционной системы (USB audio и kernel streaming). Основной сложностью явился выбор базового инструмента для реализации: в документации производителя ОС описаны разнообразные инструменты для работы с устройствами, но место каждого инструмента в общей картине функционирования ОС понять довольно трудно; в документации уделяется много внимания мелочам в ущерб описанию общей картины происходящего. „Пробелы“ документации пришлось восполнять в процессе проектирования. В ходе этой работы было реализовано несколько небольших экспериментальных проектов. В итоге затраты на реализацию этой части оказались довольно велики, причем значительная их доля пришлась не на решение собственно задачи, а на исследование свойств инструментария.

Этот пример может служить иллюстрацией того, как повторное использование относительно сложных инфокоммуникационных технологий сдерживается отсутствием эффективных выразительных средств для донесения их сути (на уровне назначения и архитектуры) до проектировщика и/или нежеланием поставщика изыскивать и использовать такие средства.

Развитие методов и технологий высокоуровневого проектирования ПрК. Представленные примеры убедительно демонстрируют значимость этапов архитектурного проектирования ПрК и необходимость совершенствования соответствующих методологий и инструментов высокоуровневого проектирования [6].

Однако это плохо сочетается с доминирующей моделью знаний в области вычислительной техники, в основе которой лежит каноническая организация ВС с центральным программируемым процессором интерпретирующего типа, на базе которого в рамках ОС организуется прикладной вычислительный процесс посредством создания программы на языке высокого уровня с последующей трансляцией в команды аппаратного процессора и вызовы функций ОС.

Данная модель профессиональных знаний пригодна в секторе прикладного программирования на стандартных аппаратно-программных платформах и неэффективна в области ВсС.

Изменить взгляд специалиста на организацию ВсС и повысить эффективность HLD можно, прежде всего, посредством формального расширения пространства поиска решений при создании ВсС за счет:

— формирования системы высокоуровневых (архитектурных) абстракций для представления ВсС (процесса проектирования, базовых элементов ВС, системного уровня, оценки архитектурных решений);

— выделения функциональных и нефункциональных требований технического задания в аспекты процесса проектирования;

— широкого использования принципа платформо-ориентированного проектирования;

— совместного проектирования инструментальной и целевой компоненты проекта;

— достижения динамического баланса „альтернатив“ проектирования между фазами „проектирование/реализация“ ВсС и „разработка/исполнение“ вычислительного процесса, а также между аппаратной и программной реализацией функций.

Заключение. Проектирование ПрК требует внедрения перспективных методов и технологий. Конкурентоспособность изделий, построенных на заимствованных шаблонных решениях, всегда будет невысока. Поэтому необходимо разумное сочетание технологий повторного использования и создания оригинальных продуктов. Взгляд на ВсС через призму высокоуровневого проектирования открывает перспективы преодоления кризиса сложности в инфокоммуникационной отрасли, поддерживая как эффективное освоение готовых решений и технологий, так и быстрое и надежное создание оригинальных продуктов. Это в первую оче-

редь актуально для таких областей проектирования, как сложные приборные контроллеры и системы.

СПИСОК ЛИТЕРАТУРЫ

1. *Платунов А. Е., Постников Н. П.* Перспективы формализации методов проектирования встроенных систем // Электронные компоненты. 2005. № 1. С. 24—29.

2. *Sangiovanni-Vincentelli A.* Quo Vadis SLD: Reasoning about Trends and Challenges of System-Level Design // Proc. IEEE. 2007. Vol. 95, is. 3. P. 467—506.

3. *Lee E., Neuendorffer S., Wirthlin M.* Actor-oriented design of embedded hardware and software systems // J. Circuits, Systems, and Computers. 2003. Vol. 12, N 3. P. 231—260.

4. *Ковязин Р. Р., Постников Н. П.* Разработка проблемно-ориентированных процессоров // Науч.-техн. вестн. СПбГУ ИТМО. 2010. Вып. 70. С. 81—85.

5. *Buschmann F., Buschmann F., Meunier R., Rohnert H., Sommerlad P., Stal M.* Pattern-Oriented Software Architecture // Chichester. NY—Toronto—Singapore: John Wiley and Sons Ltd., 2001. Vol. 1. A System of Patterns. P. 53—70.

6. *Platunov A. E., Kustarev P. V.* Problems of Abstract Representation of Embedded Systems at High-level Stages Design // Proc. Intern. Workshop on Networked embedded and control system technologies: European and Russian R&D cooperation (NESTER). 2009. P. 100—107.

Сведения об авторах

Иван Сергеевич Болгаров — ООО „ЛМТ“, Санкт-Петербург; инженер; E-mail: ivan.bolgarov@gmail.com

Наталья Андреевна Маковецкая — ООО „ЛМТ“, Санкт-Петербург; инженер; E-mail: mna@d1.ifmo.ru

Алексей Евгеньевич Платунов — д-р техн. наук, профессор; Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики, кафедра вычислительной техники; E-mail: platunov@lmt.ifmo.ru

Николай Павлович Постников — канд. техн. наук; ООО „ЛМТ“, Санкт-Петербург; главный инженер; E-mail: pnp@d1.ifmo.ru

Рекомендована кафедрой вычислительной техники

Поступила в редакцию 08.02.12 г.

УДК 004.2

А. В. Николаенков

АСПЕКТНЫЕ ТЕХНОЛОГИИ В СИСТЕМАХ С ПРЕОБЛАДАЮЩЕЙ ПРОГРАММНОЙ КОМПОНЕНТОЙ

Понятие аспекта положено в основу ряда подходов к разработке вычислительных систем, от технологии к технологии его трактовка различается. Проведен анализ использования понятия аспекта в технологиях создания программного обеспечения и в аспектной технологии проектирования встраиваемых систем.

Ключевые слова: *аспектная технология, аспектно-ориентированное программирование, субъектно-ориентированное программирование, композиционные фильтры, адаптивное программирование.*

Введение. Аспектные технологии [1] позволяют решать задачу проектирования встраиваемых систем (ВсС) на абстрактном уровне. Аспект используется для декомпозиции задачи проектирования и контроля качества ее решения. Схожие идеи положены в основу аспектных подходов к разработке программных систем. В настоящей работе анализируется трактовка аспекта в технологиях создания программного обеспечения (ПО) и аспектного проектирования ВсС.

Несмотря на значительные различия между ВсС и системами с доминирующей программной компонентой, на концептуальном уровне возможно использование схожих подходов к работе с аспектами. Сопоставление и анализ способов использования аспектов в области разработки программных систем будут полезны два развития аспектной технологий.

Сопоставление аспектной технологии проектирования и аспектных технологий программирования выполнялось в рамках работы [1], в настоящей статье приводится более детальный анализ.

