ТЕОРЕТИЧЕСКИЙ И ПРИКЛАДНОЙ НАУЧНО-ТЕХНИЧЕСКИЙ ЖУРНАЛ

Издается с сентября 2010 г.

Журнал издается при поддержке Отделения математических наук РАН, Отделения нанотехнологий и информационных технологий РАН, МГУ имени М.В. Ломоносова, МГТУ имени Н.Э. Баумана, ОАО "Концерн "Сириус".

СОДЕРЖАНИЕ

Журнал зарегистрирован в Федеральной службе по надзору в сфере связи, информационных технологий и массовых коммуникаций. Свидетельство о регистрации ПИ № ФС77-38590 от 24 декабря 2009 г.

Журнал распространяется по подписке, которую можно оформить в любом почтовом отделении (индексы: по каталогу агентства "Роспечать" — **22765**, по Объединенному каталогу "Пресса России" — **39795**) или непосредственно в редакции.
Тел.: (499) 269-53-97. Факс: (499) 269-55-10.
Http://novtex.ru/pi.html E-mail: prin@novtex.ru

Журнал включен в систему Российского индекса научного цитирования.
Журнал входит в Перечень научных журналов, в которых по рекомендации ВАК РФ должны быть опубликованы научные результаты диссертаций на соискание ученой степени доктора и кандидата наук.

THEORETICAL AND APPLIED SCIENTIFIC AND TECHNICAL JOURNAL

SOFTWARE ENGINEERING

PROGRAMMNAYA INGENERIA

№ 5
May
2014

Published since September 2010

CONTENTS

Information about the jornal is available online at:
http://novtex.ru/pi.html, e-mail:prin@novtex.ru

УДК 004.75+004.41

А. С. Шундеев, канд. физ.-мат. наук, вед. науч. сотр., НИИ механики МГУ имени М. В. Ломоносова,
e-mail: alex.shundeev@gmail.com

Система распределенных вычислений на базе платформы ERLANG/OTP

Представлена архитектура программной системы для решения задачи организации распределенных вычислений. На каждом динамически формируемом наборе вычислительных узлов может выполняться вычислительное задание. Такое задание определяется именем программы (исполняемого файла) и аргументами командной строки. Результатом его выполнения является стандартный вывод программы. На каждом узле запущен компонент целевой программной системы в виде отдельного Erlang/OTP-приложения. Подобное приложение предоставляет возможность запустить процесс, под управлением которого задание будет выполнено, а его результат будет доставлен потребителю. Потребитель имеет возможность получить текущий список узлов, на которых могут выполняться его вычислительные задания.

Ключевые слова: *распределенные вычисления, Erlang/OTP*

A. S. Shundeev

Distributed Computing Systems Based on the Platform Erlang/OTP

We consider the following problem statement distributed computing. There is a dynamically generated set of computing nodes. At each node, such a computing task may be performed. Computing task is determined by the name of the program and command line arguments. The result of the computational tasks is the standard output of the program. On each compute node running the target component of a software system as a separate Erlang/OTP application. Such an application provides the ability to start a process, which manages the computing task will be executed and its result will be delivered to the consumer. The consumer has the opportunity to get a current list of nodes on which it can run computational tasks.

Keywords: *distributed computing, Erlang/OTP.*

Введение

Распределенные вычисления представляют собой область информационных технологий, которая активно развивается с конца 1980-х гг. [1]. С конца 1990-х гг. распределенные вычисления тесно связаны с грид-технологиями [2].

В основе распределенных вычислений лежит понятие *пакетной программы*, а также вычислительной системы, допускающей выполнение программ в пакетном режиме. Под пакетной программой подразумевается программа, не требующая во время своего выполнения интерактивного взаимодействия с пользователем. В качестве вычислительной системы может выступать как обычная рабочая станция, так и кластерные, в том числе суперкомпьютерные системы. Последнее более характерно для грид-систем.

Более абстрактным по отношению к понятию пакетной программы является понятие *вычислительного задания*. В простейшем случае вычислительное задание определяется пакетной программой и требованиями к вычислительной системе, на базе которой пакетная программа может выполняться. В более сложных случаях вычислительное задание включает в себя перечень пар "пакетная программа — требования к вычислительной системе". Подобный перечень определяет альтернативные способы выполнения вычислительного задания. Например, в зависимости от доступных в данный момент вычислительных систем, одно и то же вычислительное задание может быть выполнено либо как последовательная программа на обычной рабочей станции, либо как параллельная MPI-программа в кластерной системе.

Используют и более сложные типы вычислительных заданий. К их числу можно отнести *композитные задания*. Композитные задания представляют собой набор обычных вычислительных заданий, а также набор правил, задающий порядок их выполнения. Как правило, подобные правила задают в виде ациклических графов [1].

Программная система, организующая распределенные вычисления, обрабатывает входной поток вычислительных заданий. В упрощенном виде обработка включает решение взаимосвязанных задач планирования и контроля. Планирование заключается в выборе для каждого вычислительного задания "подходящей" вычислительной системы для его выполнения. Контроль реализуется путем постоянного мониторинга хода выполнения вычислительного задания, а также разрешения нештатных ситуаций.

Следует обратить внимание на те условия, в которых зарождались и отрабатывались базовые решения технологии распределенных вычислений. Эти условия характеризуются большим дефицитом (по сравнению с состоянием на сегодня) вычислительных ресурсов. Сопутствующая бизнес-модель включает три типа субъектов. К первому типу относятся клиенты, заинтересованные в решении своих вычислительных задач. Ко второму типу относятся владельцы вычислительных систем, а к третьему — посредник, непосредственно организующий распределенные вычисления.

В рамках данной бизнес-модели предполагается выстраивание достаточно сложных схем реализации отношений взаимного доверия между потенциальными клиентами и владельцами вычислительных ресурсов. Например, владелец рабочей станции может выдвинуть следующие условия, которые должны неукоснительно соблюдаться: рабочую станцию можно использовать только в ночное время, при этом размер используемой памяти жесткого диска не должен превышать заданного лимита.

Учитывая большое число владельцев вычислительных ресурсов, их переменный состав, а также неоднородность самих вычислительных ресурсов, для задач планирования и контроля пришлось реализовать достаточно сложные технические решения [1].

В настоящее время ситуация существенно изменилась. Учитывая развитие облачных технологий (доступность соответствующих сервисов), от модели "клиент—посредник—владелец" в ряде случаев можно отказаться. По всей видимости, данная модель останется актуальной только для крупномасштабных грид-систем.

Распределенная пакетная обработка данных остается востребованным архитектурным шаблоном, который используется при создании целевых информационных систем. В связи с этим обстоятельством актуальной является задача построения облегченных (не базирующихся на бизнес-модели "клиент—посредник—владелец") и адаптированных под облачные инфраструктуры программных систем организации распределенных вычислений. В данной работе представлено описание такой программной системы.

Платформа Erlang/OTP

Кратко опишем основные механизмы языка программирования Erlang [3] и базирующейся на нем платформы OTP (*Open Telecom Platform*), которые были использованы при создании системы распределенных вычислений.

Одним из базовых элементов языка Erlang является *процесс*. Подобные процессы выполняются параллельно и независимо друг от друга. Процессы языка Erlang являются легковесными в том смысле, что на их порождение и выполнение требуется значительно меньшее количество ресурсов по сравнению с традиционными процессами и потоками операционной системы. Процессы взаимодействуют друг с другом при помощи обмена сообщениями.

Программы на языке Erlang компилируются в байт-код, которой выполняется виртуальной машиной. Виртуальным машинам могут быть присвоены имена. В этом случае они становятся *узлами* (*node*). Подобные узлы могут взаимодействовать друг с другом. Процессы, запущенные на разных узлах, могут обмениваться сообщениями. Использование распределенных узлов позволяет осуществлять горизонтальное масштабирование целевых программных систем, реализованных на базе языка Erlang. Вертикальное масштабирование достигается при использовании в рамках одного узла параллельно запущенных процессов, реализующих обработку данных.

В соответствии с архитектурными принципами OTP все процессы делят на два типа — *рабочие* процессы и *супервизоры*. Рабочие процессы отвечают за непосредственную обработку данных, а супервизоры отвечают за запуск (в том числе повторный) других процессов. Приложение на основе OTP может быть представлено в виде дерева, внутренним вершинам которого приписаны супервизоры, а листовым вершинам — рабочие процессы. Каждый супервизор в таком дереве управляет только своими непосредственными потомками.

Существует несколько способов организации взаимодействия Erlang-программы с внешними системами. Одним из способов является использование механизма *портов* (*ports*). Через порт виртуальная машина Erlang может запустить внешний процесс операционной системы. Взаимодействие с ним осуществляется через его стандартный ввод/вывод. На базе механизма портов реализуется стандартная функция *os:cmd*. Эта функция на вход получает строку, содержащую команду командного интерпретатора операционной системы. Результатом функции является строка, содержащая стандартный вывод выполнения данной команды.

Архитектура

Программная система распределенных вычислений включает три типа OTP-приложений, а именно *Job*, *Reg* и *Web*. Приложения типа *Job* отвечают за непосредственное выполнение вычислительных заданий. На каждом доступном вычислительном хосте может быть запущено приложение этого типа.

Корневой супервизор приложения *Job* имеет двух потомков. Первый потомок представляет собой рабочий процесс *job_stat_gen*, отвечающий за сбор статистики. Второй потомок — супервизор *job_sup*. Через этот супервизор запускаются рабочие процессы *job_gen*, отвечающие за непосредственное выполнение вычислительных заданий.

Реализация процесса типа *job_gen* базируется на использовании стандартной функции *os:cmd*. Таким образом, вычислительное задание задается командой командного интерпретатора операционной системы.

В большинстве случаев это строка, содержащая имя исполняемого файла и аргументы командной строки. Результатом выполнения вычислительного задания является стандартный вывод выполненной команды. При запуске на выполнение вычислительного задания соответствующий процесс *job_gen* посылает информационное сообщение процессу *job_stat_gen*. Информационные сообщения также посылаются при завершении (успешном и не успешном) выполнения вычислительного задания. Таким образом, в состоянии процесса *job_stat_gen* формируется актуальная статистика о выполняющихся и ранее выполненных заданиях.

При запуске и штатной остановке приложения типа *Job* посылают информационные сообщения приложению *Reg*. Таким образом осуществляется централизованная регистрация (удаление из списка зарегистрированных) приложений типа *Job*. Кроме того, приложение *Reg* периодически, по таймауту запрашивает статистику у каждого зарегистрированного приложения *Job*. Данная информация является основой для решения задачи планирования.

Приложение типа *Web* предоставляет веб-интерфейс для взаимодействия конечных пользователей с системой распределенных вычислений. Отличительной чертой приложения *Web* является возможность моделирования композитных заданий.

Композитные задания

Как видно из названия, композитное задание представляет собой набор обычных вычислительных заданий, а также правила, задающие порядок их выполнения. Выделяют два задания, которые обозначают как *pre* и *post*. Остальные задания называют *регулярными*. Данные вычислительные задания определяют исполняемые файлы и аргументы командной строки. Отметим, что у регулярных заданий исполняемый файл один и тот же. Варьироваться могут только аргументы командной строки.

Выполнение композитного задания всегда начинается с выполнения *pre*-задания. Первый передаваемый аргумент командной строки интерпретируется как число регулярных заданий в составе композитного задания. Результат выполнения *pre*-задания (его стандартный вывод) интерпретируется следующим образом. Это строка значений, разделенных пробелами. Сначала идут вычисленные аргументы командной строки для первого регулярного задания, затем для второго регулярного задания и т. д.

После выполнения *pre*-задания осуществляется параллельный запуск регулярных заданий. После завершения выполнения всех регулярных заданий запускается *post*-задание. Результаты выполнения регулярных заданий формируют значения аргументов командной строки *post*-задания. В случае успешного выполнения всех составных вычислительных заданий (соответствующие процессы операционной системы завершаются с кодом 0) стандартный вывод *post*-задания интерпретируется как результат успешного выполнения композитного задания. В противном случае считают, что композитное задание выполнилось с ошибкой.

Рассмотрим пример композитного задания, осуществляющего распараллеливание процесса интегрирования функции.

Предположим, что исполняемый файл *integrate.py* осуществляет интегрирование некоторой функции. При запуске данного исполняемого файла ему передают три аргумента — границы интегрирования и число точек разбиения. Например, запуск *./integrate.py* 1.0 2.0 100 означает, что интегрирование будет осуществляться на отрезке [1.0, 2.0] со 100 точками разбиения. Исполняемый файл *integrate.py* соответствует регулярным вычислительным заданиям.

Вычислительному заданию типа *pre* соответствует исполняемый файл *split.py*. При запуске данного исполняемого файла ему передают четыре аргумента — число параллельно выполняемых регулярных заданий, границы интегрирования и число точек разбиения. Например, результатом запуска *./split.py* 2 1.0 2.0 100 может служить строка вида 1.0 1.5 50 1.5 2.0 50. Данная строка интерпретируется следующим образом. После выполнения *pre*-задания будут параллельно запущены два регулярных задания. Запуск первого регулярного задания будет иметь вид *./integrate.py* 1.0 1.5 50, а второго — *./integrate.py* 1.5 2.0 50.

Вычислительному заданию типа *post* соответствует исполняемый файл *sum.py*, который интерпретирует свои аргументы командной строки как числа, суммирует эти числа и печатает эту сумму в стандартный поток вывода.

Заключение

В данной статье представлено краткое описание архитектуры программной системы организации распределенных вычислений в виде OTP-приложения. Необходимо отметить, что платформа OTP в данном случае является всего лишь инструментом, хотя и очень удобным. Однако, как это было отмечено во введении, основная задача состояла в построении программной системы организации распределенных вычислений, не базирующейся на бизнес-модели "клиент—посредник—владелец".

Построенная система удовлетворяет основным требованиям, выдвигаемым к подобным системам, к числу которых относятся горизонтальная и вертикальная масштабируемость, отказоустойчивость, возможность выполнения как одиночных, так и композитных заданий. Горизонтальная масштабируемость достигается за счет увеличения числа запущенных приложений типа *Job* на доступных вычислительных сетевых хостах. Вертикальная масштабируемость достигается за счет использования параллельных Erlang-процессов. Организация OTP- приложения в виде иерархии супервизоров и рабочих процессов обеспечивает высокий уровень отказоустойчивости системы.

Список литературы

1. **Thain D., Tannenbaum T., Livny M.** Distributed Computing in Practice: the Condor Experience // Concurrence and Computation: Practice and Experience 2005. V. 17, N. 2—4. P. 323—356.

2. **Васенин В. А., Шундеев А. С.** Эволюция технологии грид // Информационные технологии. 2012. № 1. С. 2—9.

3. **Laurent S.** Introducing Erlang. Sebastopol: O'Reilly Media, 2013. 204 p.

УДК 004.04

В. М. Баканов, д-р техн. наук, проф., e-mail: vbakanov@hse.ru, **В. В. Палагин**, аспирант, НИУ ВШЭ, г. Москва

Повышение эффективности параллельных программ по времени их выполнения за счет рационального размещения данных в распределенной памяти вычислителя

Обсуждаются вопросы эффективного распределения данных по локальным ресурсам (оперативной памяти) многопроцессорных вычислительных систем архитектуры MPP в целях минимизации простоев вычислителей во время обмена данными между узлами многопроцессорных вычислительных систем. Предлагается формальная модель процесса, реализующего такое распределение, на уровне исходных текстов программ и методика оптимизации.

Ключевые слова: *параллельное программирование, архитектура MPP, распределение данных по локальным ресурсам памяти вычислительных узлов, выбор оптимального (рационального) распределения*

V. M. Bakanov, V. V. Palagin

Improving the Efficiency of Parallel Programs by the Time They Run through Judicious Placement of Data in a Distributed Memory Calculator

This paper discusses the issues of effective distribution of data across local ethyl resources (RAM) Multiprocessor Computing Systems (MVS) MPP architecture to minimize downtime calculators during communication between nodes MVS. The formal model of the process of implementing such a distribution, source-level programs and the technique, optimization.

Keywords: *parallel programming, architecture MPP, the distribution of data across local storage resources of compute nodes, selects the optimal (rational) distribution*

Решение многих стратегически важных задач науки и техники, промышленности и других сфер экономики требует использования суперкомпьютерных систем с соответствующим математическим и программным обеспечением. По данным ресурса TOP 500 [1] наиболее высокопроизводительными из них являются (вследствие повышенной по сравнению с конкурентами масштабируемостью) многопроцессорные вычислительные системы (МВС) архитектуры MPP (*Massively Parallel Processing*) и вычислительные кластеры. Оба класса МВС характеризуются наличием собственной (локальной) оперативной памяти на каждом вычислительном узле (ВУ) и развитыми аппаратно-программными средствами обмена данными между различными ВУ [2]. Одним из проблемных вопросов, замедляющих процесс вычислений на таких системах, получивший название "стена памяти", является запаздывание доставки необходимых для обработки данных от ВУ-хранителю данных к ВУ-обработчику. Последний при этом простаивает в ожидании. Как следствие, задача рационального (минимизирующего время обмена данными) распределения исходных данных по ВУ актуальна. Ее решение позволит снизить время выполнения программы без изменения аппаратной части МВС.

Известным (для программ общего назначения) является метод максимального увеличения размера и, как следствие, времени выполнения гранул параллелизма (участков программы, выполняющихся без обмена данными с иными участками). Относительная доля временных интервалов обмена по сравнению со временем собственно счета снижается [3]. Однако для программ, реализующих сеточные методы, описанный поход малоперспективен, так как гранулы малы по определению и являются телами циклов по индексированным переменным. В задачах моделирования в механике, физике и т. п. сеточные методы (в первую очередь метод конечных разностей) используют очень часто, поэтому решение поставленной задачи очень важно именно для подобного типа приложений.

Инструментарий современных систем параллельного программирования предполагает наличие директив распределения исходных данных по ВУ (под ВУ будем понимать вычислительный узел МВС, обладающий собственной оперативной памятью и одним/несколькими процессорными элементами). В некоторых системах параллельного программирования указанное распределение может быть явно задано языковыми средствами (например, директивы DISTRIBUTE в DVM, DISTRIBUTION INDEX в НОРМА [4]), в других системах (например, в PVM, MPI [2]) процесс полностью перекладывается на программиста. Правила распределения имеют варианты (в DVM их три [2]), наиболее простым является описание, вдоль какого из индексных направлений осуществляется разбиение массива на части (полосы или прямоугольники для двумерных матриц). Данное распределение действует для некоторой программной области и в ее пределах не изменяется.

Назовем *блоком* часть программы, для которой заданное распределение является неизменным. *Гнездом циклов* (*cycle nest*) будем называть структуру вложенных до некоторого уровня циклов. Блок при этом может включать любое число гнезд циклов (ГЦ). Предпосылкой для рассмотрения нескольких ГЦ в рамках блока является зависимость по данным, возникающая в случае обработки пересекающихся массивов данных $Mass = Mass_{in} \cup Mass_{out}$, где $Mass_{in}$ — множество исходных переменных; $Mass_{out}$ — множество переменных после прохождения ГЦ. Через ГЦ j_a (если $a < b$, то вложенный цикл j_a встречается в исходном коде программы раньше, чем цикл j_b) будем обозначать множество переменных в массиве до выполнения цикла: $In(j_a) = Mass_{in}$, множество переменных в массиве после прохождения всех итераций цикла будем обозначать $Out(j_a) = Mass_{out}$.

Таким образом, при получении доступа к одному и тому же массиву переменных между двумя циклами возникает два типа взаимоотношений:

- зависимость по чтению $dep_{in,\ in}(j_b, j_a) = \{In(j_b) \cap \cap\ In(j_a)\}$, при которой ГЦ не изменяют данные в массиве, но используют их в процессе вычислений;
- зависимость по записи $dep_{out,\ in}(j_b, j_a) = \{In(j_b) \cap \cap\ Out(j_a)\}$, где $a < b$ и ГЦ j_b работает с данными, измененными (перезаписанными) предыдущим циклом j_a.

Формальное описание такого подхода представлено на рис. 1.

На рис. 1 приведены два способа задания распределения частей массивов по ВУ. Вариант DISTRIBUTE $i = 1, ..., N$ означает, что все массивы, содержащие индексное направление i, будут распределены полосами (для одномерных массивов) вдоль этого индекса на N ВУ (такой вид директивы распределения применяется в системе НОРМА). Вариант DISTRIBUTE MASSIVE_NAME FORMAT позволяет для конкретного массива MASSIVE_NAME задать распределение, которое определяется квалификатором FORMAT (характерно для системы программирования DVM).

Термин "распределение содержащих индексное направление i массивов по этому индексу" означает, что при распределении исходного массива по отдельным ВУ массив "разрезается" на ленты вдоль заданного направления. Например, массив $C[i = 0, ..., i_1][j = 0, ..., j_1]$ при этом будет распределен на линейку из трех ВУ (с обычными для таких случаев правилами округления до целого). Соответственно при распределении на 2D-решетку ВУ в оперативной памяти (ОП) каждого из них будет присутствовать не лента, а прямоугольный подблок каждого массива.

При локальных вычислениях на ВУ доступ к элементам массивов для каждого тела ГЦ определяется числом операций индексирования и правилами расположения элементов в ОП, принятыми для конкретного языка программирования (в языке Fortran элементы массива расположены в ОП последовательно по столбцам, в языке С — по строкам). С точки зрения временных затрат на обмен данными между ВУ представленная выше схема оптимальна по критерию времени выполнения при более частом изменении индекса j, чем при изменении индекса i (меньшее число пересылок элементов массива между ВУ). Исходя из этих соображений можно предсказать, для каких из входящих в блок гнезд заданное распределение будет рациональным, а для каких не будет. Общее время выполнения блока представляет собой суперпозицию временных интервалов выполнения операций всеми гнездами блока и является целевой функцией, подлежащей минимизации. Заметим, что данная формализация не учитывает кэширование на каждом ВУ. Подробный учет этого явления сильно усложняет описание процесса.

Распределение данных (для первого варианта распределения) представим в виде списка (множества реализаций) $D_K = \{d_1, d_2, d_3, ..., d_k, ..., d_K\}$, где d_k — элемент списка D_K, описывающий конкретное распределение, например, $d_k = \{i_1, i_2, i_3\}$, где i_1, i_2, i_3 — индексные направления в массивах, используемые при их "разрезании" для распределения на различные ВУ. Например, при $K = 1$, где K — общее число вариантов распределения, имеем $D_K = \{i_1\}$ — распределе-

Распределение данных по процессорам в блоке	1. DISTRIBUTE massive_name format 2. DISTRIBUTE INDEX *i*=1..*N*	
Начало блока *(beginning of code block)*		
	Mass_A[*N,N*] Mass_X[*N,N*]	Исходные массивы данных для распределения
	DO $i=i_1, i_2, i_3$ BODY $_{1,i}$ DO $j=j_1, j_2, j_3$ BODY $_{1,j}$ DO $k=k_1, k_2, k_3$ BODY $_{1,k}$ END *k* END *j* END *i*	Начало гнезда циклов #1 глубиной 3 *(beginning of cycle nest)* Конец гнезда циклов #1
	DO $i=i_1, i_2, i_3$ BODY $_{2,i}$ DO $j=j_1, j_2, j_3$ BODY $_{2,j}$ END *j* END *i*	Начало гнезда циклов #2 глубиной 2 *(beginning of cycle nest)* Конец гнезда циклов #2
Конец блока		

Рис. 1. Вид программного блока при формальном описании процесса рационального распределения массивов данных по ВУ МВС архитектуры MPP

ние на топологию "линейка" вдоль индексного направления i_1; при $K = 2$ имеем $D_K = \{\{i_1, i_2\}, \{i_2, i_1\}\}$ — распределение на топологию "2D-решетка" вдоль индексных направлений i_1 и далее вдоль i_2 и i_2 и далее вдоль i_1; причем $\{i_1, i_2\} \neq \{i_2, i_1\}$.

Блок опишем как совокупность ГЦ. Блок представим списком $F_M = \{f_1, f_2, f_3, ..., f_m, ..., f_M\}$, где f_m — элемент списка F_M, описывающий гнездо циклов m; M — общее число ГЦ в блоке. Описание гнезда включает последовательный список индексов (начиная изнутри гнезда, например, для блока на рис. 1 это будет $\{k, j, i\}$, $\{j, i\}$ при $M = 2$) и дополнительные данные, учитывающие диапазон и шаг изменения каждого индекса, а также параметры, характеризующие трудоемкость каждого тела цикла (тела циклов на каждом уровне гнезд представлены как $\text{BODY}_{m,i}$, где m — номер гнезда в блоке, i характеризует определенный цикл в данном гнезде). Трудоемкость каждого тела гнезда определяется числом арифметических и индексных операций в теле цикла.

При заданных обозначениях схема алгоритма поиска рационального распределения (с точки зрения минимума потерь времени на обмен данными между отдельными ВС) будет соответствовать схеме, изображенной на рис. 2.

Принципиально возможны три перечисленных далее подхода к варьированию входных параметров D и

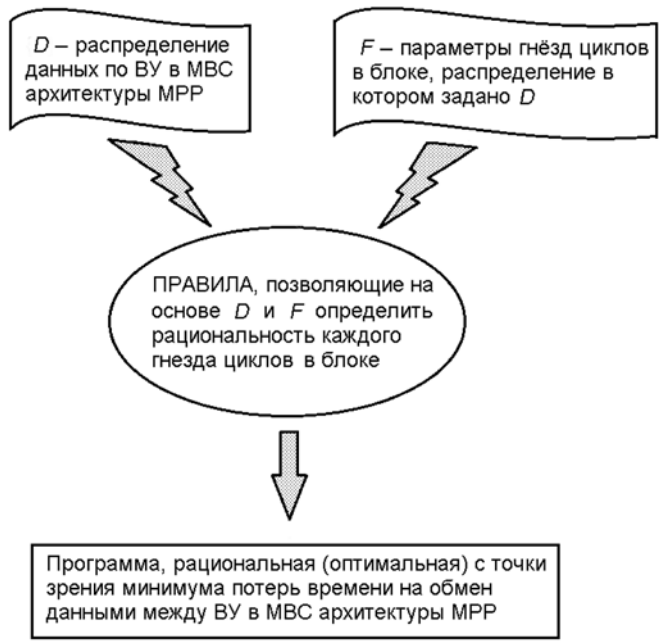


Рис. 2. Общая схема поиска оптимального с точки зрения минимума потерь времени на обмен данными между ВУ распределения

F (*var* и *const* обозначают действия изменения и постоянства соответственно).

1) $D = var$, $F = const$;
2) $D = const$, $F = var$;
3) $D = var$, $F = var$.

Первый подход соответствует выбору распределения, минимизирующего время выполнения блока, к которому оно применяется без модификации структуры ГЦ (этот подход предлагается использовать в данной работе). Второй подход предполагает эквивалентные преобразования ГЦ в блоке при неизменном параметре распределения. Такой подход используют в оптимизирующих и распараллеливающих системах, таких как BERT 77, FORGExplorer, V-Ray, OPC. При реализации третьего подхода изменению подвергают как распределение, так и блок, что является наиболее сложным вариантом оптимизационной задачи.

Так как в каждом теле циклов в блоке выполняются как операции индексации при обращении к элементам массивов, так и чисто вычислительные операции, задача оптимизации должна применять балансировочный подход к анализу трудоемкостей выполнения каждого ГЦ. Для информации о трудоемкости гнезда дополним каждый элемент F_M данными в виде подсписка N_{calc}, N_{mem}, T_{calc}, T_{mem} — число арифметических операций, число операций индексирования, время выполнения арифметической операции и время выполнения операции индексирования соответственно для цикла глубины l ($l = 1, ..., l_M$). Тогда время однократного выполнения тела цикла номер l будет равно

$$t_{m,\,l} = N_{calc} \times T_{calc} + N_{mem} \times T_{mem},$$

а полное время выполнения цикла глубиной l гнезда m определится как

$$T_{m,\,l} = t_{m,\,l}\,(i_2 - i_1)/i_3,$$

где i_1, i_2, i_3 — конечное значение, начальное значение и приращение индексной переменной в цикле соответственно; случаи неполного выполнения циклов при использовании операторов условного перехода требуют дополнительного уточнения.

Время $T_{nest,\,m}$ выполнения всего m-го ГЦ равно

$$T_{nest,\,m} = \sum_{s=1}^{s=l_m}\left[\prod_{n=s}^{n=l_m}\left(T_n\frac{i_{2,\,n} - i_{1,\,n}}{i_{3,\,n}}\right)\right], \qquad (1)$$

где T_n — время выполнения тела цикла на глубине n (s, $n = 1, ..., l$, нумерация от наиболее глубокого к внешнему циклу гнезда); $i_{2,\,n}$, $i_{1,\,n}$, $i_{3,\,n}$ — начальное значение, конечное значение и приращение индексной переменной в цикле глубиной n соответственно (для упрощения индексы m и l опущены).

Время T_{block} выполнения блока из m гнезд определится суперпозицией времен выполнения всех входящих в блок ГЦ:

$$T_{block} = \sum_{m=1}^{m=M} T_{nest,\,m}. \qquad (2)$$

Анализ приведенных выражений затрудняется сложностью количественного определения T_{calc} и T_{mem}, зависящих от конкретной аппаратной платформы МВС. Однако из вида представленных выражений ясно, что рост и T_{calc}, и T_{mem} приводит к увеличению времени выполнения программы. В целях достижения аппаратной независимости при поиске экстремумов (минимизации) выражений (1) и (2) возможно применять относительное время выполнения операций (например, $T_{calc} = 1$ ед. времени). Однако распределение исходных данных влияет на время T_{mem} доступа к ним (можно принять, например, $T_{mem} = 1$ для некоторого базового случая и варьировать T_{mem} в зависимости от заданного распределения). Имея возможность определить T_{mem} для каждой операции доступа к данным по индексу (зависит от D при фиксированном F, т. е. $T_{mem} = f(d_1, d_2, ..., d_K)$, получим функцию цели для поставленной задачи:

$$T_{block} = f_B(d_1, d_2, ..., d_K) \to \min, \qquad (3)$$

где каждое d_K задает распределение исходных данных по ВУ в пределах блока.

Операция нахождения минимума (3) упрощается, так как D является конечным множеством (причем обычно небольшого размера). В этом случае вполне возможно применение метода простого перебора для минимизации T_{block}. Однако в том случае, если блок содержит более одного ГЦ, модификация задания распределения данных повлияет на время выполнения каждого целевого ГЦ в блоке (рис. 3).

В некоторых случаях задача может быть упрощена с использованием принципа выбора и оптимизации наиболее "весомых" в плане трудоемкости гнезд из общего блока. В этом случае определяем долю (вес) каждого гнезда в блоке согласно формуле

$$P_m = T_{nest,\,m}/T_{block}.$$

Затем проводим выбор рационального D только для ГЦ номер $m = p$ с максимальным весом:

$$T_{nest,\,p} = f_N(d_1, d_2, ..., d_K) \to \min. \qquad (4)$$

Технической реализацией (4) является разработка программного модуля-препроцессора для анализа и изменения исходного кода параллельной программы.

Эффективность применения формул (1) или (2) определяется правильностью оценки каждого T_{mem} для заданного D. Ключевым моментом здесь является система правил, на основе которых можно априорно судить о благоприятности целевого варианта распределения исходных данных по ВУ D при заданных параметрах блока F, к которому оно применяется.

Система правил представляет собой набор шаблонов выполнения типовых операций над матрицами/векторами (матрица — двухмерный массив, вектор — одномерный массив), которые наиболее часто используют в параллельных программах для научно-технических расчетов. Под типовыми операциями подразумевают: умножение/сложение матриц/векто-

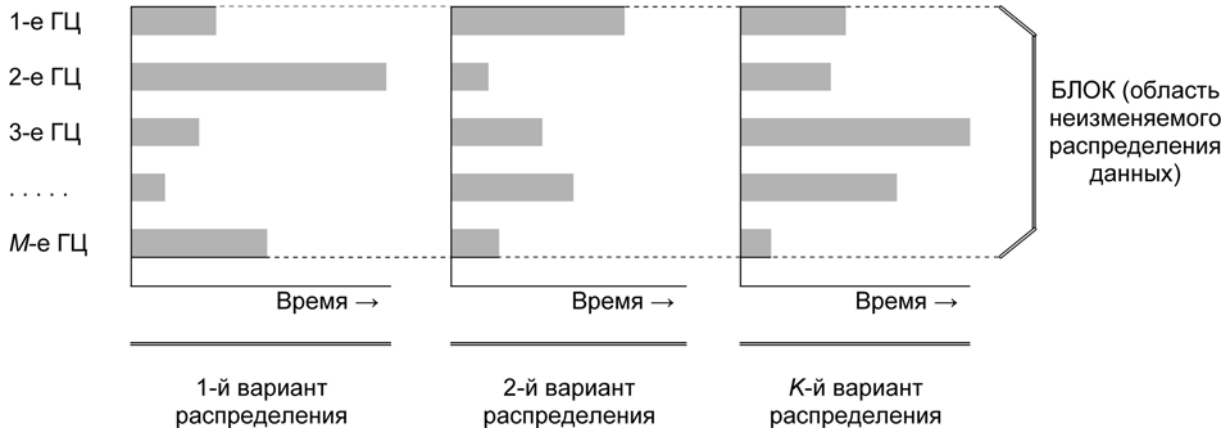


Рис. 3. Влияние распределения данных на время выполнения ГЦ в блоке

Рис. 4. Формирование системы правил на основе практических экспериментов

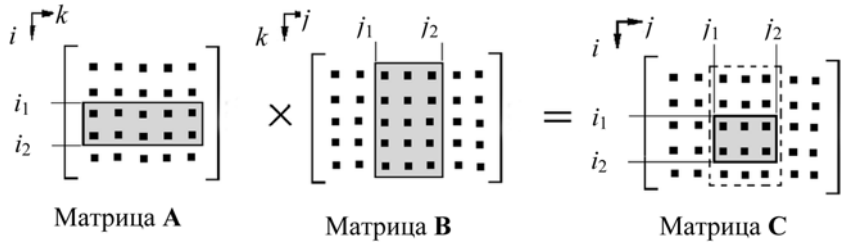


Рис. 5. Схема индексации массивов при параллельном умножении матриц ленточным методом

ров; поиск максимума/минимума в матрице/векторе; умножение/сложение элементов матрицы/вектора; транспонирование матрицы; вычисление обратной матрицы; решение СЛАУ. Каждая типовая операция характеризуется структурой построения ГЦ — глубиной вложенности, порядком следования индексов, набором арифметических операций в теле цикла.

В свою очередь шаблон выполнения представляет собой значение рационального распределения данных для каждой типовой операции, полученное из практических экспериментов с использованием вычислительных ресурсов кластерной системы МГУПИ и суперкомпьютерного комплекса МГУ им. М. В. Ломоносова [6] (рис. 4).

Для каждой тестовой параллельной программы, реализующей одну типовую операцию в рамках блока, опытным путем было получено время выполнения в зависимости от выбранного параметра распределения данных.

Исходя из представленных соображений, проанализируем несложную операцию умножения 2D-матриц $\mathbf{C} \leftarrow \mathbf{A} \times \mathbf{B}$ порядка N, записанную в классическом виде на С-подобном языке (обнуление матрицы-результата опущено):

$$\text{for}(i = 0;\ i < N;\ i{+}{+})$$
$$\text{for}(j = 0;\ j < N;\ j{+}{+})$$
$$\text{for}(k = 0;\ k < N;\ k{+}{+})$$
$$C[i][j] += A[i][k] \times B[k][j].$$

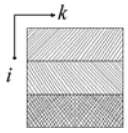
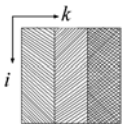
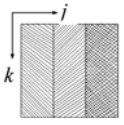
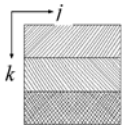
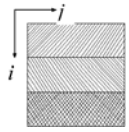
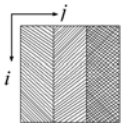
Для этого алгоритма (ленточный метод умножения при распараллеливании) индексация исходных и результирующей матриц соответствует схеме, представленной на рис. 5.

Анализ операции умножения матриц в локальной памяти каждого ВУ на основе схемы размещения обрабатываемых массивов в распределенной памяти МВС (с учетом предпочтений для языков программирования Fortran и C) приведен в табл. 1.

Для программ с организацией вычислений, при которой процессы разделены и запускаются параллельно на нескольких ВУ, работая каждый в своей локальной памяти, общее время выполнения складывается из времени обмена сообщениями по коммуникационной сети T_{net} и времени выполнения процессов локально T_{local}. Соответственно будет определяться

Таблица 1

Анализ эффективности доступа к данным в локальной памяти ВУ при умножении матриц с точки зрения возможностей кэширования

Индексируемый элемент	Распределение								
	Вдоль индекса *i*			Вдоль индекса *j*			Вдоль индекса *k*		
	Схема	Fortran	C	Схема	Fortran	C	Схема	Fortran	C
A[*i*][*k*] (чтение)		–	+	Не определено				+	–
B[*k*][*j*] (чтение)	Не определено				+	–		–	+
C[*i*][*j*] (чтение + запись)		–	+		+	–	Не определено		
Суммарный эффект (Fortran/C)	−2/+2			+2/−2			0/0		

Примечание: знаками +/– отмечено положительное/отрицательное влияние параметра распределения на время доступа к данным в целевой параллельной программе. Направлением штриховки обозначены части массивов, находящихся в ОП разных ВУ. Суммарный эффект определён как сумма положительного и отрицательного влияния распределения на время доступа к элементам массивов.