Аспектные технологии программирования. Технология объектно-ориентированного проектирования (ООП) программных систем основана на принципах иерархической декомпозиции. В работе [2] наглядно показан ее главный недостаток: при построении иерархии между объектами реального мира предполагается наличие не зависящих от контекста рассмотрения иерархических связей. Для построения иерархии выделяется доминирующий критерий и второстепенные связи вытесняются „за скобки“. Отсутствие достаточного внимания к второстепенным критериям декомпозиции при написании программы приводит к появлению в программном коде повторяющихся конструкций (scattering) и перемешиванию реализации второстепенной функциональности с основной (tangling). Для работы с второстепенными (горизонтальными) связями вводится понятие аспекта. Анализ литературы выявил четыре основных реализации этих идей: аспектно-ориентированное программирование (АОП), субъектно-ориентированное программирование (СОП), композиционные фильтры (КФ), адаптивное программирование (АП).

В АОП наиболее развита реализация аспектных идей [3]. Основное инструментальное средство — AspectJ. Аспект рассматривается как единица модульности высокоуровневого объектно-ориентированного языка программирования и объединяется с элементами основной иерархии с помощью „точек включения“ (point-cuts). В рамках технологии АОП прорабатываются языковые средства для работы с аспектами и эффективные методы реализации модульности на основе аспектной идеи.

Идеи СОП были представлены в статье [4]. В рамках СОП-реализации организуются „субъекты“ и предлагаются методы для работы с ними. Идеи СОП развились в подход многомерного разделения ответственностей [5]. Несмотря на использование этих идей в инструментальном средстве Hyper/J, активных исследований в этой области не отмечено.

Подход КФ [6] позволяет рассматривать взаимодействующие объекты как обменивающиеся сообщениями сущности и предлагает набор инструментов для гибкой работы с сообщениями. На основе предложенных инструментов реализуются базовые механизмы ООП (наследование, агрегация) и АОП.

В АП программный код отделяется от структуры программы с последующим объединением непосредственно перед этапом выполнения программы. Аспектные идеи воплощаются в АП путем установки связей между реализациями аспектов и „точками включения“. Детальное описание АП представлено в статье [7]. Авторами проделана значительная работа по доказательству корректности предлагаемого подхода, рассмотрено внедрение идей АП не только в объектно-ориентированное, но и в функциональное программирование [8]. Одним из наиболее известных достижений АП можно считать формулирование закона Деметры [9], определяющего правильное построение связей между элементами программы.

Аспектная технология проектирования (АТП) [1] — высокоуровневая методика проектирования ВсС. В качестве аспектов АТП выделяет частные проблемы проектирования и предлагает подходы для работы с ними на всех этапах жизненного цикла ВсС. Примерами аспектов могут служить „надежность“, „стоимость“, „энергопотребление“. Задача проектирования рассматривается АТП как задача многокритериальной оптимизации в рамках определенных аспектами ограничений.

Широкая трактовка аспекта необходима для разработки ВсС с контролируемыми свойствами. Основное назначение аспекта АТП — построение абстракций для анализа свойств системы. Введение аспектных ограничений обеспечивает гибкость работы с проектным пространством и способствует проведению более тщательного анализа проектных решений.

Аспектные технологии программирования трактуют аспект как единицу модульности, упрощающую анализ системы и повышающую коэффициент повторного использования. Введение аспектов приводит к увеличению числа компонентов системы, а явное выделение уровня связей — к росту ее сложности и общей связности. Это обусловливает ограниченное применение аспектной технологии и использования ее для реализации ортогональных аспектов, содержащих небольшое количество простых механизмов. В качестве аспектов в программных системах выделяются журналирование, авторизация, аудит и др. Эффективная работа с аспектными технологиями при разработке программных систем требует поддержки аспектов на этапах проектирования и разработки требований.

В представленных подходах назначение аспектов существенно различается. Аспектные идеи, заложенные в основу рассматриваемых технологий, позволяют вводить дополнительные способы выделения компонентов и методы работы с ними. При использовании аспектных идей отмечаются общие проблемы организации взаимодействия между аспектами, анализа аспектных отношений, отсутствия унифицированных методов работы с аспектами на всех этапах жизненного цикла вычислительной системы.

Заключение. Для систем с преобладающей программной компонентой необходимо развивать средства высокоуровневого проектирования. Существующие аспектные методы могут выступать в качестве эффективного средства архитектурного проектирования при условии распространения понятия аспекта и его инструментальной поддержки на все этапы создания вычислительной системы.

СПИСОК ЛИТЕРАТУРЫ

1. *Платунов А. Е.* Теоретические и методологические основы высокоуровневого проектирования встраиваемых вычислительных систем: Автореф. дис. …д-ра техн. наук. СПб: СПбГУ ИТМО, 2010. 39 с.

2. *Ostermann K.* Modules for hierarchical and crosscutting models. Technischen Universitat Darmstadt, 2003.

3. *Kiczales G., Mezini M.* Aspect-oriented programming and modular reasoning // Proc. 27th Intern. Conf. on Software engineering — ICSE ’05. NY, USA: ACM Press, 2005. P. 49.

4. *Harrison W., Ossher H.* Subject-oriented programming // ACM SIGPLAN Notices. 1993. Vol. 28, N 10. P. 411—428.

5. *Tarr P. et al.* N Degrees of Separation: Multi-Dimensional Separation of Concerns // Proc. 21st Intern. Conf. on Software engineering — ICSE ’99. NY, USA: ACM Press, 1999. P. 107—119.

6. *Bergmans L., Aksits M.* Composing crosscutting concerns using composition filters // Communications of the ACM. 2001. Vol. 44, N 10. P. 51—57.

7. *Lieberherr K., Orleans D., Ovlinger J.* Aspect-oriented programming with adaptive methods // Communications of the ACM. ACM Press. 2001. Vol. 44, N 10. P. 39—41.

8. *Chadwick B.* Functional Adaptive Programming. Northeastern University, 2010.

9. *Lieberherr K., Holland I., Riel A.* Object-oriented programming: an objective sense of style // ACM SIGPLAN Notices. 1988. Vol. 23, N 11. P. 323—334.

Сведения об авторе

Алексей Витальевич Николаенков — аспирант; Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики, кафедра вычислительной техники; E-mail: nickolaenkov@gmail.com

Рекомендована кафедрой вычислительной техники

Поступила в редакцию 08.02.12 г.

УДК 621.3.85

А. А. Ожиганов, И. Д. Захаров

СИСТЕМА АВТОМАТИЗИРОВАННОГО ПРОЕКТИРОВАНИЯ ПСЕВДОРЕГУЛЯРНЫХ КОДОВЫХ ШКАЛ

Предлагается система автоматизированного проектирования псевдорегулярных кодовых шкал, основными задачами которой являются получение рисунка кодирующей маски шкалы и размещение на ее информационных дорожках считывающих элементов с учетом заданных физических ограничений.

Ключевые слова: *система автоматизированного проектирования, преобразователь перемещения, псевдорегулярная кодовая шкала, считывающие элементы.*

В современных информационно-измерительных системах широко применяются устройства аналого-цифрового преобразования, одним из видов которых являются преобразователи перемещений, построенные по методу считывания [1]. В качестве кодированного элемента в таких преобразователях могут быть использованы псевдорегулярные кодовые шкалы (ПРКШ) [2]. В силу особенностей этим шкалам свойственно большое разнообразие вариантов построения кодирующих масок, даже в пределах одной разрядности [3]. Поэтому разработку ПРКШ целесообразно выполнять с использованием системы автоматизированного проектирования.