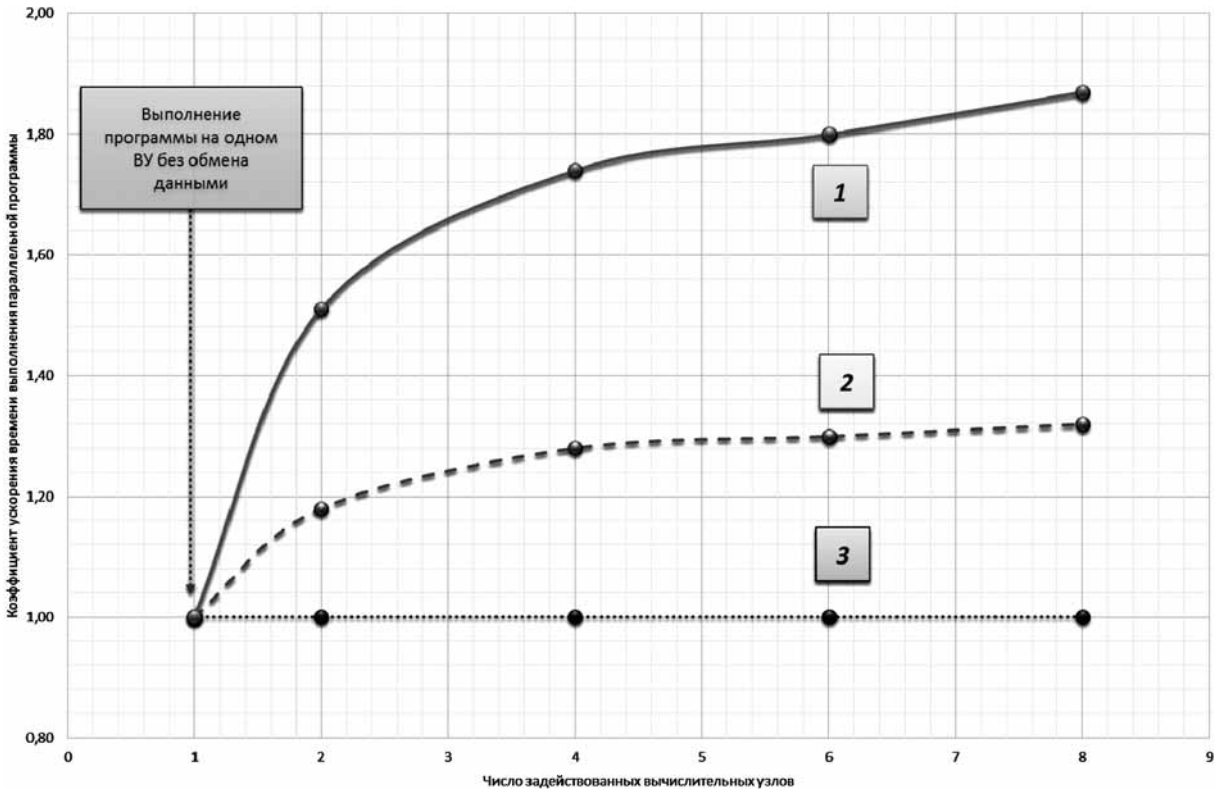
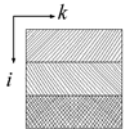
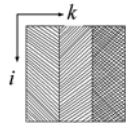
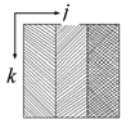
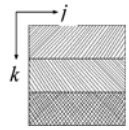
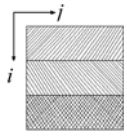
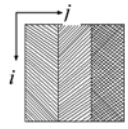


Рис. 6. Влияние обмена данными между ВУ на время выполнения параллельной программы

Таблица 2

Анализ эффективности доступа к данным при обмене сообщениями в процессе вычислений на ВУ

Индексируемый элемент	Распределение					
	Вдоль индекса *i*		Вдоль индекса *j*		Вдоль индекса *k*	
	Схема	Результат	Схема	Результат	Схема	Результат
A[*i*][*k*] (чтение)		по *i* (−1) по *k* (+1)	Пересылка массива целиком			по *i* (+1) по *k* (-1)
B[*k*][*j*] (чтение)	Пересылка массива целиком			по *j* (−1) по *k* (+1)		по *j* (+1) по *k* (-1)
C[*i*][*j*] (чтение + запись)		по *i* (−2) по *j* (+2)		по *i* (+2) по *j* (-2)	Не определено	
Суммарный эффект	по *i* (−3) по *j* (+2) по *k* (+1)		по *i* (+2) по *j* (−3) по *k* (+1)		по *i* (+1) по *j* (+1) по *k* (−2)	

Примечание: обозначение (+1) используется при совпадении направления индекса с большей длиной полосы разбиения (при изменении данного индекса нет перехода от полосы к полосе, что подразумевает подкачку данных с другого ВУ); обозначение (−1) используется при несовпадении направления индекса (соответственно для массивов *A* и *B* проводится единичное обращение к ОП ВУ, в то время как для массива *C* — двойное обращение). Направлением штриховки обозначены части массивов, находящиеся в ОП разных ВУ. Суммарный эффект определяется как сумма положительных и отрицательных влияний распределения данных на время выполнения при доступе к элементам массивов.

эффективность целевого значения распределения данных, в том числе классом решаемых задач. Для параллельных программ, у которых $T_{net} \gg T_{local}$, предложенный метод даст ощутимый эффект.

Характерным примером здесь являются реализации различных операций над массивами большой размерности, в том числе решение СЛАУ, при незначительной трудоемкости расчетных формул в ГЦ. На рис. 6 показано ускорение (снижение времени выполнения по сравнению с последовательным выполнением) параллельной программы умножения квадратных матриц порядка 10^3 элементов (числа двойной точности DOUBLE PRECISION) в зависимости от значения распределения данных и числа задействованных ВУ (аппаратная платформа МГУПИ). При задействовании одного ВУ выполняется только начальная инициализация обрабатываемых данных, тем самым становится возможным исключить влияние обмена данными между ВУ на общее время выполнения. Как видно на графике, для одного ВУ время выполнения для всех трех вариантов распределения в пределах погрешности, в то время как при задействовании уже двух ВУ время выполнения будет значительно отличаться (кривая *1* — ускорение 1,17, кривая *2* — ускорение 1,5, кривая *3* — без ускорения, наихудший вариант).

Для данной параллельной программы время локального выполнения незначительно по сравнению со временем обмена данными, которое составляет основную долю от общего времени выполнения.

В том случае, если рассматривать эффективность доступа к данным на примере процедуры умножения матриц с точки зрения минимума обращений к данным из ОП другого ВУ по изменяющимся индексам *i*, *j*, *k* в теле цикла (без учета возможного кэширования элементов массивов на ВУ), то общая картина будет иной (табл. 2).

На рис. 7 показана зависимость ускорения параллельной программы, реализующей решение СЛАУ методом Гаусса—Жордана, от значения распределения данных и числа задействованных ВУ (аппаратная платформа суперкомпьютерного комплекса МГУ им. М. В. Ломоносова [5]). Как видно на графике, время выполнения будет значительно отличаться (группа *1* — ускорение 3,5—5,1 раза; группа *2* — 1,9—2,2 раза; группа *3* — без ускорения, наихудший вариант, принятый за точку отсчета).

На рис. 7 цифрой *1* показан пример двух вариантов распределения, иллюстрирующий максимальное ус-

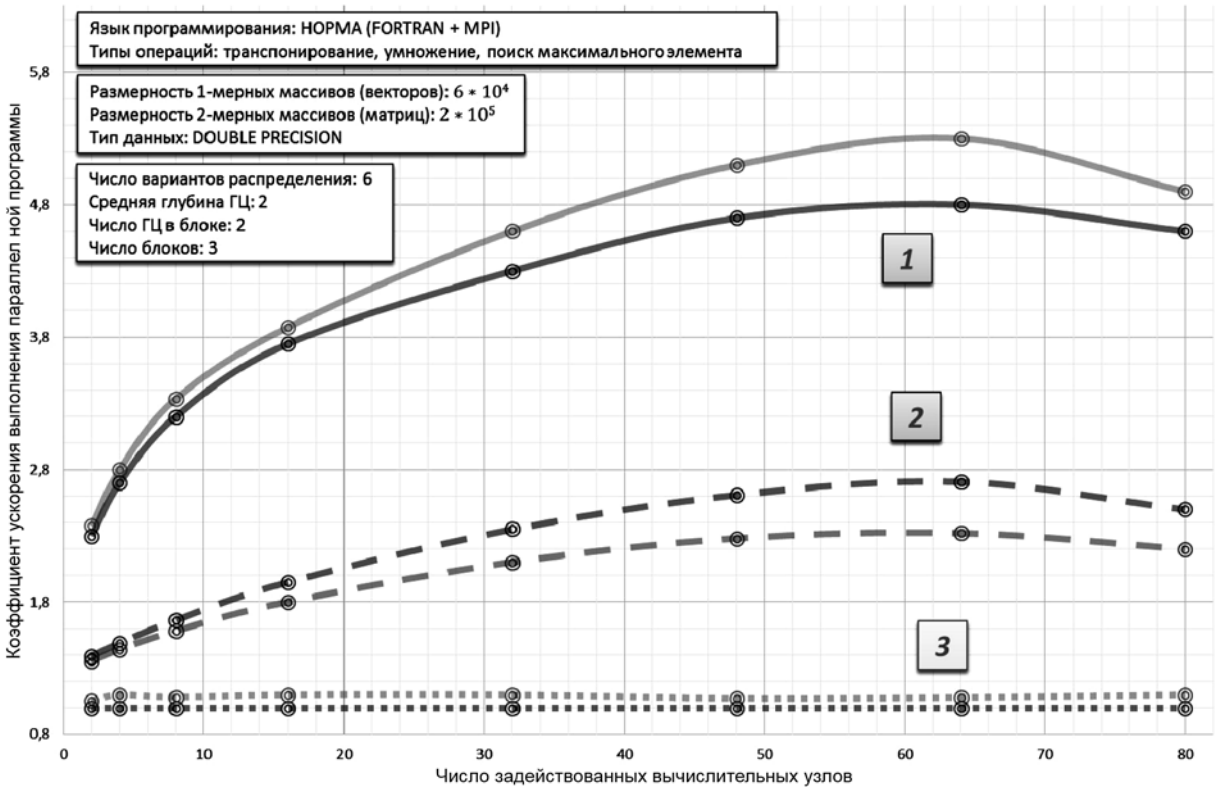


Рис. 7. Зависимость ускорения параллельной программы от значения распределения данных

корение при значениях распределения *jik* – *ji* – *ji/jki* – – *ij* – *ij*; цифрой *2* — пример двух вариантов распределения, демонстрирующий относительно незначительное ускорение при значениях распределения *ikj* – – *ij* – *ij/ijk* – *ij* – *ij*; цифрой *3* обозначен пример двух худших вариантов распределения без ускорения при значениях распределения *kij* – *ji* – *ji/kij* – *ji* – *ji*. Для данной параллельной программы использование различных вариантов размещения исходных данных в распределенной памяти МВС оказывает существенное влияние на время обмена данными между ВУ (после преодоления порога в 48 ВУ — 4,6 раза по сравнению с наихудшим вариантом в группе *3* и 2,1 раза по сравнению с промежуточным вариантов в группе *2*).

Однако создание системы правил, на основе которой принимается решение о рациональном характере заданного распределения данных в программе, требует проведения значительного числа экспериментов на различных аппаратных платформах. Существуют также ограничения на набор типовых операций, которые могут быть оценены с помощью подобной системы правил, что накладывает отпечаток на целевой класс параллельных программ, который может содержать ограниченное число различных вариаций ГЦ. Несмотря на отмеченные ограничения, применение предложенной стратегии оптимизации в некоторых случаях позволяет добиться прироста производительности в 4—5 раз в зависимости от числа задействованных ВУ для параллельных программ, обрабатывающих массивы большой размерности.

Список литературы

1. **Статистика** 500 самых мощных компьютеров мира. Электронный ресурс. URL: http://top500.org/statistics/list

2. **Воеводин В. В., Воеводин Вл. В.** Параллельные вычисления. СПб.: БХВ-Петербург, 2004. 608 с.

3. **Баканов В. М.** Априорная количественная оценка эффективности параллельных программ на конкретных многопроцессорных системах // Программная инженерия. 2011. № 1. С. 34—38.

4. **Андрианов А. Н., Бугеря А. Б., Ефимкин К. Н., Задыхайло И. Б.** Норма. Описание языка. Рабочий стандарт. Препринт ИПМ им. М. В. Келдыша РАН. № 120. 1995. 52 с.

5. **Палагин В. В.** К вопросу об ускорении параллельных программ для научно-технических расчетов за счет преобразования вложенных циклов // Программная инженерия. 2013. № 2. С. 21—24.

6. **Воеводин Вл. В., Жуматий С. А., Соболев С. И.** и др. Практика суперкомпьютера "Ломоносов". М.: Открытые системы, 2012.

УДК 004.056.5

Д. А. Корнев, аспирант, инж., Московский государственный университет путей сообщения,
e-mail: da.kornev@gmail.com

Моделирование динамического состояния виртуальной инфраструктуры с использованием сетей Петри

Приведены результаты моделирования динамических процессов в виртуальной инфраструктуре при разных алгоритмах управления ее ресурсами. Определены критерии эффективности управления виртуальной инфраструктурой. Методами векторной оптимизации рассчитаны значения целевой функции при взаимовлияющих критериях оптимизации и различных алгоритмах управления ресурсами.

Ключевые слова: *виртуальная инфраструктура, сеть Петри, моделирование распределения ресурсов, оптимальное управление ресурсами*

D. A. Kornev

Dynamic Simulation of State Virtual Infrastructure Using Petri Nets

The article presents results of modeling of dynamic processes in a virtual environment for different algorithms for management of its resources. Defined performance criteria virtual infrastructure management. Methods of vector optimization objective function value calculated at conflicting criteria optimization algorithms and various resource management.

Keywords: *virtual infrastructure, Petri net, modeling resource allocation, optimal resource management*

В настоящее время корпоративные информационные системы являются одним из основных средств производства любого предприятия. Для успешного использования корпоративной информационной системы необходимо управлять ее ресурсами и обеспечивать требуемый уровень безопасности. Поскольку эти задачи должны решаться еще на стадии проектирования системы, к настоящему времени разработаны различные методы моделирования и прогнозирования функционирования информационной системы. К ним относят логико-вероятностные модели, модели нечеткой логики, ДП-модели. Однако, как отмечено в работах [1, 2], анализ динамики ресурсных потоков целесообразно выполнять с использованием аппарата сетей Петри. Этот аппарат позволяет объединить преимущества графового представления и дискретной динамической модели системы, рассчитывать количественные показатели ее работы, которые характеризуются параллельными и асинхронными процессами. Аппарат сетей Петри может быть с успехом использован и для определения эффективности работы корпоративной виртуальной инфраструктуры (ВИ).

Рассмотрим ВИ, созданную на базе хоста, и состоящую из хоста, гипервизора и виртуальных машин (ВМ). Обобщенные ресурсы хоста *SRC* (объем оперативной памяти, тактовая частота процессора, емкость, число операций ввода-вывода в секунду и время обращения к жесткому диску) позволяют одновременно функционировать пяти ВМ ($SRC = 5$). Связь ВМ с хостом и отдельных ВМ между собой осуществляется через гипервизор, который обеспечивает распределение ресурсов хоста между всеми ВМ. В конечном итоге, именно гипервизор определяет эффективность работы ВИ на доступном аппаратном обеспечении.

Для исследования работы этой системы разработана ее модель в терминах временной стохастической сети Петри, которая определяется совокупностью объектов [1, 2] (рис. 1):

$$\Pi = \{P,\ T,\ I,\ O,\ \mu\},$$

где $P = \{p_1, p_2, ..., p_n\}$ — конечное множество позиций; $T = \{t_1, t_2, ..., t_m\}$ — конечное множество переходов; I — входная функция переходов, определяющая кратность входных дуг переходов $I(t_j)$; O — выходная функция переходов, определяющая кратность выходных дуг переходов $O(t_j)$; μ — вектор маркировки.

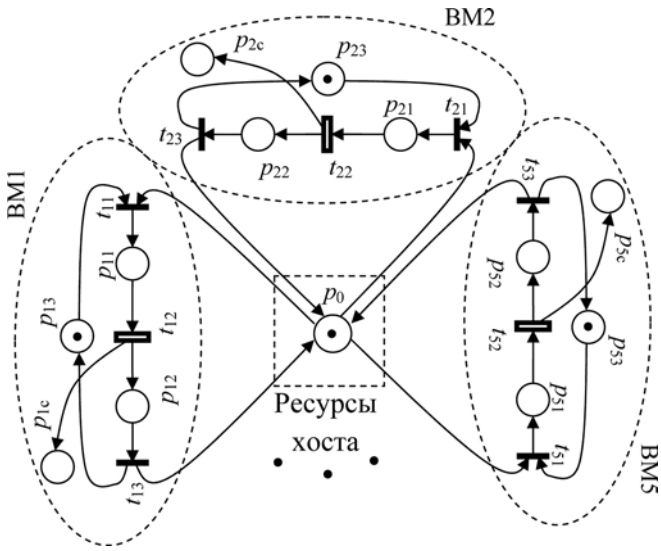


Рис. 1. Виртуальная инфраструктура в терминах сети Петри

Маркировка сети определяется отображением множества позиций на множество натуральных чисел N:

$$\mu: P \to N.$$

Графически, в терминах расширенных сетей Петри, модель ВИ представляется как двудольный ориентированный маркированный граф, состоящий из вершин двух типов — позиций и переходов, соединенных между собой дугами.

Разрешение на выполнение перехода $t_j \in T$ определяется условием [1, 2]

$$\mu(p_i) \geqslant \#(p_i, I(t_j)), \qquad (1)$$

т. е. переход t_j разрешен при некоторой маркировке $\mu(p_i)$, если позиция $p_j \in P$ имеет разметку не меньшую, чем кратность дуги, соединяющей p_i и t_j.

Результатом выполнения разрешенного перехода $t_j \in T$ является новая маркировка μ', определяющаяся соотношением [1]

$$\mu'(p_i) = \mu(p_i) - \#(p_i, I(t_j)) + \#(p_i, O(t_j)). \qquad (2)$$

Моделирование в сети Петри осуществляется на событийном уровне. Переходы отображают действия, происходящие в системе, а позиции — состояния, предшествующие этим действиям, и состояния, принимаемые системой после выполнения действия. Анализ результатов моделирования позволяет определить динамическое состояние системы при любых алгоритмах управления и выполняемых процедурах.

Модель функционирования ВИ, представленная в терминах сети Петри, содержит 21 позицию и 15 переходов:

$$P = \{p_0, p_{11}, p_{12}, p_{13}, p_{1c}, p_{21}, \ldots, p_{gl}, \ldots, p_{53}, p_{5c}\};$$

$$T = \{t_{11}, t_{12}, t_{13}, t_{21}, \ldots, t_{gf}, \ldots, t_{52}, t_{53}\}.$$

Элементами множества позиций P являются: p_0 — разделяемая среда (ресурс хоста); p_{11}—p_{13}, p_{21}—p_{23}, p_{31}—p_{33}, p_{41}—p_{43}, p_{51}—p_{53} — состояние ВМ 1-5; (p_{11}, p_{21}, p_{31}, p_{41}, p_{51} — предоставление ресурсов ВМ 1-5; p_{12}, p_{22}, p_{32}, p_{42}, p_{52} — освобождение ресурсов ВМ 1-5; p_{13}, p_{23}, p_{33}, p_{43}, p_{53} — ожидание ресурсов хоста ВМ 1-5); p_{1c}, p_{2c}, p_{3c}, p_{4c}, p_{5c} — счетчики операций.

Элементами множества позиций T являются: t_{11}—t_{13}, t_{21}—t_{23}, t_{31}—t_{33}, t_{41}—t_{43}, t_{51}—t_{53} — процессы распределения ресурсов хоста и формирования запросов ВМ 1-5 (t_{11}, t_{21}, t_{31}, t_{41}, t_{51} — выделение ресурсов ВМ 1-5; t_{12}, t_{22}, t_{32}, t_{42}, t_{52} — работа с предоставленными ресурсами ВМ 1-5; t_{13}, t_{23}, t_{33}, t_{43}, t_{53} — возвращение ресурсов ВМ 1-5 хосту).

Для проверки адекватности модели реальной ВИ проанализированы результаты моделирования работы ВИ, состоящей из хоста с $SRC = 1$ и двух ВМ (рис. 2). Динамическая модель такой ВИ в терминах сетей Петри в соответствии с формулами (1)—(2) будет иметь следующий вид:

$$\begin{cases}
\mu'(p_0) = \mu(p_0) + 1(\#(p_0, I(t_{13})) = 1) + \\
+ 1(\#(p_0, I(t_{23})) = 1) - 1(\#(p_0, O(t_{11})) = 1) - \\
- 1(\#(p_0, O(t_{21})) = 1); \\
\mu'(p_{11}) = \mu(p_{11}) + 1(\#(p_{11}, I(t_{11})) = 1) - \\
- 1(\#(p_{11}, O(t_{12})) = 1); \\
\mu'(p_{12}) = \mu(p_{12}) + 1(\#(p_{12}, I(t_{12})) = 1) - \\
- 1(\#(p_{12}, O(t_{13})) = 1); \\
\mu'(p_{13}) = \mu(p_{13}) + 1(\#(p_{13}, I(t_{13})) = 1) - \\
-1(\#(p_{13}, O(t_{11})) = 1); \\
\mu'(p_{1c}) = \mu(p_{1c}) + 1(\#(p_{1c}, I(t_{12})) = 1); \\
\mu'(p_{21}) = \mu(p_{21}) + 1(\#(p_{21}, I(t_{21})) = 1) - \\
- 1(\#(p_{21}, O(t_{22})) = 1); \\
\mu'(p_{22}) = \mu(p_{22}) + 1(\#(p_{22}, I(t_{22})) = 1) - \\
- 1(\#(p_{22}, O(t_{23})) = 1); \\
\mu'(p_{23}) = \mu(p_{23}) + 1(\#(p_{23}, I(t_{23})) = 1) - \\
- 1(\#(p_{23}, O(t_{21})) = 1); \\
\mu'(p_{2c}) = \mu(p_{2c}) + 1(\#(p_{2c}, I(t_{22})) = 1).
\end{cases} \qquad (3)$$

Функции входов I и выходов O для (3) представлены в табл. 1 и 2 соответственно.

В соответствии с формулой (3) было выполнено моделирование работы ВИ с двумя ВМ. Моделью предполагалось, что ресурсы хоста предоставляются любой из ВМ на одинаковый интервал времени Δt_p = = 50 усл. ед. времени. Динамический процесс работы для каждой ВМ в терминах сети Петри представляет собой следующие операции: запрос на предоставление ресурсов хоста; ожидание получения ресурсов Δt_o; работа с предоставленными ресурсами Δt_p; возвращение ресурсов хосту (рис. 3). На диаграммах работы системы видно, что ресурс сначала предоставляется первой ВМ, в это время вторая ВМ ожидает ресурс. Когда первая ВМ освобождает ресурс, он передается второй ВМ и т. д.

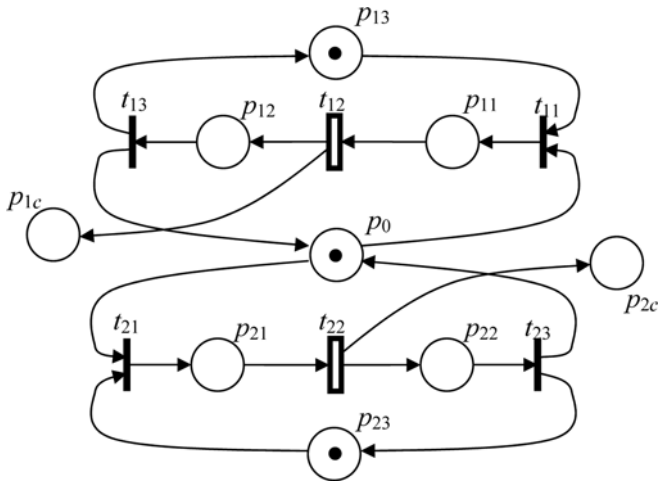


Рис. 2. Виртуальная инфраструктура из двух ВМ в терминах сети Петри

Таблица 2

Матрица выходов *O* модели ВИ с *SRC* = 1 и двумя ВМ

	t_{11}	t_{12}	t_{13}	t_{21}	t_{22}	t_{23}
p_0	0	0	1	0	0	1
p_{11}	1	0	0	0	0	0
p_{12}	0	1	0	0	0	0
p_{13}	0	0	1	0	0	0
p_{1c}	0	1	0	0	0	0
p_{21}	0	0	0	1	0	0
p_{22}	0	0	0	0	1	0
p_{23}	0	0	0	0	0	1
p_{2c}	0	0	0	0	1	0

Таблица 1

Матрица входов *I* модели ВИ с *SRC* = 1 и двумя ВМ

	t_{11}	t_{12}	t_{13}	t_{21}	t_{22}	t_{23}
p_0	1	0	0	1	0	0
p_{11}	0	1	0	0	0	0
p_{12}	0	0	1	0	0	0
p_{13}	1	0	0	0	0	0
p_{1c}	0	0	0	0	0	0
p_{21}	0	0	0	0	1	0
p_{22}	0	0	0	0	0	1
p_{23}	0	0	0	1	0	0
p_{2c}	0	0	0	0	0	0

В том случае, если ВИ с *SRC* = 2 содержит две ВМ, то характер диаграмм, отражающих работу системы, меняется (рис. 4). Когда ресурсов хоста достаточно для обслуживания двух ВМ, он может предоставляться им одновременно. Это исключает интервалы ожидания ресурса для обеих ВМ в процессе их обслуживания. При этом ресурс хоста, как и в случае *SRC* = 1, используется непрерывно. Очевидно, что при *SRC* = 2 эффективность работы ВМ повышается при такой же эффективности работы хоста.

Таким образом, анализ процессов распределения ресурсов в ВИ, полученных путем моделирования, соответствует работе реальной системы. Если ресурсов ВИ достаточно только для работы одной ВМ, то вторая ВМ всегда находится в состоянии ожидания, пока ресурс не освободится, только после этого ресурс становится доступен для второй ВМ. При этом ресурс ВИ используется непрерывно. В случае, когда ресурсов

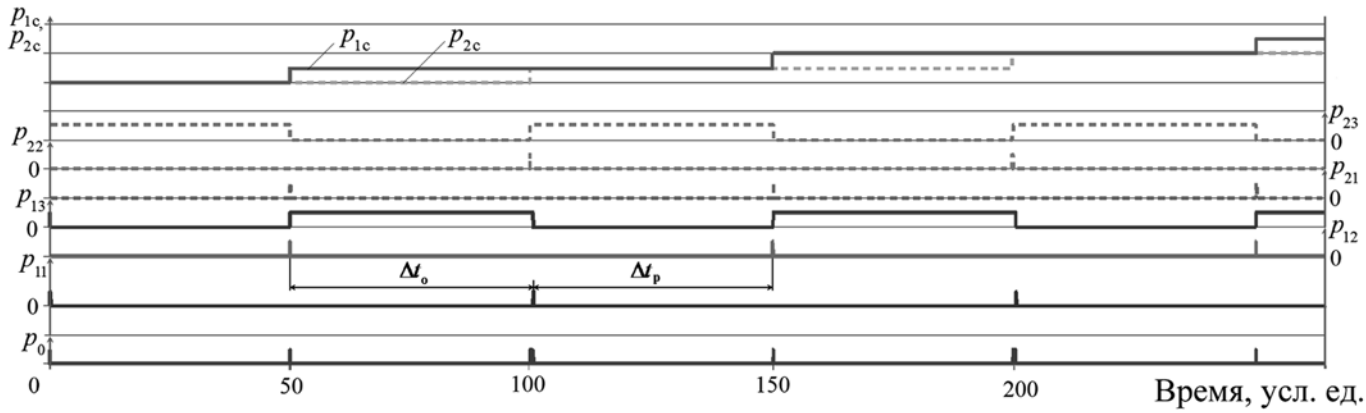


Рис. 3. Динамический процесс работы ВИ при *SRC* = 1 и двух ВМ с одинаковым временем использования ресурса

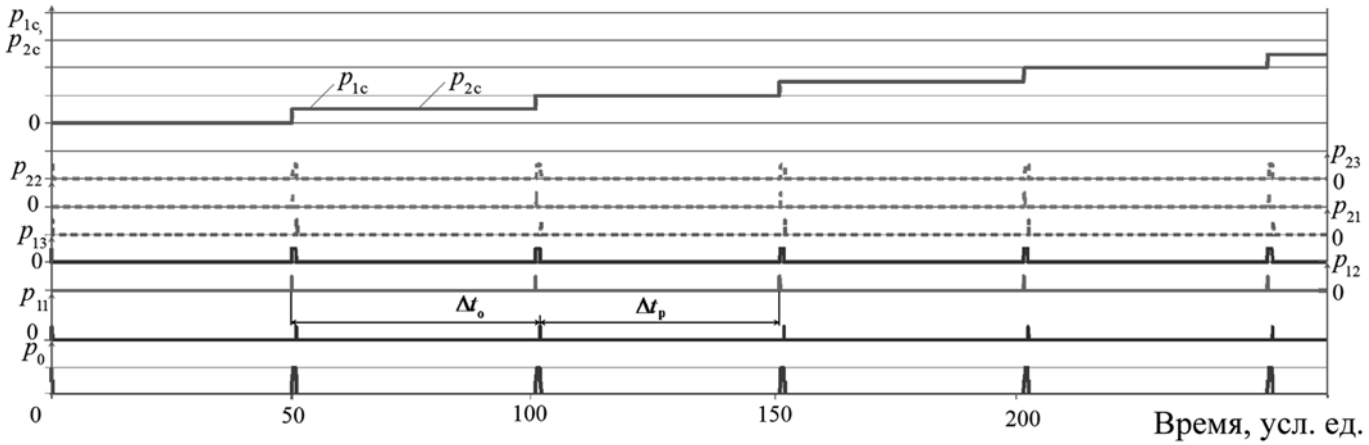


Рис. 4. Динамический процесс работы ВИ при *SRC* = 2 и двух ВМ с одинаковым временем использования ресурса

хоста достаточно для функционирования двух ВМ, они могут работать непрерывно и независимо друг от друга, что получено при моделировании ВИ.

В реальных условиях ВИ работает со случайным характером загрузки. Этот процесс был рассчитан с использованием разработанной модели для ВИ с *SRC* = 5 и пятью ВМ. Время предоставления ресурса хоста каждой ВМ задавалось с использованием генератора случайных чисел (рис. 5). На представленных диаграммах видно, что, если продолжительность предоставления ресурсов ВМ является случайной величиной, характер загрузки хоста ВИ меняется: запросы на предоставление ресурса от ВМ поступают хаотично, время их работы с ресурсом в соответствии с алгоритмом работы генератора случайных чисел меняется по равномерному закону распределения.

При создании любой ВИ важно определить ее структуру и закон управления ресурсами. Как и всякой сложной системе, ВИ соответствуют различные критерии оценки эффективности работы. Это приво-

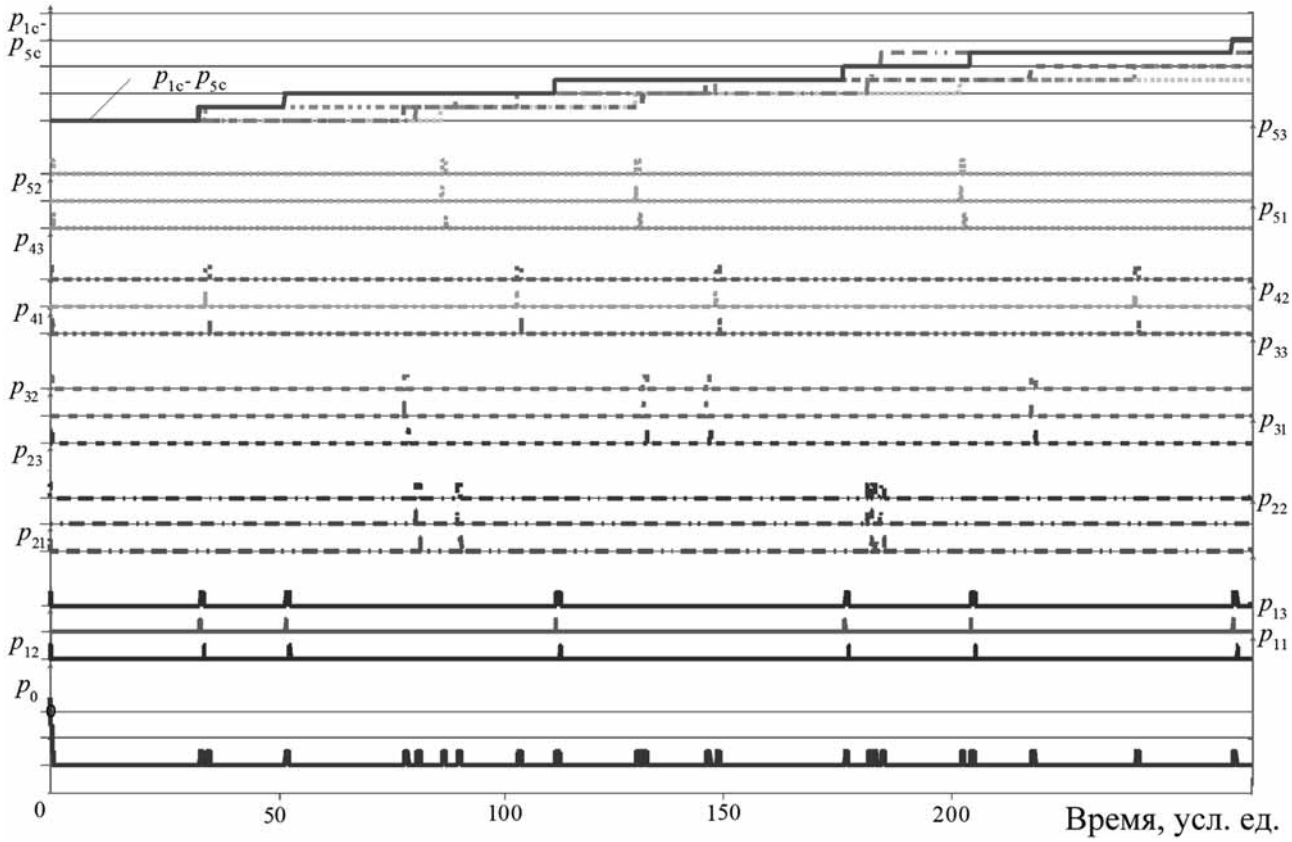


Рис. 5. Динамический процесс работы ВИ при *SRC* = 5 и пяти ВМ со случайным временем использования ресурса

дит к задаче оптимизации с векторной целевой функцией

$$Ц(u) = \{K_1(u),\ K_2(u),\ ...,\ K_g(u),\ ...,\ K_G(u)\} \to \min,$$

где $K_1(u)$, $K_2(u)$, ..., $K_g(u)$, ..., $K_G(u)$ — частные критерии оптимизации; u — параметр управления, принадлежащий множеству U, $u \in U$.

В связи с этим поиск эффективного алгоритма управления ресурсом ВИ относится к задачам векторной оптимизации [3]. Для пользователя в качестве основных критериев эффективности выступают, как правило, быстродействие работы и степень использования ресурса ВИ. В качестве быстродействия целесообразно рассматривать время ожидания ресурса каждой ВМ Δt_o, а степень использования ресурса количественно можно оценивать по числу ВМ, функционирующих на хосте с данным ресурсом. Очевидно, что критерий $K_1 = \Delta t_o$, определяющий время ожидания ресурса, должен иметь минимальные значения, а критерий $K_2 = g$, определяющий число ВМ, функционирующих на хосте с ограниченным ресурсом, — максимальные значения:

$$\begin{cases} K_1 = K_1(u) \to \min \\ K_2 = K_2(u) \to \max, \end{cases} \qquad (4)$$

где u для данной задачи можно рассматривать как распределение ресурса ВИ между отдельными ВМ.

Поиск оптимального распределения ресурса такой системы сводится к определению множества неулучшаемых решений (оптимизации по Парето) [4, 5]. В данном случае множеству неулучшаемых решений соответствует отрезок $[a, b]$ функции $K_1(u) = f(K_2(u))$ (рис. 6), а оптимальное решение определяется приближением значений K_1 и K_2 к утопической точке $K_{ут}$, которая по определению не принадлежит к множеству неулучшаемых решений и поэтому достигнута быть не может.

В задачах векторной оптимизации для отыскания вектора управлений, обеспечивающих парето-оптимальные решения, используют обобщенный критерий

$$\sum_{q=1}^{Q} a_q K_q(u) = \min_{u \in U}, \qquad (5)$$

где a_q — весовой коэффициент q-го критерия оптимизации.

При анализе работы ВИ существует неопределенность, обусловленная выбором значений весовых коэффициентов для зависимости (5) в линейной комбинации критериев. Поэтому в данном случае целесообразно использовать минимаксное решение задачи оптимизации, когда минимизируется наибольшее из отклонений каждого критерия от утопической точки (минимаксное чебышевское решение) [5]:

$$\max|K_q(u) - K_{ут}| \to \min_{u \in U}.$$

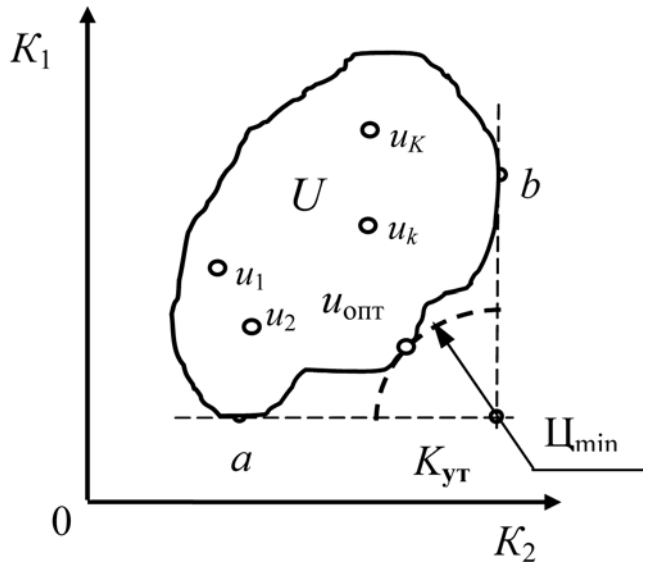


Рис. 6. Определение множества неулучшаемых решений распределения ресурса ВИ:
u_1, u_2, ..., u_k, ..., u_K — параметры управления

Поскольку в данном случае эффективность использования ВИ определяется двумя взаимосвязанными критериями $Ц(K_1, K_2)$, оптимизацию целесообразно выполнять с использованием стратегии взвешенных сумм [5]. Использование этой стратегии позволяет многокритериальную задачу минимизации вектора $Ц(u)$ преобразовать в скалярную величину, представляющую собой взвешенную сумму всех критериев

$$\text{minimize}_{u \in U} Ц(u) = \sum_{q=1}^{Q} w_q K_q(u)^2, \qquad (6)$$

где $\text{minimize}_{u \in U} Ц(u)$ — минимизация наибольшего из отклонений функции $Ц(u)$ от утопической точки; w_q — взвешенные коэффициенты, которые должны соответствовать относительной значимости частных критериев.

Применительно к ВИ с учетом зависимости (5) оптимальное решение находится минимизацией расстояния от точки $K_{ут}$ до линии неулучшаемых решений на плоскости критериев (K_1, 0, K_2) (см. рис. 6):

$$\begin{cases} Ц(u) = \min; \\ Ц^2(u) = [K_1(u) - K_{1\min}]^2 + [K_2(u) - K_{2\max}]^2. \end{cases} \qquad (7)$$

Таким образом, в соответствии с формулой (7) $Ц(u)$ определится минимизацией радиус-вектора от $K_{ут}$ до линии неулучшаемых решений (см. рис. 6).