Предлагаемая система, укрупненная схема которой показана на рис. 1, содержит несколько модулей, различающихся по функциональному назначению. Основной функцией вычислительного модуля является объектное представление проектируемой ПРКШ. Пользователь имеет возможность манипулировать ПРКШ при помощи графического интерфейса пользователя (ГИП), который выделяется в отдельный модуль. Отклик модуля ПРКШ на действия

пользователя, в свою очередь, отображается посредством модуля графического представления кодовой шкалы (ГПКШ). Особого внимания также заслуживает расширяемое множество модулей-генераторов, используемых для формирования кодирующих масок (КМ) дорожек ПРКШ.

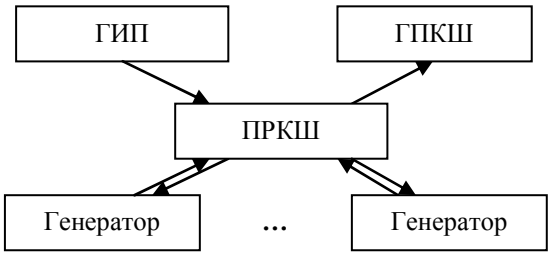


Рис. 1

Объектная иерархия модуля ПРКШ (рис. 2) спроектирована таким образом, чтобы максимально отразить технологические особенности проектируемого изделия. Так, младшим объектом иерархии является квант (q) кодовой дорожки (КД) [2]. Множество квантов можно объединить в абстракцию, представляющую собой один период информационной КД. Синтез КМ осуществляется посредством выбираемого объекта-генератора двоичной рекуррентной последовательности. Конкретный тип генератора определяется пользователем в процессе проектирования. Полученный объект соответствует периоду информационной КД, предоставляя вышестоящему объекту иерархии интерфейс для работы с периодом информационной КД в целом.

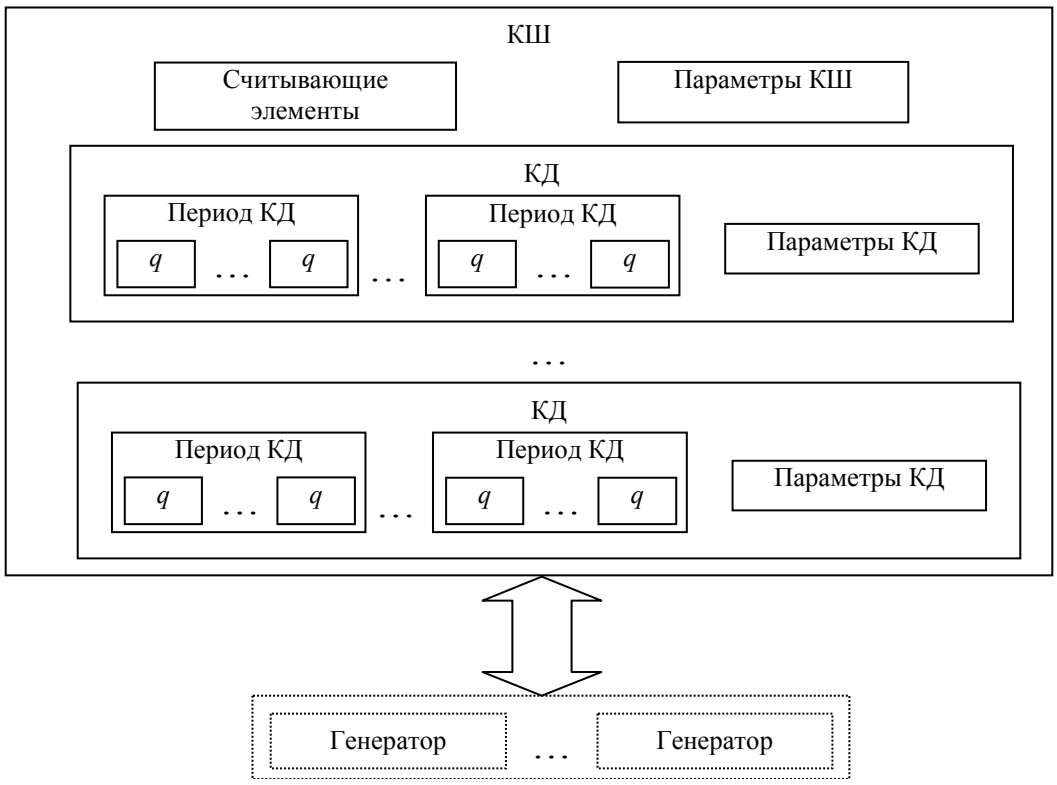


Рис. 2

Выше в объектной иерархии системы расположен объект КД, который функционально состоит из двух частей, отвечающих за логическую и физическую составляющие. Объекты периодов информационной КД составляют основу логической части объекта КД. В физической части хранятся либо вычисляются такие параметры КД, как диаметр, число периодов, линейные

размеры кванта и др. Объект КД предоставляет интерфейс для работы графического модуля, позволяющего рассчитывать местоположение конкретного кванта информационной КД.

Кодовая шкала является верхним уровнем объектной иерархии модуля ПРКШ. В функции КШ входит хранение списка объектов КД, предоставление интерфейса работы с ними, а также взаимодействие со считывающими элементами (СЭ) и их физическими параметрами. Для этих целей в состав КШ включена сущность, отвечающая за размещение СЭ на всех КД, из которых состоит данная КШ. Физические ограничения для КШ определяются как ее физическими параметрами (расстояние между КД и линейные размеры квантов каждой информационной КД), так и конструктивными особенностями преобразователя на основе ПРКШ в целом. Для устранения неоднозначности результатов считывания с ПРКШ информации СЭ должны размещаться с необходимой избыточностью [4].

Модули-генераторы различаются по типу генерируемых двоичных рекуррентных последовательностей. Такие модули состоят из двух частей — вычислительной и интерфейсной. В вычислительной части, выполняемой в отдельном потоке, реализуется алгоритм генерации двоичной рекуррентной последовательности, а интерфейсная часть предоставляет полученные последовательности модулю ПРКШ и осуществляет контроль над вычислительной частью, останавливая и возобновляя вычислительный поток. Известны алгоритмы формирования двоичных рекуррентных последовательностей, пригодные для формирования КМ шкалы и оптимизированные для использования в САПР кодовых шкал [3, 5]. Рассматриваемая в настоящей работе САПР ПРКШ позволяет унифицировать их реализацию, инкапсулируя вычислительную часть и предоставляя единый интерфейс.

Круговая ПРКШ представляет собой диск с нанесенной на него КМ в виде информационных КД и СЭ. Вдоль информационных КД размещены СЭ, с которых снимается информация об угловом положении ПРКШ.

В качестве входных данных модуля ПРКШ, получаемых из ГИП, выбираются внутренний и внешний диаметры КМ, минимальные линейные размеры кванта информационной КД, расстояние между КД, размеры СЭ, а также минимальные расстояния между ними.

Ширина i-й КД (i=1, …, k) вычисляется модулем ПРКШ в соответствии с выражением

$$\Delta_i = \frac{\left(D_{\text{ex}} - D_{\text{in}} - (k-1)w\right)}{k},$$

где D_{in} и D_{ex} — внутренний и внешний диаметр КМ, а w — минимальное расстояние между соседними КД.

С учетом того, что ширина всех КД, входящих в проектируемую КШ одинакова, внутренний диаметр D_i i-й КД может быть определен из соотношения

$$D_i = D_{\text{in}} + (\Delta_i + w)i = \frac{iD_{\text{ex}} + (k-i)D_{\text{in}} + iw}{k}.$$

Линейный размер кванта КД вычисляется в соответствии с выражением

$$q_i = \frac{\pi D_i}{N_i P_i},$$

где N_i — число квантов в одном периоде i-й информационной КД, а P_i — число периодов КМ i-й КД.

Число квантов каждой кодирующей маски КД определяется разработчиком ПРКШ и задается посредством выбора соответствующего генератора последовательности. В случае

формирования КМ i-й информационной КД на основе последовательностей де Брейна число ее квантов равно 2^n, где n — степень последовательности де Брейна.

Число периодов текущей информационной КД определяется рекуррентно из соотношения

$$P_i = P_{i-1} N_{i-1}, \quad P_0 = 1.$$

На сформированной КМ информационной КД размещаются СЭ, число которых равно степени последовательности, лежащей в основе построения маски конкретной дорожки.

Корректность размещения СЭ на КД проверяется путем нахождения всех кодовых комбинаций, получаемых при изменении угла поворота ПРКШ на один квант младшей КД. Размещение СЭ корректно тогда и только тогда, когда все получаемые с ПРКШ кодовые комбинации уникальны. В противном случае необходимо синтезировать другую КМ.

При синтезе каждой КД должны выполняться следующие условия.

1. Линейные размеры одного кванта информационной КД должны быть больше либо равны минимально различаемым размерам кванта для выбранных СЭ.

2. Расстояния S на КД между СЭ не должны превышать определенной для используемого типа СЭ величины:

$$S = \sqrt{r_1^2 \sin^2(\alpha_1 - \alpha_2) + (r_0 - r_1 \cos(\alpha_1 - \alpha_2))^2},$$

где r_0 и r_1 — расстояния между центром круговой КМ и центрами квантов, над которыми размещаются СЭ ($r_0 > r_1$); α_1 и α_2 — углы между заданной нулевой позицией (край кванта, с которого начинается отсчет) и квантами, над которыми размещаются СЭ.

Расстояния между центром круговой КМ и центром любого кванта i-й информационной КД определяются из выражения

$$r_i = D_i + \frac{\Delta_i}{2}.$$

В случае формирования КМ на основе последовательности де Брейна угол между нулевой позицией и j-м квантом i-й КД определяется из соотношения

$$\alpha_j^i = \frac{180°(2j+1)}{2^{n_i} P_i}.$$

Вследствие воздействия СЭ друг на друга, например выделяемым тепловым излучением, необходимо стремиться к максимальному увеличению расстояния между ними. Поэтому алгоритм перебора вариантов размещений СЭ в модуле ПРКШ должен начинаться с проверки таких случаев, когда СЭ распределены по шкале равномерно.

В работе представлена система автоматизированного проектирования ПРКШ, подробно рассмотрена объектная иерархия основного вычислительного модуля. На основе рассмотренного подхода реализована САПР ПРКШ на языке C++/Qt4. Подобную систему можно применять для проектирования преобразователей угловых перемещений, использующих различные виды ПРКШ.

СПИСОК ЛИТЕРАТУРЫ

1. *Домрачев В. Г., Мейко Б. С.* Цифровые преобразователи угла: принципы построения, теория точности, методы контроля. М.: Энергоатомиздат, 1984. 328 с.

2. *Ожиганов А. А., Прибыткин П. А.* Псевдорегулярные кодовые шкалы для цифровых преобразователей угла // Науч.-техн. вестн. СПбГУ ИТМО. 2011. Вып. 1. С. 67—72.

3. *Ожиганов А. А., Захаров И. Д.* Применение последовательностей де Брейна для построения псевдорегулярных кодовых шкал // Науч.-техн. вестн. НИУ ИТМО. 2012. Вып. 2(78). С. 69—74.

4. *Азов А. К., Ожиганов А. А.* Устранение неоднозначности считывания в преобразователях перемещения с рекурсивными кодовыми шкалами // Информационные системы. 2001. № 6. С. 39—42.

5. *Ожиганов А. А., Захаров И. Д.* Использование порождающих полиномов М-последовательностей при построении псевдослучайных кодовых шкал // Изв. вузов. Приборостроение. 2011. Т. 54, № 6. С. 49—56.

Сведения об авторах

Александр Аркадьевич Ожиганов — д-р техн. наук, профессор; Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики, кафедра вычислительной техники; E-mail: ojiganov@mail.ifmo.ru

Илья Дмитриевич Захаров — аспирант; Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики, кафедра вычислительной техники; E-mail: zakharov_ilya@hotmail.com

Рекомендована кафедрой вычислительной техники

Поступила в редакцию 08.02.12 г.

УДК 004.031.6

С. О. ЧУРАЕВ

МЕТОД РЕВЕРСИВНОЙ СЛУЧАЙНОЙ ВЫБОРКИ ДЛЯ ИЗМЕРЕНИЯ С ПИКОСЕКУНДНЫМ РАЗРЕШЕНИЕМ ВРЕМЕНИ ЗАДЕРЖКИ В ЭЛЕМЕНТАХ ИНТЕГРАЛЬНЫХ СХЕМ

Предложен обеспечивающий пикосекундную точность метод измерения времени задержки по фронту и спаду сигнала в наноразмерном логическом вентиле интегральной схемы. Приведено схемотехническое решение, реализующее данный метод измерения.

Ключевые слова: *метод случайной выборки, встраиваемые времяцифровые преобразователи, пикосекундное разрешение, разделение задержек по фронту и спаду, инерциальная задержка.*

Введение. При проектировании цифровых наноразмерных интегральных схем (ИС) одной из актуальных проблем остается высокоточное измерение временных параметров логических вентилей. К основным параметрам относится значение времени задержки переключения по фронту t_r и спаду сигнала t_f, которое используется в моделях элементов при проектировании схемы в САПР. До недавнего времени данная задача решалась с помощью сравнительно простых внешних измерительных устройств, таких как осциллограф и генератор, а также линии задержки, включающей 5—10 и более однотипных, последовательно соединенных элементов.

Существует класс измерительных схем, условно классифицируемых как *встраиваемые* времяцифровые преобразователи (ВВЦП). Это специализированные схемы, выполненные по единому технологическому процессу и размещаемые вблизи объекта тестирования, обеспечивающие интенсивное контролируемое взаимодействие с ними [1]. Разместив ВВЦП рядом с объектом тестирования, можно существенно снизить влияние оборудования на объект измерения, снизить трудоемкость изготовления, энергопотребление и время измерения. В ВВЦП реализуются методы прямого отсчета, нониусного совпадения фаз, Уилкинсона, случайной выборки (МСВ) [2], фазового накопления ошибки [3, 4].