Поскольку критерии, принятые в соответствии с (2), имеют разную размерность, а их значения могут отличаться на несколько порядков (в зависимости от масштабов измерения значений K_1 и K_2), целевую

функцию целесообразно определить через их относительные величины

$$\begin{cases} \bar{K}_{1g}(u) = \dfrac{K_{1g}(u) - K_1^*}{K_1^{**} - K_1^*}; \\ \bar{K}_{2g}(u) = \dfrac{K_{2g}(u) - K_2^*}{K_2^{**} - K_2^*}, \end{cases} \quad (8)$$

где K_1^*, K_2^*, K_1^{**}, K_2^{**} — минимальные и максимальные значения частных критериев, найденных при решении задачи оптимизации по данному критерию соответственно; K_{1g}, K_{2g} — текущие значения частных критериев, полученные при параметрах управления, расположенных в множестве Парето.

С учетом систем (7) и (8) при равной значимости принятых критериев эффективности использования ресурса ВИ целевая функция в относительной системе координат определится как

$$\text{Ц}(u) = \sqrt{(\bar{K}_{1g}(u))^2 + (\bar{K}_{2g}(u))^2} \to \min. \quad (9)$$

В соответствии с формулой (9) была исследована ВИ с ресурсом $SRC = 5$ и различным числом ВМ $1 \leqslant g \leqslant 20$, функционирующих на хосте. Предполагалось, что выделение ресурса каждой ВМ носит случайный характер, т. е. время работы каждой ВМ с ресурсом изменяется в диапазоне $10 \leqslant \Delta t_p \leqslant 100$ усл. ед. времени и имеет равномерное распределение $F(\Delta t_p) = \text{const}$. При этих условиях по формуле (8) были рассчитаны относительные значения критериев эффективности $\bar{K}_1(u)$ и $\bar{K}_2(u)$ (рис. 7). Результаты расчетов показали, что при принятых ограничениях $10 \leqslant \Delta t_p \leqslant 100$ усл. ед.

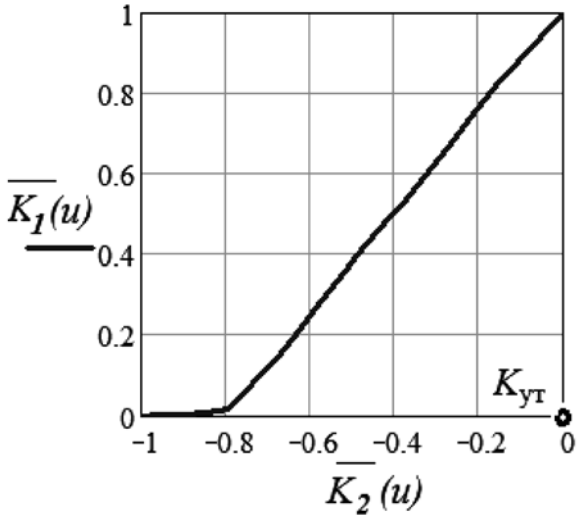


Рис. 7. Функциональная зависимость частных критериев эффективности использования ресурсов ВИ при $SRC = 5$ и $F(\Delta t_p) = \text{const}$

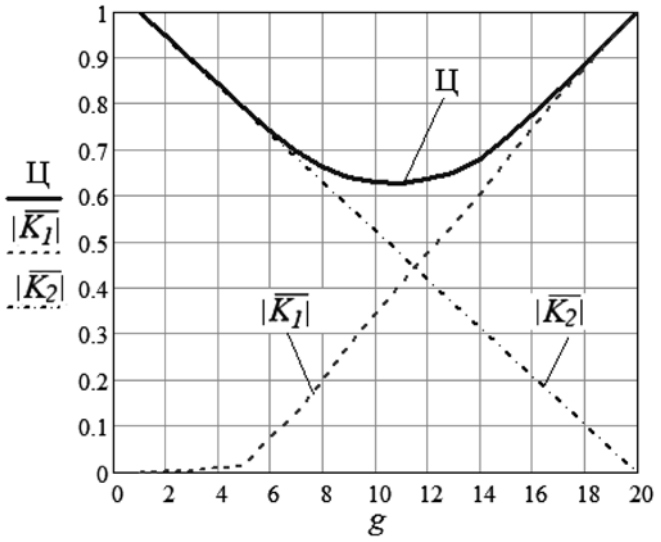


Рис. 8. Значения целевой функции и относительных частных критериев эффективности в зависимости от числа ВМ, функционирующих на хосте при $SRC = 5$ и $F(\Delta t_p) = \text{const}$; $|\bar{K}_1|$, $|\bar{K}_2|$ — абсолютные значения относительных критериев эффективности

времени и $1 \leqslant g \leqslant 20$ все полученные значения зависимости $\bar{K}_1(u) = f(\bar{K}_2(u))$ относятся к множеству неулучшаемых решений управления ВИ.

По этой причине целевая функция эффективности использования ресурса ВИ Ц(u) рассчитывалась по формуле (9). Получено, что при равной значимости частных критериев эффективности $\bar{K}_1(u)$ и $\bar{K}_2(u)$ ВИ с ресурсом $SRC = 5$ при $F(\Delta t_p) = \text{const}$ будет использоваться наиболее эффективно, если на хосте функционируют одиннадцать ВМ (рис. 8).

Анализ значений Ц(u) показывает, что алгоритм распределения ресурса между ВМ значительно влияет на эффективность работы всей ВИ. Очевидно, что предложенная методика расчета функционирования ВИ, базирующаяся на ее динамической модели в терминах сетей Петри и методах векторной оптимизации распределения ресурсов, позволит определять рациональную структуру ВИ и при задании других критериев эффективности с различной степенью их значимости.

Список литературы

1. **Котов В. Е.** Сети Петри. М.: Наука: Главная редакция физико-математической литературы, 1984. 160 с.

2. **Лескин А. А., Мальцев П. А., Спиридонов А. М.** Сети Петри в моделировании и управлении. Л.: Наука, 1989. 133 с.

3. **Наумов В. С.** Использование сетей Петри при моделировании процесса транспортно-экспедиционного обслуживания // Автомобильный транспорт. 2009. № 24. С. 120—124.

4. **Соболь И. М.** Выбор оптимальных параметров в задачах со многими критериями. М.: Дрофа, 2006. 175 с.

5. **Теория** автоматического управления: Нелинейные системы / Под ред. А. В. Нетушила. 2-е изд., перераб. и доп. М.: Высшая школа, 1983. 432 с.

6. **Трифонов А. Г.** Многокритериальная оптимизация. URL: http://matlab.exponenta.ru/optimiz/book_1/16.php

УДК 004.946, 519.17

М. В. Михайлюк, д-р физ.-мат. наук, зав. отд., НИИ системных исследований РАН, г. Москва,
e-mail: mix@niisi.ras.ru

Двумерные виртуальные пульты управления в тренажерных комплексах

Описана технология создания и использования виртуальных двумерных пультов управления в тренажерных и обучающих комплексах. Технология обеспечивается четырьмя программными компонентами — визуальным редактором для создания виртуального пульта; модулем моделирования движения элементов управления, редактором функциональных управляющих схем и модулем расчета управляющих сигналов.

Ключевые слова: *виртуальные пульты управления, системы визуализации, тренажерные комплексы*

M. V. Mikhaylyuk

2D Virtual Control Panels in Simulators

In the paper we describe the technology and using of virtual 2D control panels for simulators and learning systems. Technology is provided by four program components — visual editor for creation the virtual control panel, control elements moving modeling module, editor for functional control schemes and the module for computing control signals.

Keywords: *virtual control panels, visualization systems, simulators*

Введение

В настоящее время во многих задачах возникает необходимость осуществлять управление сложными динамическими объектами или процессами с помощью пультов управления. К таким задачам относятся управление роботами и манипуляторами, управление работой атомных станций и ситуационных центров, управление летательными аппаратами и т. д. Обучение и тренировки операторов управлению сложными динамическими процессами в реальной среде проводить затруднительно, а часто и невозможно. Это связано, прежде всего, с большой стоимостью сложных технических устройств, возможностью их поломки или повреждения (особенно на начальных стадиях тренировки), трудностью моделирования агрессивной среды и опасных условий работы, невозможностью отработки нештатных ситуаций и т. д. Поэтому в последнее время все большее распространение получили видеотренажерные системы, в которых оператор осуществляет тренировку в виртуальной среде, управляя виртуальными моделями реальных роботов, манипуляторов, летательных аппаратов и т. д. [1, 2]. В этих системах целесообразно использовать и виртуальные пульты управления. Создание виртуальных пультов и их эксплуатация требуют малых затрат, кроме того, работа с ними может проводиться еще на стадии проектирования реальных пультов. В этом случае на виртуальных моделях пультов можно проверять их эргономичность и удобство работы, подгоняя их конструкцию к удобному и безопасному использованию. Виртуальные пульты можно использовать в видеотренажерах, виртуальных лабораториях, обучающих стендах и даже в реальных установках вместо реальных пультов управления. Технологически для этого можно использовать сенсорные экраны.

Следует выделить два подхода к реализации работы системы управления: на основе событийной логики и на основе структурной схемы [3]. Первый подход основан на событиях, которые генерируются при переключении виртуальных элементов управления. Система управления в соответствии с этими событиями выполняет команды, изменяющие параметры одного или группы объектов виртуальной сцены. Недостатком этого подхода является сложная реализация обработки событий для случаев, когда реакция на переключение элемента управления зависит от состоя-

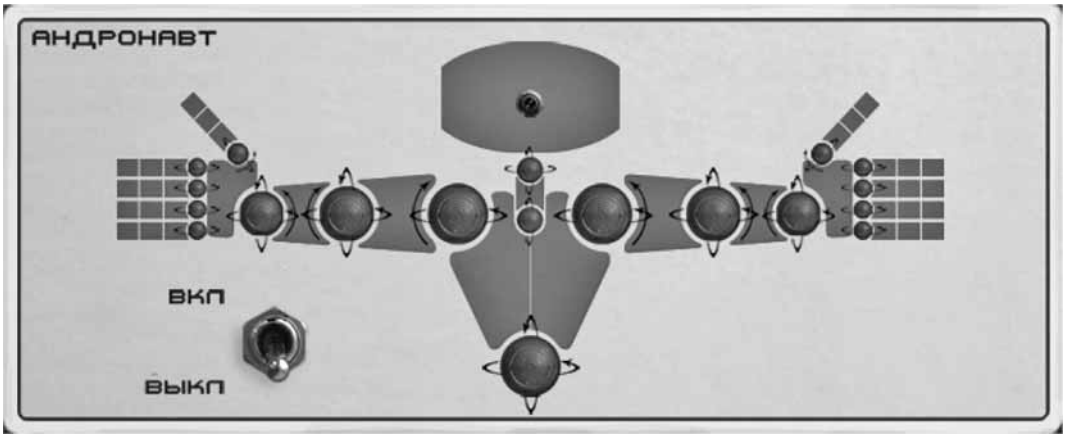


Рис. 1. Пульт управления роботом

ния других элементов. Поэтому такой подход применяется для пультов с простой логикой.

Второй подход основан на обработке сигналов. Предполагается, что каждый виртуальный элемент управления формирует некоторый виртуальный сигнал, значение которого зависит от состояния элемента.

В настоящей статье описаны технология и методы создания и работы с виртуальными двумерными (2D) пультами управления. Эта технология включает четыре программных компонента: визуальный редактор для создания виртуального пульта; модуль моделирования движения элементов управления; редактор функциональных управляющих схем и модуль расчета управляющих сигналов.

Создание двумерного виртуального пульта

Двумерный пульт управления отображается на экране компьютера в виде картинки. Визуальный редактор для создания двумерных виртуальных пультов представляет собой специальный конструктор, содержащий библиотеку различных элементов, в число которых входят:

- статические двумерные изображения (в частности, для моделирования подложки пульта);
- динамические элементы, включающие наборы статических изображений, соответствующих разным состояниям элемента управления;
- кнопки различного типа — фиксируемые и нефиксируемые, с автоотжатием и без него;
- переключатели разного типа, например, двухпозиционные и трехпозиционные тумблеры, галетные переключатели с несколькими фиксированными состояниями;
- регуляторы с плавным изменением параметра;
- джойстики с одной или двумя степенями свободы, позволяющие управлять, например, двигателями движущихся механизмов и т. д.

Для каждого из элементов двумерного виртуального пульта на этапе конструирования должны быть назначены одно или несколько двумерных изображений, описывающих визуальное представление элемента в разных его состояниях. Эти изображения могут быть созданы в произвольном графическом редакторе или получены с помощью съемки элементов реальных пультов управления, на основе которых создают виртуальные пульты. Например, для трехпозиционной кнопки требуется три различных изображения. Элементы можно просто перетаскивать из библиотеки на поле создаваемого пульта, задавая для них необходимые параметры работы (тип, число фиксированных положений, набор изображений элемента в различных позициях и т. д.). На рис. 1 показан виртуальный пульт управления антропоморфным роботом, созданный с помощью описанной технологии.

Размер пульта управления можно задать любым и изменять его во время создания пульта. Важной опцией является задание реального размера виртуального пульта на любом мониторе. Это бывает необходимо, чтобы избежать привития оператору ложных навыков, связанных с размером пульта (т.е. у оператора могут возникнуть затруднения при работе на реальном пульте, размер которого отличен от размера виртуального пульта, на котором проводились тренировки). После создания виртуальный пульт сохраняется в файле, из которого он может быть впоследствии загружен для корректировки или доработки пульта управления.

Моделирование интерактивного воздействия на элементы управления

Пользователь (оператор) осуществляет воздействие на управляющие элементы двумерного виртуального пульта с помощью мыши, либо с помощью сенсорного экрана. При наведении указателя мыши на произвольный элемент управления может отобра-

Рис. 2. Положения тумблера при переключении

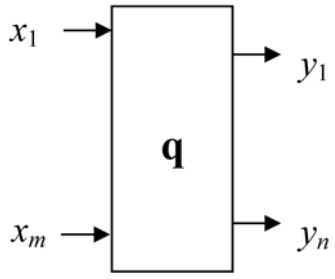


Рис. 3. Блок структурной схемы

жаться всплывающая текстовая подсказка с описанием этого элемента. Воздействие пользователя на элементы управления включает в себя такие стандартные операции как одиночное нажатие, двойное нажатие, нажатие и последующее перемещение и отпускание элемента. Определяя положение курсора мыши на экране, направление его движения и тип нажатия мыши, можно вычислить, на какой элемент управления 2D-пульта и в какую сторону воздействует пользователь. На основе этого (в зависимости от типа элемента) моделируются и визуализируются движения элемента, например, кнопка нажимается, тумблер переключается и т. д. Для этого используют сохраненные при создании пульта изображения элемента в различных его положениях (рис. 2). При переключении элемента учитывается достаточность «усилия» воздействия, а именно — тумблер переключается только тогда, когда курсор мыши перешел через среднее положение переключателя. В противном случае он возвращается в текущее положение.

Редактор функциональных управляющих схем

Управляющим элементам виртуального пульта соответствуют функциональные схемы, которые включаются в общую схему системы управления. Задачей этих схем является формирование управляющих сигналов при воздействии пользователя на элементы управления. Например, при нажатии двухпозиционной кнопки некоторая сигнальная лампочка может загореться, а при ее отжатии — потухнуть. Функциональные управляющие схемы также создаются с помощью специального редактора, в котором реализована библиотека функциональных блоков (логические, арифметические, алгебраические, дифференциальные и т. д.). Преимущество использования такого редактора для создания функциональной схемы системы управления заключается в наглядном способе ее представления. Каждый блок схемы выводится в редакторе в виде пиктограммы, которая отображает либо элемент управления, с которым связан данный блок, либо символ, определяющий функциональное назначение блока. Блок имеет некоторое число входов m, некоторое число выходов n, вектор $\mathbf{q}$ внутренних состояний и реализует некоторый вектор функций $(y_1, ..., y_n) = (f_1(x_1,...x_m, \mathbf{q}), ..., f_n(x_1,...x_m, \mathbf{q}))$. В общем случае значения x_i, q_i и f_i являются действительными числами, в частности, они могут быть целыми числами из некоторого диапазона или булевскими значениями из множества {0,1}.

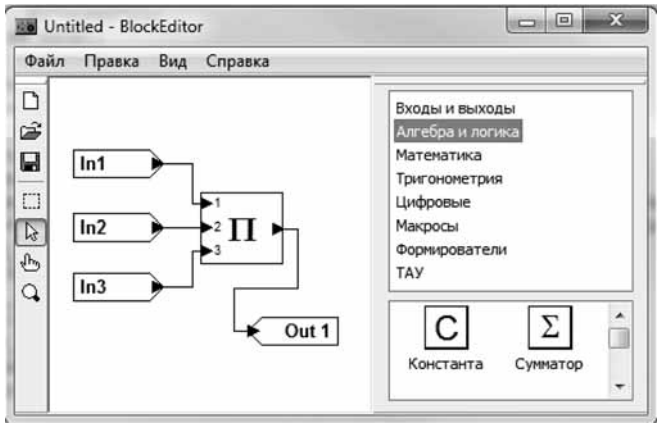


Рис. 4. Пример функциональной схемы

На рис. 3 показано общее схематическое изображение блока. В качестве примеров можно привести блоки, реализующие сумму или произведение всех входов, логические операции от двух переменных (конъюнкция, дизъюнкция, отрицание, сложение по модулю 2 и т. д.), тригонометрические функции (синус, косинус, арктангенс и т. д.) и другие операции. Многие блоки имеют индивидуальные параметры настройки, которые задаются через отдельное диалоговое окно. Например, для блока «Сумматор» через диалог настройки можно задать количество входов. Связи между блоками в редакторе представляются в виде линий. Кроме этого, сами блоки, а также их входы и выходы в редакторе имеют всплывающие подсказки, что позволяет быстро получить краткую информацию о них. Все это в целом позволяет быстро разрабатывать, анализировать и выявлять ошибки функционирования структурной схемы системы управления.

Блоки можно перетаскивать мышью из библиотеки на поле редактора, задавать их параметры и соединять их входы и выходы между собой. Некоторые входы блоков образуют входы функциональной схемы и привязываются к элементам управления. Аналогично, некоторые выходы образуют выходы схемы, сигналы с которых подаются на приборы, сигнальные лампочки или исполнительные органы. В результате воздействия на элемент управления на выходе его блока формируется новый управляющий сигнал (например, для кнопки — логический 0 или 1, для джойстиков и регуляторов — вещественное значение от 0 до 1). В результате, на выходах функциональной схемы также вычисляется одно или несколько результирующих значений, которые далее передаются на исполнительные механизмы динамической системы (например, входные напряжения электрических двигателей для виртуальных роботов или манипуляторов, сигналы включения и выключения фар виртуальных механизмов и другие управляющие параметры). На рис. 4 показан пример управляющей функциональной схемы в редакторе. После создания структурной схемы в редакторе она сохраняется в бинарном файле.