Метод случайной выборки. В классической задаче теории вероятностей определяется вероятность P попадания случайной равномерно распределенной величины X (СВ X) на известный участок длиной L, лежащий на известном интервале длиной T.

В МСВ используется предположение о том, что вероятность числа попаданий m СВ X на измеряемый участок пропорциональна L и не зависит от его положения на оси абсцисс, при условии, что события „попадания“ в измеряемый участок $P(A)$ и „промаха“ $P(\bar{A})$ СВ X будут образовывать полную группу вероятностей $P(A)+P(\bar{A})=1$. Для этого интервал T разбивается на k равных элементарных отрезков длиной Δt.

Тогда, зная значение T и полагая, что L связана функционально-вероятностной зависимостью с P, получим:

$$L = f\left(P(X),T\right) = T\frac{m}{m+s} = k\Delta t\frac{m}{m+s}. \qquad (2)$$

При подаче тактового сигнала тестовый элемент проходит четыре циклично сменяющихся состояния „00“, „01“, „10“, „11“, из которых информативны лишь два: „01“ — задержка по фронту t_r и „10“ — задержка по спаду сигнала t_f. Частота выборки участков фронта и спада для всех интервалов пропорциональна длине этих участков. Таким образом, приравняв значение длины интервала T к периоду тактовой частоты T_{ref}, можно рассчитать время задержки по фронту и спаду сигнала:

$$t_r = \frac{m_r}{m_r+s}T_{\text{ref}}; \quad t_f = \frac{m_f}{m_f+s}T_{\text{ref}},$$

где m_r и m_f — количество попаданий СВ X в участки фронта и спада соответственно. Технически „попадания“ СВ X реализуются за счет использования элементов выборки и хранения (*S*/*H*), соединенных со входом и выходом тестируемого элемента и управляемых случайным сигналом. В дальнейшем захваченная пара сигналов подается на вход декодера „2 в 4“, где преобразуется в один из четырех сигналов разрешения работы счетчиков случайных импульсов. Поэтому через некоторое время в счетчиках начинает формироваться число, пропорциональное длине соответствующего „участка“ измерения, которое затем пересчитывается с помощью T_{ref} в величину задержки сигнала.

Метод реверсивной случайной выборки (МРСВ) с сокращением неинформативных интервалов. Основная идея метода строится на предположении о том, что задержка в тестовом элементе по отношению к длительности периода тактового сигнала $t_{r,f}/T_{\text{ref}}$ мала и является величиной постоянной. Поэтому для того чтобы уменьшить дисперсию результата измерения, следует увеличить это соотношение.

Наиболее просто уменьшить период тактовой частоты T_{ref}. Но невозможно бесконечно увеличивать частоту тактового сигнала в связи с шунтирующим действием контактной группы. Не решает проблемы и использование размещаемого на кристалле модуля умножителя частоты с автоподстройкой фазы. Кроме того, использование случайной частоты для управления элементами выборки и хранения, а также счетчиками может приводить к непредсказуемой работе схемы вследствие появления метастабильных состояний в измерительной схеме. Поэтому следует найти альтернативный способ сокращения длины неинформативных участков низкого „00“ и высокого „11“ уровней относительно величины T_{ref}.

Обозначим общее время измерения как „*диапазон*“ $D = NT_{\text{ref}}$ (N — число тактов). Рассмотрим случай, когда на вход буферного элемента подается случайный сигнал, а на вход элементов выборки/хранения и управляющий вход счетчиков — тактовый (рис. 1).

Момент выборки по фронту тактового сигнала по отношению к смене состояний в схеме будет являться также случайным. При этом элементарные события попадания в L и промаха не будут формировать полную группу событий для одного периода T_{ref}: $P(A)+P(\overline{A})\neq 1$. Это условие выполнимо только для всего диапазона измерения, который будет представлять собой сумму n всех участков фронта — R_i, спада — F_i и неинформативных участков диапазона D „00“ L_i и „11“ H_i. Сгруппировав сходные участки измерения, получим:

$$\sum_{i=1}^{n} L_i + \sum_{i=1}^{n} H_i + \sum_{i=1}^{n} R_i + \sum_{i=1}^{n} F_i = NT_{\text{ref}}. \tag{3}$$

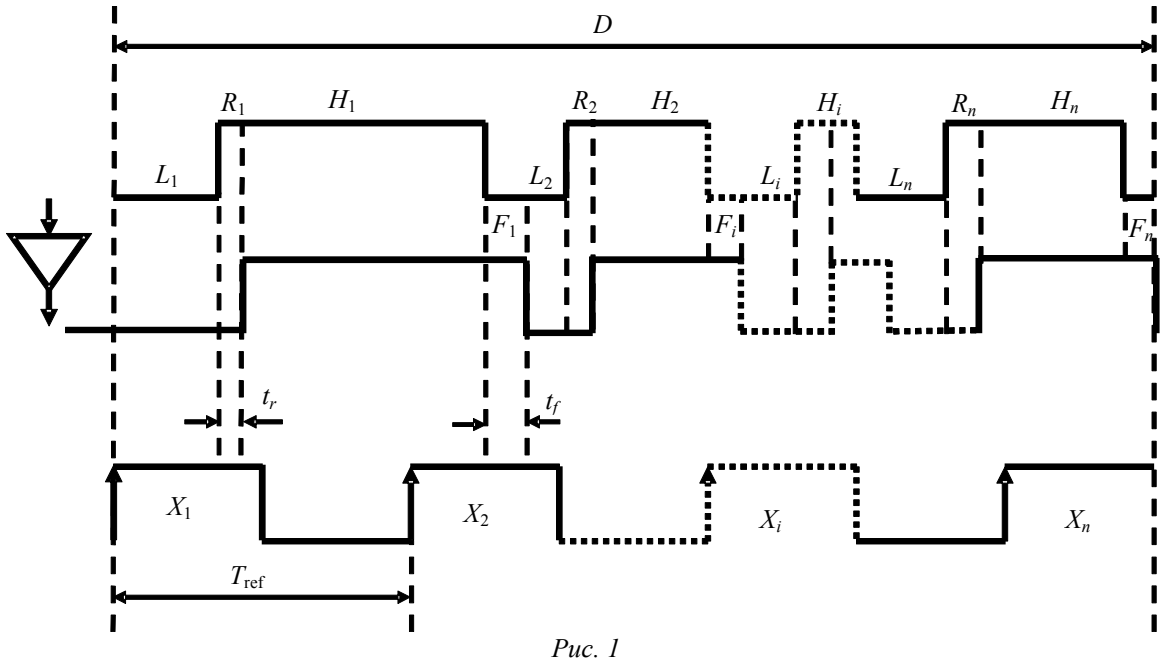


Рис. 1

Учитывая, что $R_1=R_2=\ldots=R_n$ и $F_1=F_2=\ldots=F_n$, запишем:

$$\sum_{i=1}^{n} L_i + \sum_{i=1}^{n} H_i + n(R+F) = NT_{\text{ref}}. \tag{4}$$

Разделив на число случайных импульсов n правую и левую части, запишем:

$$\frac{\sum_{i=1}^{n} L_i}{n} + \frac{\sum_{i=1}^{n} H_i}{n} + \frac{n(R+F)}{n} = \frac{NT_{\text{ref}}}{n}. \tag{5}$$

Величина $\dfrac{\sum_{i=1}^{n} L_i}{n}$ (и $\dfrac{\sum_{i=1}^{n} H_i}{n}$) есть не что иное, как среднее арифметическое, которое при увеличении n сходится по величине к математическому ожиданию μ_H (μ_L), тогда:

$$\mu_H + \mu_L + R + F = \frac{NT_{\text{ref}}}{n}. \tag{6}$$

Значение L можно вычислить по формуле

$$L = T_{\text{ref}} P \frac{N}{m+s} = T_{\text{ref}} \frac{m}{m+s} \frac{N}{R_n}, \tag{7}$$

где R_n — число периодов (фронтов) случайного сигнала. Таким образом, при аппаратной реализации МРСВ в схему должны добавиться счетчики числа случайных периодов сигнала n и периодов тактового сигнала N (рис. 2).

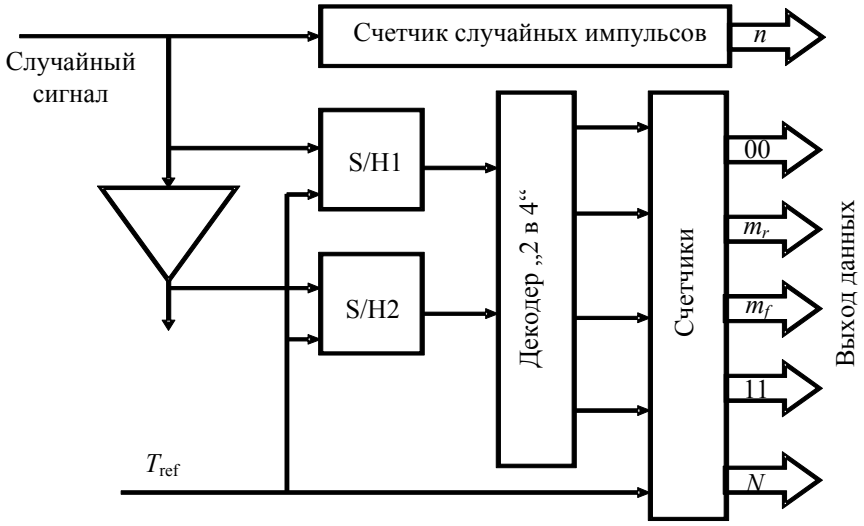


Рис. 2

К особенностям работы схемы следует отнести то, что счетчик случайных импульсов должен иметь рабочий период меньше минимально возможного времени появления случайных фронтов. Этого можно добиться, используя сумматор по модулю 2 массива кольцевых генераторов различной длины для формирования случайной частоты, а наиболее быстродействующий кольцевой генератор — для тактирования конечного автомата, подсчитывающего число случайных фронтов.

Опытным путем было установлено, что при замене опорной частоты на случайную, и наоборот, в схеме на рис. 2 количество попаданий в участок фронта m_r и спада m_f увеличивается, что позволяет более прецизионно измерять время задержки. Однако время тестирования увеличивается пропорционально уменьшению количества выборок.

МРСВ работает при условии, что максимально возможный период генерации случайного сигнала на всем диапазоне измерения меньше периода опорного сигнала. Таким образом, для увеличения эффективности работы МРСВ необходимо повышать частоту генерации случайного сигнала, но при этом не допускать работу на очень высоких частотах в граничных условиях, при которых в счетчике случайных импульсов возможно появление метастабильных состояний.

В качестве варианта решения данной задачи в схеме вместо генерации случайного сигнала может быть использован простой кольцевой „псевдослучайный“ генератор. Вследствие отсутствия синхронизации между внешним генератором (опорной частоты) и внутренним („псевдослучайным“) постоянно возрастает разность фаз, усиленная эффектами фазового дрожания сигнала, имеющего нормально распределенный характер. Тогда фазу тактирующего сигнала можно считать случайной по отношению к фазе внутреннего кольцевого генератора. Таким образом, можно более точно предсказать минимально возможную частоту работы счетчика случайных импульсов уже на этапе проектирования средствами САПР.

Необходимо отметить, что схема позволяет оценить так называемую инерциальную задержку элемента. При уменьшении длительности случайного импульса на выходах схемы формируются „ложные“ состояния фронта и спада сигнала, которые добавляются в конечный результат измерения. Зная точное значение времени задержки и вычтя его из полученного результата, можно косвенно оценить величину инерциальной задержки элемента.

Выводы. Таким образом, в данной реализации схемы частота появления участков фронта и спада сигнала существенно выше, чем в схеме, реализующей стандартный МСВ и, следовательно, количество неинформативных участков меньше. За счет этого возможно повысить прецизионность измерения, не прибегая к увеличению тактовой частоты опорного генератора или усложнению схемной реализации дополнительными модулями. Кроме того, требования к стабильности опорной частоты в данном случае играют второстепенную роль, уступая требованию формирования точного промежутка времени, характеризующего весь диапазон измерений D. В дальнейшем используя различные методы нониусного старт-стопного механизма, можно еще понизить ошибку измерения.

СПИСОК ЛИТЕРАТУРЫ

1. *Платунов А. Е.* Теоретические и методологические основы высокоуровневого проектирования встраиваемых вычислительных систем. Дис. … докт. техн. наук. СПб: НИУ ИТМО, 2011.

2. *Maggioni S., Veggetti A., Bogliolo A., Croce L.* Random sampling for on-chip characterization of standard cell propagation delay // Proc. Intern. Symp. on Quality Electronic Design. 2003. P. 41—45.

3. *Ruffoni M., Bogliolo A.* Direct Measures of Path Delays on Commercial FPGA Chips // Proc. 6th IEEE Workshop on Signal Propagation on Interconnects. 2002. P. 157—159.

4. *Churayev S., Biryuchinskiy S., Melnikov K., Paltashev T.* Phase shift accumulation method for timing characterization // Proc. IEEE 2nd Intern. Conf. on Photonics (ICP). 2011. P. 1—5.

Сведения об авторе

Сергей Олегович Чураев — аспирант; Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики, кафедра вычислительной техники; E-mail: sergey.churayev@yahoo.com

Рекомендована кафедрой вычислительной техники

Поступила в редакцию 08.02.12 г.

SUMMARY

P. 7—13.

CONTEXT MANAGEMENT IN INFORMATION SYSTEMS

The problem of creation an intelligent agent in information systems based on Semantic Web principles is considered. The necessity of including the context data in search query is justified. A conceptual model of intelligent agent implementing the search with automatic context control is proposed.

Keywords: information system, context, ontology, information retrieval.