Модуль расчета управляющих сигналов

Модуль расчета управляющих сигналов выполняет вычисления выходных значений функциональной схемы в реальном масштабе времени (т.е., например, каждые 40 мс) на основе новых входных значений. При этом используется иерархическая структура схемы, позволяющая правильно организовать последовательность вычислений. Для каждого выхода каждого блока вычисляется его глубина — максимальная длина пути от данного выхода блока до выходов схемы. Если выход не связан ни с одним блоком, то его глубина равна 0; если с выходом связан только один блок, который не имеет выходов или эти выходы не связаны с другими блоками, то глубина равна 1 и т. д. На рис. 5 представлена схема, на которой для каждого выхода указана его глубина. Глубины выходов используют для определения последовательности обработки блоков в процессе вычисления управляющих сигналов функциональной схемы. Кроме того, для ускорения обработки проводится пересчет только тех поддеревьев схемы, которые содержат блоки, изменившие свое состояние со времени предыдущего расчета. Если состояние элементов управления не меняется, то расчет схемы не проводится (если нет блоков, генерирующих сигнал). Например, как видно на рис. 5, при нажатии на виртуальную кнопку *3* при расчете схемы участвуют только три блока: блок данной кнопки, блок «&» и блок, связанный с индикатором виртуального пульта управления. Такой подход позволяет значительно уменьшить вычислительные издержки при расчете схемы по сравнению с тем, как если бы схема считалась

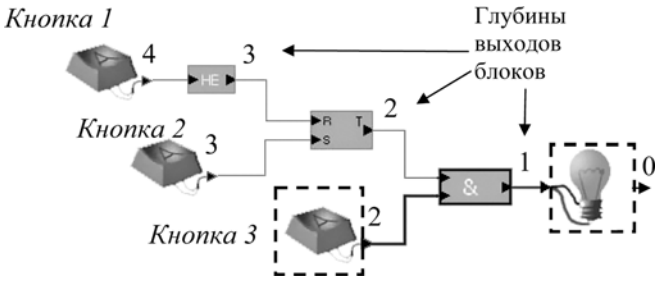


Рис. 5. Глубины выходов схемы

каждый раз полностью. Это обеспечивает реальный режим времени работы системы управления при использовании сложных структурных схем.

Вычисления в схеме проводятся многократно, например, для каждого кадра визуализации. Будем называть каждое такое вычисление тактом. В этом случае результаты вычислений некоторых блоков одного такта могут быть использованы в этих блоках для вычислений в других (последующих) тактах. Тогда они сохраняются в памяти блока в виде вектора **q** его внутренних состояний.

Использование виртуальных пультов в тренажерных системах

Видеотренажерный комплекс в общем случае состоит из подсистемы управления, подсистемы расчета динамики и подсистемы визуализации (рис. 6). Опишем схему работы видеотренажера. Оператор видит на экране монитора виртуальную сцену. Для изменения положения или состояния динамического объекта

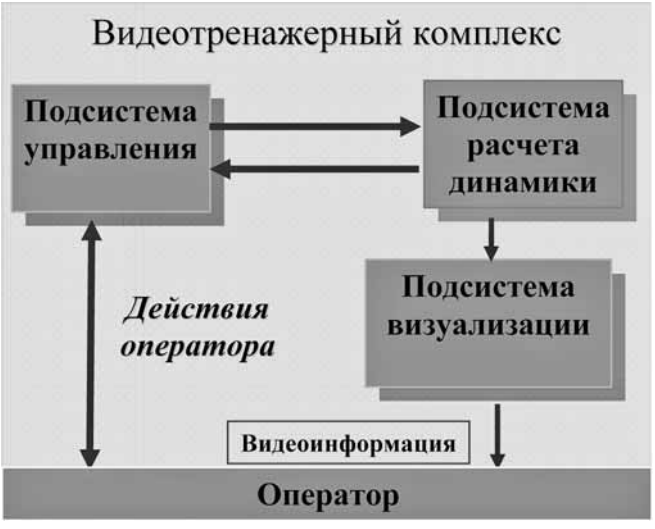


Рис 6. Структура видеотренажера

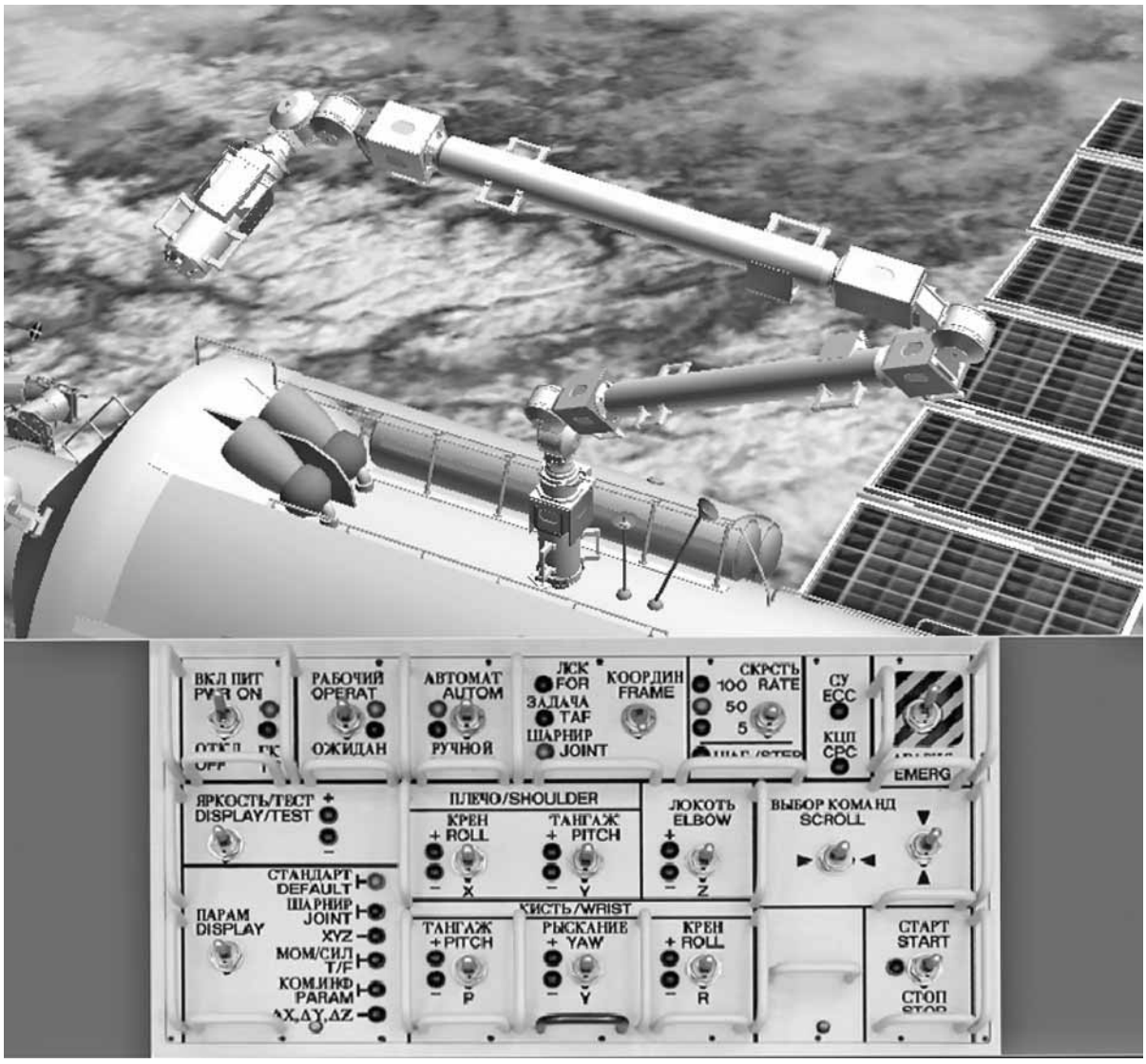


Рис. 7. Модель космического манипулятора "Эра" с виртуальным пультом управления

сцены он воздействует на элементы управления. В результате воздействия система управления вычисляет управляющие сигналы и передает их в подсистему динамики. Подсистема динамики рассчитывает новые положение, ориентацию или состояние управляемого объекта и передает эту информацию в подсистему визуализации. Эта последняя подсистема осуществляет синтез изображения виртуальной сцены с новыми параметрами динамического управляемого объекта на экране монитора. Так как весь этот цикл занимает не более 40 мс, то у оператора создается впечатление плавного и непрерывного движения динамических объектов.

В НИИСИ РАН разработан полный набор указанных выше подсистем видеотренажерного комплекса [4, 5]. Реализация данного комплекса проведена на персональных компьютерах под управлением операционной системы Windows с использованием графической библиотеки OpenGL и графических акселераторов Geforce (версии 7600 и выше). В настоящее время система успешно эксплуатируется в ряде организаций, к числу которых относится Российский государственный научно-исследовательский испытательный центр подготовки космонавтов им. Ю.А.Гагарина. На рис. 7 показана стыковка космических модулей с помощью манипулятора «Эра», при этом управление манипулятором осуществляется с помощью двумерного виртуального пульта.

Заключение

Дальнейшее развитие направления виртуальных пультов управления возможно в области технологий виртуального окружения. Для этого необходимо реа-

лизовывать трехмерные пульты и обеспечивать взаимодействие с ними оператора в стереорежиме с помощью компьютерной перчатки и шлема виртуальной реальности. Для отслеживания положения и ориентации головы оператора и его руки целесообразно использовать системы трекинга.

С помощью такой технологии можно реализовать виртуальную комнату с пультами управления и приборами, расположенными на стенах. Оператор будет иметь возможность перемещаться по комнате и воздействовать на элементы виртуальных пультов. В случае использования современных компьютерных перчаток обеспечиваются также тактильные ощущения от таких воздействий.

Данная работа выполняется в рамках Программы фундаментальных исследований ОНИТ 1, проект № 2.9.

Список литературы

1. **Михайлюк М. В.** Тренировочный комплекс операторов робототехнических средств // Сб. тр. 8-й Междунар. науч.-практ. конф. "Современные информационные технологии управления экологической безопасностью, природопользованием, действиями в чрезвычайных ситуациях". 7—11 сентября 2009. Пос. Рыбачье, Крым. Киев: АДЕФ-Украина, 2009. С. 171—176.

2. **Михайлюк М. В.** Тренировочный комплекс операторов робототехнических средств по ликвидации последствий чрезвычайных ситуаций // Сб. статей "Фундаментальные проблемы системной безопасности". Вып. 2. М.: Вузовская книга, 2010. С. 441—444.

3. **Новые** методы управления сложными системами. М.: Наука, 2004. 336 с.

4. **Михайлюк М. В., Трушин А. М.** Видео тренажерные системы управления космическими роботами и манипуляторами // Космический форум 2011, посвященный 50-летию полета в космос Ю. А. Гагарина (сборник материалов). М.: ИМБП РАН, 2011. С. 74.

5. **Михайлюк М. В., Торгашев М. А.** Система "GLView" визуализации для моделирующих комплексов и систем виртуальной реальности // Вестник РАЕН. 2011. Т. 11. № 2. С. 20—28.

ИНФОРМАЦИЯ

23 – 24 октября 2014 г. в Москве, в центре Digital October пройдет Десятая юбилейная конференция

"Разработка ПО/CEE-SECR 2014"

За годы существования SECR заслуженно стал одним из важнейших событий ИТ-индустрии. Более 900 специалистов соберутся на конференцию в этом году: от программистов и представителей академического сообщества до предпринимателей и инвесторов. Участников ждут выступления экспертов мирового уровня, мастер-классы, дискуссии, общение со спикерами и многое другое.

Во время конференции выступления пройдут одновременно в несколько потоков, что даст участникам возможность выбирать наиболее интересную для себя тему в каждый момент.

Среди уже известных приглашенных спикеров:

Мики В. Мантл — автор книги "Управляя неуправляемым: правила, инструменты и идеи по управлению людьми и командами в разработке ПО".

За сорок лет своей карьеры Мики работал над многими известнейшими программными продуктами и управлял крупными международными командами в таких компаниях, как Pixar, SONY/Gracenote и Broderbund Software.

Дино Эспозито — технический евангелист JetBrains; автор популярных книг для .NET-разработчиков и архитекторов, опубликованных Microsoft Press. Тренер с большим опытом работы и первоклассный консультант.

В своем докладе "Never mind the Mobile Web; Here's the Device Web" Дино развенчает некоторые мифы адаптивного веб-дизайна и поднимет вопрос о действительно разных способах отображения HTML.

Приглашаем Вас:

- Подать доклад — поделитесь своим уникальным опытом с коллегами и выступите на одной сцене с признанными лидерами мировой и российской индустрии разработки ПО. Авторы принятых работ участвуют в конференции бесплатно.
- Принять участие — этот SECR станет юбилейным, а значит наиболее ярким и интересным за всю историю конференции. Успейте зарегистрироваться со скидкой! Действуют специальные предложения группам, студентам.

Более подробная информация на сайте www.secr.ru

УДК 004.5

Я. А. Туровский, канд. мед. наук, доц., зав. лаб., Воронежский государственный университет,
e-mail: yaroslav_turovsk@mail.ru

Оптимизация работы синхронного нейрокомпьютерного интерфейса на основе селекции каналов электроэнцефалограммы

Представлены методы, обеспечивающие обучение программно-аппаратной части нейрокомпьютерных интерфейсов (НКИ) для проведения выбора из множества электродов подмножества, формирующего наилучшее отношение сигнал/шум для последующего пространственного и временного накопления сигнала. Эти методы предложены для трех типов синхронных НКИ: основанных на зрительных вызванных потенциалах, устойчивых зрительных вызванных потенциалах и когнитивных вызванных потенциалах с компонентом P300. Предложен алгоритм построения специализированного фильтра для оценки вызванных потенциалов на основе анализа цепочек локальных максимумов и минимумов в матрице квадратов коэффициентов вейвлет-преобразования. Предложенный подход позволяет существенно упростить обучение систем НКИ в случае изменения положения регистрирующих электродов и тем самым увеличить функциональные возможности синхронных НКИ.

Ключевые слова: *электроэнцефалограмма, нейро-компьютерный интерфейс, вызванные потенциалы*

Ya. A. Turovsky

The Optimization of the Synchronous Brain-Computer Interfaces on Electroencephalogram Canal Selection Based

The paper presents methods of training brain-computer interfaces (BCI) firmware to select a subset of electrodes from an aggregate, so that this subset forms the best signal/noise ratio for the subsequent spatial and temporal signal accumulation. These methods are proposed for three kinds of synchronous BCI: based on visual evoked potentials, sustainable visual evoked potentials and cognitive evoked potentials with P300 component. An algorithm offered to construct wavelet transformations of a specialized filter for evaluation of evoked potentials on the basis of analysis of the chains of local maxima and minima in the matrix of squares of the coefficients. The proposed approach allows to significantly simplify training system for BCI in case of position change of the recording electrodes, and, thereby, to increase the functionality of synchronous BCI.

Keywords: *electroencephalogram, wavelet analysis, chain of the local maximum, local spectrum*

Введение

Значительный интерес, проявившийся в последние десятилетия к системам биологической обратной связи, благодаря совершенствованию вычислительной техники привел к появлению новых каналов коммуникации между человеком и компьютером. Одним из таких каналов является нейрокомпьютерный интерфейс (НКИ). В настоящее время в мире выполняется значительный объем работ в разных направлениях, так или иначе ассоциированных с системами передачи данных напрямую между мозгом и компьютером.

Одним из таких направлений является совершенствование систем НКИ, в котором можно выделить несколько главных тем: увеличение скорости работы интерфейса, повышение точности работы интерфейса, снижение времени на обучение пользователя работе с интерфейсом и упрощение самого процесса обучения. При этом обучение НКИ как и любой другой эргатической системы заключается во взаимной адаптации человека-оператора и программно-аппаратной части системы. Особенность последнего пункта в том, что по сути обучение программно-аппаратной части НКИ (или интерфейса мозг-компьютер — ИМК) хотя и различается для синхронных, т. е. основанных на вызванных потенциалах головного мозга, и асинхронных типов интерфейсов, должно учитывать возможности разной локализации считывающих электродов (в случае наиболее распространенного подхода к НКИ, основанного на анализе электроэнцефалографических сигналов). Основной причиной разной локализации электродов является то, что электродные системы, интегрированные в специальные шапочки для регистрации электроэнцефалограмм (ЭЭГ), при повторном надевании не обеспечивают точной локализации электродов с теми же значениями координат, которые были в предшествующих экспериментах. Безусловно, чувствительность разных типов НКИ к таким изменениям локализации датчиков является разной. Однако приходится констатировать, что это в целом оказывает негативное влияние на точность работы интерфейсов.

В свете изложенного выше актуальной представляется разработка подходов, обеспечивающих автоматизированный расчет и выделение из группы электродов подгрупп, регистрирующих электрофизиологические процессы, представляющие наиболее важную информацию для детектирования выбранных пользователем команд.

Целью работы, результаты которой представлены далее, является разработка методов и создание на их основе алгоритмов селекции рабочих электродов для систем биологической обратной связи и нейрокомпьютерных синхронных интерфейсов.

Особенности выделения вызванных потенциалов головного мозга, используемых в синхронных системах НКИ

Как известно [1], в синхронных НКИ именно программно-аппаратная часть инициирует работу всей системы. Тем или иным путем испытуемому подается сигнал, после чего он должен перейти в определенное ментальное состояние и тем самым изменить электрическую активность мозга. Традиционно, лидирующие позиции по скорости и точности среди синхронных интерфейсов занимают интерфейсы, использующие устойчивые зрительные вызванные потенциалы (УЗВП, SSVEP) или потенциал P300. Помимо этих двух потенциалов используют и зрительные вызванные потенциалы [1]. На рис. 1 продемонстрированы варианты данных вызванных потенциалов (ВП): *а* — зрительный вызванный потенциал (ЗВП) ЭЭГ, частота дискретизации 5 кГц, отведение Oz [2, 3], число суммаций — 15, частота стимуляции 1 Гц; *б* — когнитивный потенциал ЭЭГ с компонентом P300, частота дискретизации 5 кГц, отведение Pz [2, 3], число суммаций — 15, незначимых стимулов — 7, значимый стимул — 1; *в* — УЗВП ЭЭГ, частота дискретизации 5 кГц, отведение Oz, число суммаций — 15, частота стимуляции 9,009 Гц.

Как известно, амплитуда фоновой ЭЭГ существенно превосходит амплитуду (ВП) [2, 3]. Таким образом, для получения пригодного для обработки ВП необходимо определенное время, требующееся для накопления нужного числа повторений реакции ЭЭГ на стимул и улучшения отношения сигнал/шум до приемлемых для дальнейшей обработки значений. Изменение отношения сигнал/шум по мере увеличения числа накоплений сигнала можно описать как в работе [3]:

$$(A_s/A_g)_n = (A_s/A_g)_1 \sqrt{n}\,, \qquad (1)$$

где A_s — амплитуда полезного сигнала (в нашем случае — потенциала SSVEP); A_g — амплитуда шумового компонента в сигнале; n — число суммаций. Индексом 1 обозначено отношение амплитуд в единичном сигнале, а индексом n — отношение амплитуд в сигнале после n накоплений.

При этом возможны два подхода к выделению ВП: когерентное накопление сигнала только с учетом повторяющихся последовательностей ЭЭГ, т. е. суммация по времени, например, для одного канала регистрации, и использование одновременно несколько относительно близко расположенных каналов — так называемая пространственная суммация. Очевидно, что, согласно формуле (1), совместная пространственная и временная суммация серии из нескольких одинаковых стимулов даст лучшее отношение сигнал/шум, чем только лишь суммация такого же числа стимулов по одному каналу.