Data on author

Igor A. Bessmertny — Cand. Techn. Sci.; St. Petersburg National Research University of Information Technologies, Mechanics and Optics; Department of Computer Technology;
E-mail: igor_bessmertny@hotmail.com

P. 13—19.

APPLICATION OF CUDA TECHNIQUE TO QUICKEN CALCULATION OF EUROPEAN OPTION PRICES BY FINITE-DIFFERENCE METHOD

Crank—Nicolson scheme for Black–Scholes partial differential equation is fully implemented on graphics processor using CUDA technique. The developed code on GPU NVIDIA GTX 580 works more than 20 times faster than the single-threaded calculation on CPU Intel Core i7 3.4 GHz, and 2—3 times faster than the best results obtained with multi-thread version based on GCD technique on CPUs 2 x Intel Xeon 3.06 GHz with 24 cores in conditions typical for high-frequency algorithmic trading systems.

Keywords: CUDA, GPGPU, parallel cyclic reduction, PCR, algorithmic trading, high-frequency trading, option, Crank—Nicolson scheme.

Data on authors

Mikhail S. Kosyakov — Cand. Techn. Sci,; St. Petersburg National Research University of Information Technologies, Mechanics and Optics; Department of Computer Technology;
E-mail: mkosyakov@gmail.com

Dmitry N. Shinkaruk — Post-Graduate Student; St. Petersburg National Research University of Information Technologies, Mechanics and Optics; Department of Computer Technology;
E-mail: dimashink@gmail.com

Alexander V. Toropov — Post-Graduate Student; St. Petersburg National Research University of Information Technologies, Mechanics and Optics; Department of Computer Technologies; Tbrix AB, engineer-programmer; E-mail: toropov@rain.ifmo.ru

Yury A. Shpolyanskiy — Dr. Phys.-Math. Sci.; St. Petersburg National Research University of Information Technologies, Mechanics and Optics; Department of Photonics and Optoinformatics; Tbrix AB, leading mathematician; E-mail: shpolyan@mail.ru

P. 20—25.

ANALYSIS OF CUDA EFFICIENCY IN SOLVING LINEAR TRIDIAGONAL SYSTEMS FOR THEORETICAL OPTION PRICING

Parallel cyclic reduction method for solving linear tridiagonal systems is implemented on GPU. The advisability of matrix formation directly in GPU global memory is shown. The approach provides a more than 20-fold acceleration as compared to single-threaded calculation. With the account for data transfer between RAM and GPU, a 5—8-fold acceleration is attained with the use of mapped memory.

Keywords: CUDA, GPGPU, systems of linear equations, parallel cyclic reduction, PCR, sweep method.

Data on authors

Dmitry N. Shinkaruk — Post-Graduate Student; St. Petersburg National Research University of Information Technologies, Mechanics and Optics; Department of Computer Technology; E-mail: dimashink@gmail.com

Yury A. Shpolyanskiy — Dr. Phys.-Math. Sci.; St. Petersburg National Research University of Information Technologies, Mechanics and Optics; Department of Photonics and Optoinformatics; Tbrix AB, leading mathematician; E-mail: shpolyan@mail.ru

Mikhail S. Kosyakov — Cand. Techn. Sci.; St. Petersburg National Research University of Information Technology, Mechanics and Optics; Department of Computer Technology; E-mail: mkosyakov@gmail.com

P. 26—30.

STUDY OF TRANSFORM CODING ALGORITHMS IN COMPRESSION OF VIDEO SEQUENCE FRAMES

A fast algorithm of transform coding for compression of intra and residual frames of video sequences is proposed. The algorithm is based on application of three-dimensional Hartley transform with fixed and variable size of the transformation matrix.

Keywords: transform coding, Hartley transform, three-dimensional algorithm, variable size matrix, image model.

Data on authors

Irina S. Rubina — Post-Graduate Student; St. Petersburg National Research University of Information Technologies, Mechanics and Optics; Department of Computer Technology; E-mail: rubren@mail.ru

Alexander Yu. Tropchenko — Dr. Techn. Sci., Professor; St. Petersburg National Research University of Information Technologies, Mechanics and Optics; Department of Computer Technology; E-mail: tau@d1.ifmo.ru

P. 31—36.

NEURAL NETWORK METHODS OF PERSON IDENTIFICATION BY FACE IMAGE

Neural network methods of recognition of a person by his face image are considered, the methods being employed in biometric identification systems.

Ключевые слова: neural networks, person recognition, biometric system.

Data on authors

Andrey A. Tropchenko — Cand. Techn. Sci.; St. Petersburg National Research University of Information Technologies, Mechanics and Optics; Department of Computer Technology; E-mail: zayka_98rus@mail.ru

Alexander Yu. Tropchenko — Dr. Techn. Sci., Professor; St. Petersburg National Research University of Information Technologies, Mechanics and Optics; Department of Computer Technology; E-mail: tau@d1.ifmo.ru

P. 37—40.

MODEL OF CONTROL OVER ACCESS TO NEWLY CREATED FILE OBJECTS

A method of control over access to newly created file objects is developed. In contrast to existing methods, the new approach excludes the "object" role from the access policy. The proposed model of access control is applied for development of requirements to mandatory information threads management affording security of developed system.

Keywords: information security, unauthorized access preventing, access rights and permissions, access control, newly created file object, information thread.

Data on authors

Konstantin A. Shcheglov — Student; St. Petersburg National Research University of Information Technologies, Mechanics and Optics; Department of Computer Technology; E-mail: schegl_70@mail.ru

Andrey Yu. Shcheglov — Dr. Techn. Sci., Professor; St. Petersburg National Research University of Information Technologies, Mechanics and Optics; Department of Computer Technology; E-mail: info@npp-itb.spb.ru

P. 41—46.

CONVERSION OF ALGORITHM MODELS

Conversion of regular expressions to finite automata and inverse conversion related to the theory of algorithms and automata are considered. Conversions of these models to block diagrams and backwards are presented. Application of the theory of algorithms to practical problems in algorithmic programming languages and testing of programs is discussed.

Keywords: regular expressions, finite automata, model algorithms, block diagram.

Data on authors

Vladimir I. Polyakov — Cand. Techn. Sci.; St. Petersburg National Research University of Information Technologies, Mechanics and Optics; Department of Computer Technology; E-mail: v_i_polyakov@mail.ru

Vladimir I. Skorubsky — Cand. Techn. Sci.; St. Petersburg National Research University of Information Technologies, Mechanics and Optics; Department of Computer Technology; E-mail: vlis@km.ru

P. 47—52.

CREATING A CAD SIMULATOR OF A ROUTED COMPUTER NETWORK USING OPEN SOURCE COMPONENTS

Problems of creating a CAD simulator of a computer network with support of high OSI level functions in routers are discussed. An approach to solve these problems is proposed, an analysis is carried out and technical details are described.

Keywords: NS-3, QoS, Qt, simulation, computer network, router, tunneling, priority queuing, WFQ, PQ, CQ, LLQ.