В то же время есть риск, что некоторые из каналов не будут содержать полезного сигнала или полезный сигнал будет иметь крайне незначительную амплитуду. В этой ситуации уровень шума значительно вырастет и тем самым сведет на нет положительные эффекты от многоканальной суммации. Важно отметить, что для разных вызванных потенциалов, используемых в НКИ, применяют разные подходы к детекции. Так, УЗВП (рис. 2, *в*) представляет собой колебания, близкие к синусоидальным, и сохраняющиеся в течение всего времени, пока пользователь смотрит на фотостимулятор, в роли которого может быть как фотодиод, так и изменяющееся изображение на экране монитора [4].

При этом частота колебаний УЗВП совпадает с частотой стимулятора. Следовательно, для выделения нужной информации из уже накопленного потенциала необходим спектральный анализ. Наиболее часто

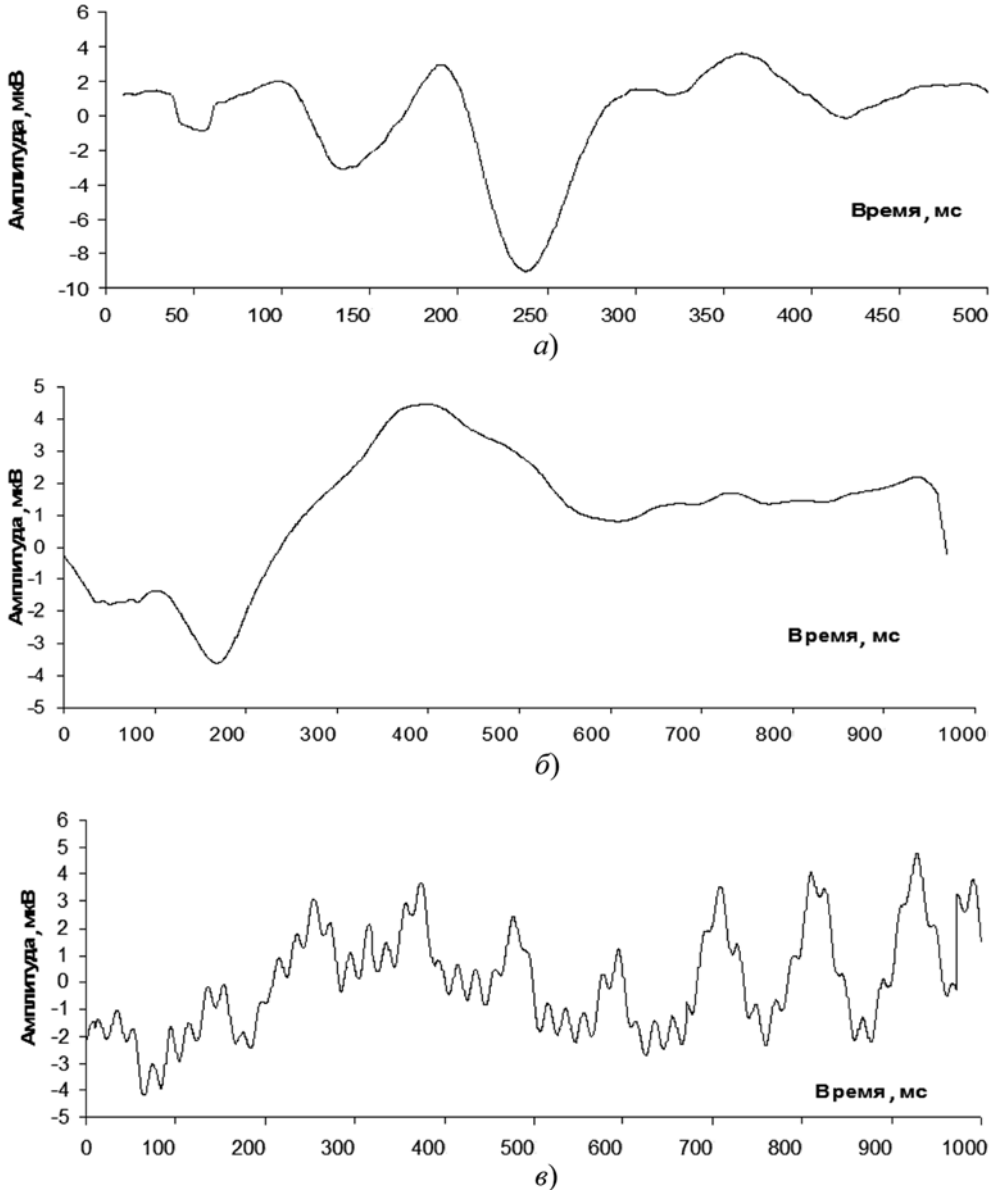


Рис. 1. Типы ВП, анализ которых использовался в синхронных НКИ

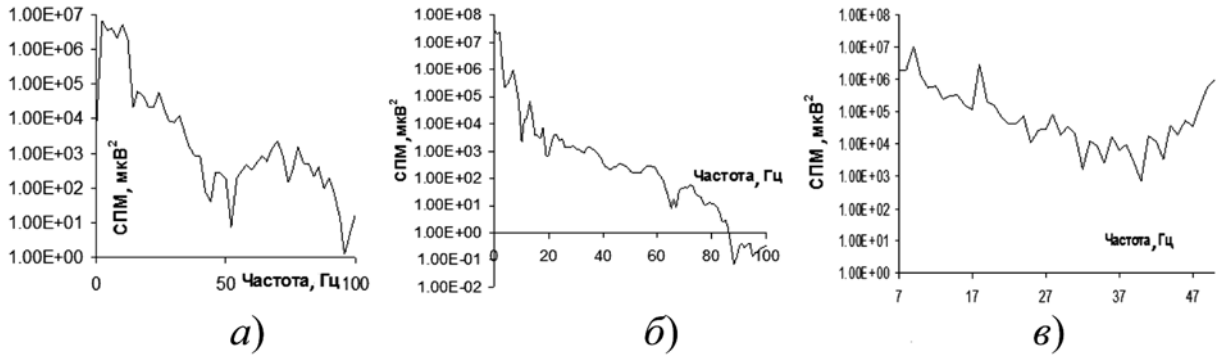


Рис. 2. Показатели спектральной плотности мощности (СПМ), рассчитанные на основе преобразования Фурье для различных ВП

для этого применяют преобразование Фурье в различных его модификациях [1, 4]. В качестве примера на рис. 2 представлены спектры Фурье для каждого из ВП (рис. 1): *а* — ЗВП; *б* — когнитивный ВП с потенциалом P300; *в* — УЗВП (SSVEP).

В то же время анализ команд пользователя на устройства-эффекторы при использовании в НКИ ЗВП или потенциала P300 в значительной мере связан именно с выделением амплитудных, а не частотных феноменов.

Выделение подмножества рабочих электродов для синхронных НКИ

Рассмотрим варианты выделения рабочих электродов для УЗВП. В этом случае необходимо использовать обучение, заключающееся в том,что пользователю предлагается в течение определенного времени внимательно смотреть на один из стимулирующих фотодиодов (либо на изменяющееся с определенной частотой изображение на экране монитора). При этом определяется, какие именно электроды, расположенные в проекции затылочной коры, дают наибольший вклад в формирование УЗВП заданной частоты. Естественно, что данный подход можно отнести к "обучению с учителем". После проведения обучения выполняют контрольный эксперимент, когда пользователь по команде программно-аппаратной части НКИ смотрит на другой стимулятор, работающий, соответственно, на другой частоте (рис. 3). В случае успешной детекции выбор электродов завершается, в противном случае цикл повторяется.

Для формализации вышесказанного определим множество $E = \{e_j(x_j, y_j, z_j)\}$ электродов, размещаемых на поверхности скальпа испытуемого. Здесь $x, y, z \in \mathbf{R}$ — координаты положения электродов; j — порядковый номер электрода. При этом начало системы координат может быть выбрано, исходя из целей эксперимента, в весьма широких пределах [5].

Следует учитывать, что локализация электродов в определенном участке скальпа позволит формализовать его положение по системам 10—10 или 10—20. Например, можно записать:

$$\exists e_j((x_j \in (x_{MaxCz}, x_{MinCz})) \wedge ((y_j \in (y_{MaxCz}, y_{MinCz})) \wedge \\ \wedge ((z_j \in (z_{MaxCz}, z_{MinCz})) \Rightarrow (e_j = e_{Cz}),$$

где x_{MaxCz} — максимальное значение координаты x для электрода Cz по системе 10—20, а x_{MinCz} — минимальное значение координаты x для электрода Cz по этой же системе. Аналогичные индексы вводятся и для координат y и z.

Подвергнем результаты когерентного накопления данных ЭЭГ из множества E электродов спектрально-

Рис. 3. Схема процесса селекции электродов для оптимизации работы НКИ на основе УЗВП

му оцениванию с использованием преобразования Фурье или вейвлет-анализа [6—8]. Введем логическую функцию $H(V)$. В случае, если значения спектральной плотности мощности или ее аналога — вейвлетной плотности мощности в определенном диапазоне превысят заданное пользователем значение, то логическая функция $H(V)$, соответствующая определенному каналу, примет значение 1 (если канал отобран для дальнейшей работы), в противном случае — примет значение 0 (если канал не используется в дальнейшей работе):

$$H(V) = \begin{cases} 1, & V(a, b) \geqslant T_V \\ 0, & V(a, b) < T_V, \end{cases} \qquad (2)$$

где T_V — пороговое значение, задаваемое пользователем для значений вейвлетной плотности мощности, а $V(a, b)$ представляет собой скейлограмму — аналог спектральной плотности мощности для матрицы вейвлет-преобразования [9]; a и b — параметры масштаба и времени для вейвлет-преобразования; $a, b \in \mathbf{R}$; $a > 0$.

Аналогично для преобразования Фурье вводится логическая функция $H(F)$:

$$H(F) = \begin{cases} 1, & \sum\limits_{k_{\min}}^{k_{\max}} F^2(k) \geqslant T_F; \\ 0, & \sum\limits_{k_{\min}}^{k_{\max}} F^2(k) < T_F, \end{cases}$$

где T_F — пороговое значение, задаваемое пользователем для спектральной плотности мощности; F^2 — спектральная плотность мощности, рассчитанная для диапазона от $k_{\min}$ до $k_{\max}$; k — частота сигнала.

Рассмотрим теперь селекцию электродов, когда ВП анализируется, в первую очередь, по амплитудному компоненту, как это делается в случае интерфейса, основанного на ЗВП или компоненте Р300. При этом необходимо задать временной диапазон, в котором будет происходить анализ ВП. Действительно, для компонента Р300 анализ должен проводиться в диапазоне значений ВП с латентным временем ~300 мс, тогда как более ранние потенциалы не всегда могут нести информацию о выборе пользователем того или иного стимула. В случае же ЗВП наоборот, более длительный временной отрезок анализа ВП может привести к попаданию в анализируемую область разряда последействия [3]. Таким образом, простейшее решение выбора нужного фрагмента ВП заключается в том, что после накопления исследуемого ВП выделяется отрезок времени, в течение которого происходит суммация, например, модулей мгновенных амплитуд ВП. В этом случае опять же необходимо провести обучение программы для выделения значимых ВП.

Алгоритм обучения будет аналогичен таковому для случая УЗВП: анализ ЭЭГ с выделением ВП для известного выбора пользователя; определение подмножества каналов, дающих наиболее выраженный ответ; тестирование подмножества канала с учетом пространственной суммации и в случае успеха сохранение имеющегося подмножества каналов в качестве рабочего для интерфейса.

Однако данный алгоритм может иметь в качестве входных параметров не только сумму мгновенных амплитуд ВП в заданном временном диапазоне, но и другие показатели, отражающие амплитудно-частотную характеристику ВП. К таким показателям можно отнести значения вейвлетной плотности мощности $W^2(a, b)$, рассчитанные в заданном диапазоне масштабов a (частот) и времени b на матрице вейвлет-коэффициентов. В этом случае наличие ВП определяют как превышение показателем $V(a, b)$ заданного порога (2).

Возможен и третий вариант. В его основу положена идея о формировании фильтров, максимально похожих на искомый фрагмент сигнала. Пусть существует ВП $Q(t)$, при этом отрезок времени, в течение которого наблюдаются феномены, на основе которых системы НКИ принимают решения о выборе пользователем тех или иных команд, находится в диапазоне от t_1 до t_2. Проведем непрерывное вейвлет-преобразование на отрезке от t_0 до $t_{\max}$, т. е. в течение времени накопления всего ВП:

$$W(a, b) = 1/\sqrt{a} \int\limits_{t_0}^{t_{\max}} Q(t)\Psi\left(\frac{t-b}{a}\right)dt,$$

где $Q(t)$ — анализируемые данные, зависящие от времени t; Ψ — вейвлет.

Используя метод анализа цепочек локальных максимумов, представленный в работах [8—10], определим границы фрагмента ВП, содержащего цепочки локальных максимумов вейвлет-спектров (скейлограмм) и ограниченные цепочками локальных минимумов во временном диапазоне от t_1 до t_2. Обнулим все значения матрицы $V(a, b)$ за пределами выделенной частотно-временной области. Соотнесем координаты матрицы $V(a, b)$ с координатами матрицы $W(a, b)$, при этом координаты (a, b) у $\mathbf{V}$ и $\mathbf{W}$ будут совпадать. Таким образом, с использованием цепочек локальных максимумов или минимумов, рассчитанных на основе набора локальных вейвлет-спектров (скейлограмм) $V(a, b)$, получаем координаты для обратного вейвлет-преобразования:

$$Q_f(t) = 1/C_\psi \iint W(a, b)\Psi(a, b)dadb/a^2,$$

где $Q_f(t)$ — восстановленный элемент ВП; C_ψ — нормировочный коэффициент.

Полученные значения $Q_f(t)$ подвергают дальнейшей нормировке для получения нулевого среднего. Таким образом, получается цифровой фильтр, содержащий в себе фрагмент ВП, необходимый для корректной работы НКИ. Осуществив свертку нового зарегистрированного ВП этим фильтром, получим значение функции $H(\tau)$, на основе которого программная

часть НКИ принимает решение о выборе пользователем той или иной команды:

$$H(\tau) = \int Q_f(t)Q(t-\tau)dt.$$

Каналы, имеющие наибольшие значения функции $H(\tau)$, и формируют в дальнейшем подмножество каналов, с которыми ведется работа программно-аппаратной частью НКИ по выделению ВП.

Таким образом, указанным выше способом проводится селекция каналов ЭЭГ для работы в системе НКИ непосредственно самой программно-аппаратной частью интерфейса.

Заключение

В работе представлены методы, позволяющие проводить обучение программно-аппаратной части нейрокомпьютерных интерфейсов для селекции из множества электродов подмножества, обеспечивающего наибольшее отношение сигнал/шум для последующего пространственного и временного накопления. Представлены методы для трех типов синхронных НКИ: основанных на зрительных вызванных потенциалах, устойчивых зрительных вызванных потенциалах и когнитивных вызванных потенциалах с компонентом Р300. На основе анализа цепочек локальных максимумов и минимумов на матрице квадратов коэффициентов вейвлет-преобразования предложен алгоритм построения специализированного фильтра для оценки вызванных потенциалов. Заявленный подход позволяет существенно упростить обучение систем НКИ в случае изменения положения регистрирующих электродов и тем самым улучшить функциональные возможности синхронных нейрокомпьютерных интерфейсов.

Автор выражает глубокую признательность С. Д. Кургалину, А. А. Вахтину, С. В. Борзунову за неоценимую помощь в проведении исследования.

Список литературы

1. **Nicolas-Alonso L. F., Gomez-Gil J.** Brain Computer Interfaces // Sensors. 2012. V. 12. P. 1211—1279; doi:10.3390/s120201211.

2. **Зенков Л. Р.** Клиническая электроэнцефалография (с элементами эпилептологии). М.: МЕДПресс-информ, 2011. 356 с.

3. **Гнездицкий В. В.** Вызванные потенциалы мозга в клинической практике. М.: МЕДПресс-информ, 2003. 264 с.

4. **Garcia G.** High frequency SSVEPs for BCI applications // Computer-Human Interaction, April 2008, Florence, Italy. 2008. URL: http://hmi.ewi.utwente.nl/chi2008/chi2008_files/garcia.pdf

5. **Гнездицкий В. В.** Обратная задача ЭЭГ и клиническая электроэнцефалография. М.: МЕДПресс-информ, 2004. 626 с.

6. **Добеши И.** Десять лекций по вейвлетам. Ижевск: НИЦ Регулярная и хаотическая динамика, 2001. 464 с.

7. **Астафьева Н. М.** Вейвлет-анализ: основы теории и примеры применения // Успехи физических наук. 1996. Т. 166, № 11. С. 1145—1170.

8. **Туровский Я. А., Кургалин С. Д., Семенов А. Г.** Исследование динамики максимумов локальных вейвлет-спектров вызванных зрительных потенциалов головного мозга // Информационные технологии. 2013. № 10. С. 46—50.

9. **Туровский Я. А., Кургалин С. Д., Вахтин А. А.** Обработка данных ЭЭГ на основе анализа частотных зависимостей и вейвлет-преобразования // Биомедицинская радиоэлектроника. 2012. № 12. С. 39—45.

10. **Туровский Я. А.** Программа PikWave 1.0. Зарегистрирована в Российском агентстве по патентам и товарным знакам, регистрационный № 2006613500.

ИНФОРМАЦИЯ

В рамках 12-й Международной конференции по численному анализу и прикладной математике (12^{th} International Conference of Numerical Analysis and Applied Mathematics — ICNAAM 2014), которая состоится на острове Родос (Греция) с 22 по 28 сентября 2014 г., будет проведена специальная сессия "Прикладные проблемы теории вероятностей и математической статистики, относящиеся к моделированию информационных систем" (Applied Problems in Probability Theory and Mathematical Statistics Related to Modeling of Information Systems — APPT+MS).

Тематика специальной сессии:

- Применение теории вероятностей в моделировании информационных систем
- Применение случайных процессов в моделировании информационных систем
- Применение математической статистики в моделировании информационных систем
- Теория массового обслуживания
- Дискретные вероятностные модели
- Модели информационной безопасности
- Современные задачи теории телетрафика

Сайт конференции http://www.icnaam.org/

УДК 004.89

М. А. Колосовский, аспирант, Алтайский государственный технический университет им. И. И. Ползунова, г. Барнаул,
e-mail: maxim.astu@gmail.com

Трекинг пешеходов в задаче видеонаблюдения за нерегулируемыми пешеходными переходами*

Статья посвящена решению задачи построения системы видеонаблюдения за пешеходными переходами. Приведен анализ особенностей этой задачи, представлена архитектура системы видеонаблюдения, описаны подсистема трекинга пешеходов и эксперименты, измеряющие качество, скорость и устойчивость системы к помехам. Эксперименты показали точность сопровождения пешеходов около 92 %, полноту 75...95 % и скорость, близкую к режиму реального времени.

Ключевые слова: *видеонаблюдение, трекинг объектов, обнаружение объектов, компьютерное зрение*

M. A. Kolosovskiy

Tracking Pedestrians for Video Surveillance for Non-Signalized Pedestrian Crossings

This paper considers the task of video surveillance for pedestrian crossings. We analyzed peculiarities of the task, proposed the structure of the video surveillance system, and described the developed tracking subsystem. We evaluated accuracy, performance and robustness of the system. The precision of tracking is about 92 %, recall — 75...95 % and its speed is close to real time mode.