Data on authors

Taufik I. Aliev — Dr. Techn. Sci., Professor; St. Petersburg National Research University of Information Technologies, Mechanics and Optics; Department of Computer Technology; Head of the Department; E-mail: aliev@d1.ifmo.ru

Vladimir V. Sosnin — St. Petersburg National Research University of Information Technologies, Mechanics and Optics; Department of Computer Technology; Assistant Lecturer; E-mail: vsosnin@mail.ru

Dmitry N. Shinkaruk — Post-Graduate Student; St. Petersburg National Research University of Information Technologies, Mechanics and Optics; Department of Computer Technology; E-mail: dimashink@gmail.com

Mikhail Yu. Tikhonov — Student; St. Petersburg National Research University of Information Technologies, Mechanics and Optics; Department of Computer Technology; E-mail: tihmihail@gmail.com
Nikita G. Burmakin — Post-Graduate Student; St. Petersburg National Research University of Information Technologies, Mechanics and Optics; Department of Computer Technology; E-mail: nekit.mail@gmail.com

P. 53—57.

FUNCTIONAL RELIABILITY OF COMPUTING SYSTEMS WITH REDISTRIBUTION OF INQUIRIES

A method for estimation of functional reliability of computing systems is proposed. The method make use of probability of inquiry execution by the system in a time not exceeding the maximum permissible value. The estimation of functional reliability considers possibilities of adaptation to refusals and changes of inquiry stream as a result of redistribution of inquiries between computing knots through the network. Efficiency of inquiries redistribution through a network is demonstrated.

Keywords: fault tolerance, distributed computing systems, redistribution of inquiries, cluster, optimization, real time.

Data on authors

Vladimir A. Bogatyrev — Dr. Techn. Sci., Professor; St. Petersburg National Research University of Information Technologies, Mechanics and Optics; Department of Computer Technology; E-mail: Vladimir.bogatyrev@gmail.com
Stanislav V. Bogatyrev — IT House, Ltd., St. Petersburg; Chief Engineer; E-mail: realloc@gmail.com
Anatoly V. Bogatyrev — Post-Graduate Student; St. Petersburg National Research University of Information Technologies, Mechanics and Optics; Department of Computer Technology; E-mail: ganglion@gmail.com

P. 57—63.

PROBLEMS OF SYNTHESIS OF SYSTEMS WITH LOSSES

Problems of synthesis of systems with losses are formulated with account for restrictions on characteristics of the system functioning. In the process of the synthesis, the buffer capacity and the system performance are defined to minimize the system cost.

Keywords: system with losses, capacity of buffer, device performance, probability of loss, delay time, cost of system.

Data on author

Taufik I. Aliev — Dr. Techn. Sci., Professor; St. Petersburg National Research University of Information Technologies, Mechanics and Optics; Department of Computer Technology; Head of the Department; E-mail: aliev@d1.ifmo.ru

P. 64—68.

PRIORITY-BASED MECHANISMS OF SERVICE QUALITY PROVISION IN MULTISERVICE COMPUTER NETWORKS

Models and methods for estimation of characteristics of multiservice computer networks with priority-based management of heterogeneous traffic are considered. Provision of QoS requirements by means of the choice of the proper priority-based mechanisms is discussed.

Keywords: multiservice computer networks, priority-based management, quality of service, heterogeneous traffic, model of functioning.

Data on author

Ludmila A. Muravyeva-Vitkovskaya — Cand. Techn. Sci.; St. Petersburg National Research University of Information Technologies, Mechanics and Optics; Department of Computer Technology; E-mail: mur-lada@yandex.ru

P. 69—73.

COMPOSITE CODE SCALES FOR CONVERTERS OF LINEAR MOVEMENTS

A method of creation of linear composite code scales with single information code path is considered. An example of the scale created with the use of the proposed method is presented.

Keywords: composite sequence, code scale, the linear composite code scale, reading-out elements

Data on author

Alexander A. Ozhiganov — Dr. Techn. Sci., Professor; St. Petersburg National Research University of Information Technologies, Mechanics and Optics; Department of Computer Technology; E-mail: ojiganov@mail.ifmo.ru

P. 73—77.

DESIGN OF INSTRUMENT CONTROLLERS

Special features of embedded computer systems (instrument controllers) in modern scientific measuring devices and complexes are considered. Difficulties of the systems design are discussed; tendencies and actual problems of the design process are described. Perspective approaches to development of high-level design methods and technologies by means of formal expansion of solution search space are proposed.

Keywords: embedded system, instrument controller, experiment automation, scanning probe microscope, SW-intensive systems, custom design, design process, high level design.

Data on authors

Ivan S. Bolgarov — LMT Ltd., St. Petersburg; Engineer; E-mail: ivan.bolgarov@gmail.com

Natalia A. Makovetskaya — LMT Ltd., St. Petersburg; Engineer; E-mail: mna@d1.ifmo.ru

Alexey E. Platunov — Dr. Techn. Sci., Professor; St. Petersburg National Research University of Information Technologies, Mechanics and Optics; Department of Computer Technology; E-mail: platunov@lmt.ifmo.ru

Nikolay P. Postnikov — Cand. Techn. Sci.; LMT Ltd., St. Petersburg; Chief Engineer; E-mail: pnp@d1.ifmo.ru

P. 78—80.

ASPECT TECHNOLOGIES IN DEVELOPMENT OF SW-INTENSIVE SYSTEMS

A comparative analysis of the use of concept of an aspect in aspect-oriented programming technologies and in technology of embedded system design is carried out.

Keywords: aspect technology, aspect-oriented programming, subject-oriented programming, compositional filters, adaptive programming.

Data on author

Alexey V. Nikolaenkov — Post-Graduate Student; St. Petersburg National Research University of Information Technologies, Mechanics and Optics; Department of Computer Technology;
E-mail: nickolaenkov@gmail.com

P. 80—84.

A COMPUTER-AIDED SYSTEM FOR PSEUDO-REGULAR CODE SCALE DESIGN

A computer-aided system for design of pseudo-regular code scales is proposed. The main goals of the system include creation of the code-scale coding pattern and reading heads placement according to the code-scale physical parameters.

Keywords: CAD, analogue-to-digital converter, pseudo-regular code-scale, reading heads.

Data on authors

Alexander A. Ozhiganov — Dr. Techn. Sci., Professor; St. Petersburg National Research University of Information Technologies, Mechanics and Optics; Department of Computer Technology;
E-mail: ojiganov@mail.ifmo.ru

Ilya D. Zakharov — Post-Graduate Student; St. Petersburg National Research University of Information Technologies, Mechanics and Optics; Department of Computer Technology;
E-mail: zakharov_ilya@hotmail.com

P. 84—88.

METHOD OF REVERSE RANDOM SAMPLING FOR MEASUREMENT OF TIME DELAY IN ELEMENT OF THE INTEGRATED CIRCUIT WITH PICOSECOND ACCURACY

A new method of propagation delay measurement in micro- and nanostructures is described. The method makes it possible to observe the propagation time delay in a single element of standard-cell library and is accurate to better than 1 picoseconds.

Keywords: random-sampling standard cell characterization, phase error accumulation methodology, gate propagation delay, high precision on-chip measurement, signal processing, BIST, DFT, memory, processor testing, MEMS testing.

Data on author

Sergey O. Churayev — Post-Graduate Student; St. Petersburg National Research University of Information Technologies, Mechanics and Optics; Department of Computer Technology;
E-mail: sergey.churayev@yahoo.com