Keywords: *video surveillance, object tracking, object detection, computer vision*

Введение

Представленные в статье результаты исследований и практических работ направлены на создание системы видеонаблюдения за пешеходными переходами для контроля соблюдения водителями правил дорожного движения. Благодаря широкому кругу решаемых с использованием систем видеонаблюдения за транспортом задач число таких систем неуклонно растет. Ключевой частью этих систем являются алгоритмы компьютерного зрения, способные стабильно отслеживать на видео участников движения. Для решения задачи видеофиксации нарушений на нерегулируемых пешеходных переходах требуются более сложные алгоритмы компьютерного зрения, нежели алгоритмы для таких задач, как контроль скоростного режима, обнаружение пересечений сплошной линии разметки или контроль ситуаций на регулируемом перекрестке. Видимо по этой причине системы для нерегулируемых переходов не представлены должным образом на российском и международных рынках.

В данной статье рассматривается часть системы видеонаблюдения, отвечающая за сопровождение (*трекинг*) обнаруженных пешеходов, способная не только отследить их перемещение, но и непрерывно обновлять список сопровождаемых объектов и восстанавливать слежение даже после ошибок системы. Составные части системы и их параметры подобраны на основе анализа особенностей задачи и оценки возможности применения существующих подходов для ее решения. Эксперименты показали точность обнаружения около 92 %, полноту — 75...95 % (в зависимости от плотности потока пешеходов) и скорость работы, близкую к режиму реального времени.

Работа такой системы осложняется следующими факторами:

- частое перекрытие объектов друг другом, что приводит к частым потерям трекером своих целей;

* Работа выполнена при поддержке Фонда содействия развитию малых форм предприятий в научно-технической сфере (госконтракт № 0068 ГУ1/2013).

- погодные условия и шумы камеры, мешающие точному извлечению признаков сопровождаемых объектов;
- сложный фон, затрудняющий отделение объектов от фона;
- неравномерность освещения, изменяющая параметры признаков объектов в процессе движения по переходу;
- необходимость анализа видео в режиме реального времени;
- необходимость настройки параметров алгоритма под условия съемки (освещенность, размер объектов и др.) на каждой новой точке наблюдения.

К настоящему времени разработаны десятки алгоритмов сопровождения объектов, снятых на видео. Однако ни один из них не показывает стабильно хороших результатов при решении рассматриваемой задачи. По этой причине при построении систем видеонаблюдения необходимо адаптировать алгоритмы под решение конкретной задачи [1]. С этой целью был выявлен ряд особенностей, которые были учтены при выборе алгоритма и его адаптации. К их числу относятся следующие:

- траектории движения объектов подчинены ряду простых шаблонов, что следует использовать для прогнозирования положения объектов при потере сопровождения после перекрытия;
- объекты в процессе движения не наклоняются и не вращаются (т. е. видны с одной и той же стороны), поза объектов меняется незначительно, что уменьшает изменение модели внешности объекта (*appearance model*) во время слежения;
- камера и фон неподвижны, что облегчает отделение признаков объекта от признаков фона;
- небольшое и, как правило, постоянное смещение объектов в процессе движения, что позволяет сузить пространство поиска при локализации объектов трекером;
- объекты имеют незначительную внутриклассовую изменчивость формы, что позволяет сделать более точную настройку алгоритмов;
- небольшая вариативность размера объектов, что устраняет необходимость использования пирамиды изображений;
- объекты исчезают только в определенных местах кадра, что позволяет отделить ложную потерю объекта от его исчезновения.

Далее представлены обзор подобных систем видеонаблюдения и их сравнение с представляемой системой; общая структура этой системы и занимаемое в ней место подсистемы трекинга; описание этой подсистемы; результаты проведенных экспериментов.

Обзор подобных работ

Системы обнаружения и сопровождения можно разделить на две группы в соответствии с их архитектурой. Работа систем первой группы заключается в последовательном выполнении следующих процедур: выделение областей активности (*блобов*); трекинг областей; распознавание областей [2, 3]. Этап распознавания может отсутствовать, если заранее известно, что все перемещения в кадре вызваны только объектами интереса [3]. Если все-таки распознавание предполагается, то оно заключается в оценке различных признаков блоба, таких как площадь и форма. Кроме того, указанным шагам может предшествовать этап подготовки изображения, например, через усиление границ [3].

Системы второй группы основаны на обнаружении объектов интереса при помощи детектора и последующем их сопровождении при помощи трекеров [4, 5]. Как и системы первой группы, такие системы зачастую перед детектированием выделяют активные области. Представляемая в данной статье система относится к этой группе. Выбор такого подхода обусловлен сложностью распознавания пешеходов по соответствующим блобам вследствие сильных шумов камеры, вызывающих сильные искажения и разрывы блобов. Предлагаемая в настоящей работе система в отличие от других позволяет восстанавливать потерянные объекты и добавлять вновь появившиеся объекты. Кроме того, конкретный детектор активности, детектор объектов и трекер выбраны с учетом особенностей поставленной задачи.

К системам второй группы можно также отнести системы, основанные только на детекторах объектов. При такой архитектуре периодически запускается детектор объектов и следы объектов на разных кадрах ставятся в соответствие друг другу, формируя траектории объектов [6]. Однако постоянные запуски детектора объектов не всегда возможны в силу его вычислительной сложности, что позволяет применять такую архитектуру только в случае, когда используется достаточно быстрый детектор.

Структура системы видеонаблюдения

Обнаружение пешеходов осуществляется при помощи специального алгоритма, называемого *детектором*. Как правило, детекторы вычислительно слишком сложны, чтобы запускаться на каждом кадре, поэтому после обнаружения объектов интереса они сопровождаются более простым с вычислительной точки зрения алгоритмом, называемым *трекером*. Однако ввиду непрерывного процесса появления новых объектов интереса, возникает необходимость периодически запускать детектор. Соответственно, нужен какой-то механизм, задающий, насколько часто и на каких областях кадра надо запускать детектор, чтобы минимизировать вычислительные затраты, но иметь при этом возможность обнаруживать новые объекты. Кроме этого, ошибки детекторов (пропуск объекта и ложное срабатывание) и ошибки трекеров (соскальзывание с объекта и потеря объекта) вынуждают дополнять этот механизм возможностью восстановления после таких ошибок.

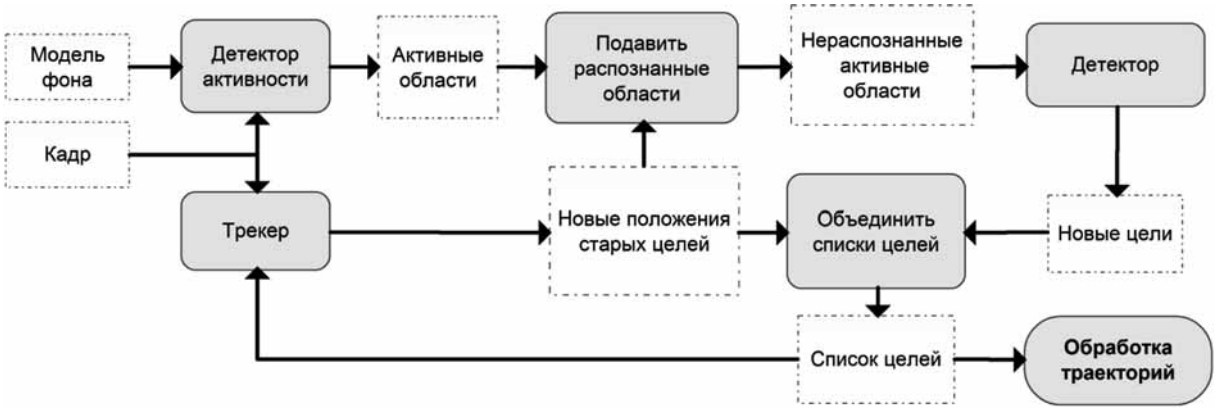


Рис. 1. Схема работы системы обнаружения и сопровождения пешеходов

Рис. 2. Области активности и уже распознанный и ведомый трекером объект (черный прямоугольник)

Рис. 3. Области активности под распознанным объектом подавлены. Детектору передается только регион вокруг нераспознанной области (белый прямоугольник)

В связи с изложенными выше причинами был разработан механизм, позволяющий решать перечисленные задачи. Его суть состоит в использовании *детектора активности,* регистрирующего движение в кадре как разность текущего кадра и модели фона с последующим применением морфологических операций. Детектор активности позволяет не только сократить площадь кадра, обрабатываемую детектором, но и восстановить сопровождение объектов, потерянных трекером.

Общая схема работы представляемого механизма показана на рис. 1. Детектор активности обнаруживает области кадра, где зарегистрировано движение. Эта информация объединяется с позициями уже распознанных целей, которые поступают от трекера (рис. 2). Те области активности, где находятся сопровождаемые цели, подавляются, а оставшиеся области специальным образом расширяются [6] и передаются детектору пешеходов (рис. 3). Детектор обнаруживает новые объекты, после чего объединенный список старых и новых целей передается трекеру для сопровождения. Информация о передвижении объектов передается в обработчик траекторий, обнаруживающий нарушения правил дорожного движения. Детальное описание разработанного детектора активности и детектора пешеходов на основе гистограмм ориентированных градиентов представлено в работе [6].

Подсистема трекинга объектов

Ввиду широкого многообразия разработанных к настоящему моменту подходов к трекингу была проанализирована возможность применения различных методов для решения рассматриваемой задачи видео-

Таблица 1

Анализ признаков, используемых для сопровождения объектов и их применимость к рассматриваемой задаче видеонаблюдения

Признаки объекта	Применимость к задаче
Градиенты [10, 11]	Камеры видеонаблюдения генерируют, как правило, сильные шумы, к которым чувствителен данный вид признаков
Контуры (силуэт) [12, 13]	Велика вероятность смешения и потери силуэтов нескольких объектов при пересечении
Края [14, 15]	Направление краев в процессе движения человека может значительно меняться, поэтому состав признаков сопровождаемого объекта будет существенно варьироваться
Оптический поток [16, 17]	Необходима дополнительная сложная обработка для восстановления сопровождения после перекрытия
Особые точки [8, 18, 19]	Восстановление после перекрытия объектов значительно легче, чем при использовании других видов признаков, особенно в условиях хорошей прогнозируемости траектории движения
Сегментация [20, 21]	Необходимо, чтобы фон и объекты имели простое цветовое содержание (один цвет, нет сложных текстур) и контрастировали друг с другом [24], что совершенно не соответствует рассматриваемой задаче
Цвет [22, 23]	Сопровождение легко восстанавливается после перекрытий, однако подавляющее большинство камер видеонаблюдения черно-белые, а градации серого дают значительно меньше информации, поэтому этот подход будет уже не так эффективен

Таблица 2.

Анализ производительности отобранных реализаций KLT-трекера (кадр 640 × 480, 50 точек)

Реализация KLT-трекера	Язык	Время, с	Кадров/с
LK_Tracker [25]	Matlab	0,1...0,15	7...10
stb [26]	C	0,03...0,04	25...33
OpenCV [27]	C/C++	0,01...0,02	50...100

наблюдения. Были рассмотрены основные аспекты, определяющие трекинг объектов интереса [7], а именно:

- признаки сопровождаемого объекта (цвет, особые точки, градиенты и др.);
- модель представления сопровождаемого объекта (точка, прямоугольник, эллипс, "скелет", деформируемая модель и др.; шаблон или гистограмма);
- механизм локализации целей на последующих кадрах (перебор по окрестности, фильтр Калмана, фильтр частиц, ядерная локализация и др.);
- метод сопоставления следов объектов на разных кадрах (метод ближайшего соседа, минимальное паросочетание и др.; двух- и многокадровые алгоритмы).

Принципиально важно правильно выбрать признаки, используемые для сопровождения объектов. В табл. 1 представлены результаты анализа возможности применения различных признаков объекта, используемых для сопровождения, к задаче видеонаблюдения за пешеходными переходами. На основе проведенного анализа сделан вывод, что наиболее подходящим вариантом является использование особых точек. Особые точки выбирают как резкие изменения градиента сразу в нескольких направлениях (углы) вследствие того, что перемещение таких точек гораздо удобнее отслеживать, чем передвижение гладких поверхностей. На настоящее время выбран один из базовых трекеров на основе особых точек, а именно Kanade-Lucas-Tomasi (KLT-трекер). Такой выбор определил использование метода поиска особых точек [8] и алгоритм локализации [9], предполагающий небольшое смещение и неизменность цвета точки в процессе движения.

В качестве модели представления признаков был выбран прямоугольник. На основе экспериментов было обнаружено, что особые точки, относящиеся к нижней части пешехода, более склонны терять свои положения на объекте. По этой причине для сопровождения используют только верхнюю часть пешеходов. Кроме того, была подобрана ширина окна, в котором ищут особые точки с тем, чтобы минимизировать получение точек, относящихся к фону или к другим объектам.

Был проведен анализ доступных реализаций KLT-трекера. Учет возможностей аппаратного обеспечения вычислительных блоков видеофиксаторов обусловил невозможность использования ряда реализаций на основе GPU и Java. После этого были проведены испытания производительности оставшихся реализаций (табл. 2). Измерялось время работы трекера на одном кадре размером 640 × 480 при локализации 50 особых точек, а также было подсчитано число кадров, которое при таком времени работы можно обработать. Результаты показали, что наибольшую эффективность дает реализация, включенная в библиотеку OpenCV, которую и планируется использовать на практике.

В ходе экспериментов было выяснено, что основной трудностью, которая появляется при использовании особых точек, является смещение точек со своих мест на объектах: точки либо перемещаются по объекту, либо переходят с одного объекта на другие, или на фон. Рис. 4 демонстрирует накопление ошибки в

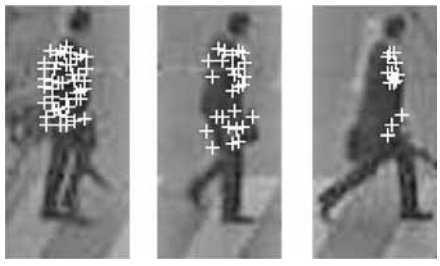


Рис. 4. Изменение набора точек в процессе трекинга

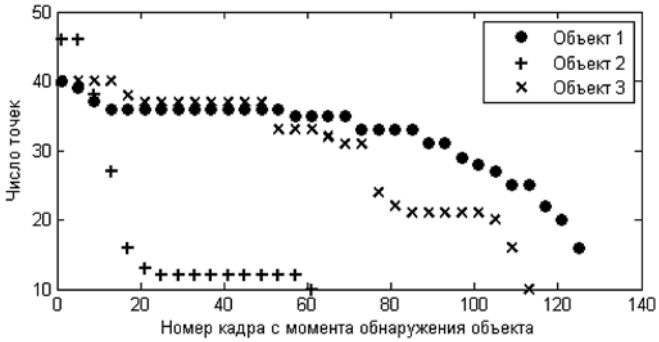


Рис. 5. Пример изменения числа точек в процессе трекинга для разных объектов. Точки на гладких объектах (например, объект 2) гораздо быстрее теряют свои места и удаляются из набора

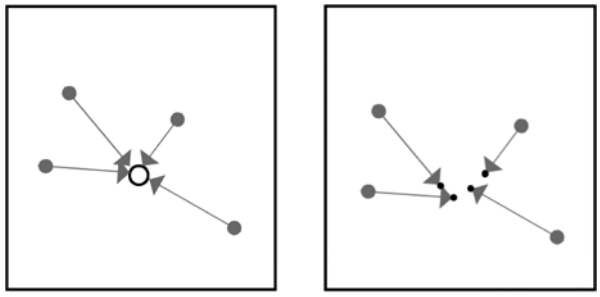


Рис. 6. Определение координат центра объекта по координатам точек:

а — начальное смещение точек относительно центра; *б* — в силу накопления ошибок [в процессе трекинга] точки указывают на разные места как на центр объекта

процессе слежения. Это особенно актуально для очень однородных объектов. Таким образом, требуется механизм фильтрации точек, потерявших свои места на объекте. В связи с этим предложен метод, при котором точки, имеющие смещение ниже или выше заданного порога, удаляют. Слишком малое смещение свидетельствует о том, что точка находится на неподвижном фоне, а слишком большое — об ошибке метода локализации, ведь объект движется с относительно небольшой скоростью без рывков. Когда число точек становится меньше подобранного порога в 10 точек (при обнаружении объекта берут не более 50 точек), то объект считается потерянным. На рис. 5 представлен пример динамики уменьшения числа точек в процессе сопровождения.

По причине нестабильности положения особых точек относительно своих мест на сопровождаемом объекте задача восстановления координат самого объекта не является тривиальной. В данной работе предложен следующий механизм определения координат объекта по координатам точек. При обнаружении объекта запоминают смещения найденных точек относительно центра объекта (рис. 6, *а*). На последующих кадрах каждая из оставшихся в наборе точек по соответствующему смещению дает координаты предполагаемого центра объекта. Точки вследствие ошибок локализации теперь будут "указывать" на разные места кадра (рис. 6, *б*), поэтому берут среднее арифметическое по каждой из координат:

$$\mathbf{d}_i = \mathbf{f}_i^{(0)} - \mathbf{C}^{(0)},$$

$$\mathbf{C}^{(j)} = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{f}_i^{(j)} - \mathbf{d}_i),\ j > 0,$$

где $\mathbf{C}^{(0)}$ — вектор координат центра объекта, обнаруженный детектором; $\mathbf{C}^{(j)}$ — вектор координат центра объекта, восстановленный по точкам на j-м кадре (кадры нумеруют с нуля, начиная с кадра, когда объект был обнаружен); $\mathbf{f}_i^{(j)}$ — вектор координат i-й точки на j-м кадре, полученный от трекера; $\mathbf{d}_i$ — вектор смещения i-й точки относительно центра объекта на нулевом кадре; N — число особых точек. Было также опробовано обновление смещения точек в процессе движения, чтобы адаптироваться к изменениям модели объекта:

$$\mathbf{d}_i^{(j+1)} = (1-\alpha)\mathbf{d}_i^{(j)} + \alpha(\mathbf{f}_i^{(j)} - \mathbf{C}_i^{(j)}),$$

где α — коэффициент скорости обновления модели; $\mathbf{d}_i^{(j)}$ — вектор смещения i-й точки на j-м кадре. Однако это приводит к чрезмерному накоплению ошибок модели и к сильному смещению прогнозируемого центра объекта относительно истинного центра.

Эксперименты

Измерение качества, скорости и устойчивости системы видеонаблюдения за пешеходными переходами проводилось на двух видеофайлах [28], характеристики которых представлены в табл. 3. К файлам прилагалась разметка каждого четвертого кадра, отмечающая пешеходов на этих кадрах.

В качестве метрик трекинга были выбраны следующие показатели:

- точность P (*precision*) — отношение числа правильно обнаруженных пешеходов к общему числу обнаружений, выданных детектором;
- полнота R (*recall*) — отношение числа правильно обнаруженных пешеходов к общему числу отмеченных пешеходов в эталонной разметке;
- F-мера, или мера Ван Ризбергена, представляющая собой среднее гармоническое точности и пол-

Таблица 3

Характеристики данных, используемых для тестирования

Файлы	Продолжительность, с	Всего кадров (размечено кадров)	Отмечено пешеходов
Видео 1	20	601 (101)	147
Видео 2	10	301 (76)	370

Таблица 4

Сравнение качества и времени работы предлагаемой системы и системы сопровождения, основанной только на детекторе

Системы	Видео 1				Видео 2			
	P	*R*	*F*	*T*, с	*P*	*R*	*F*	*T*, с
Предлагаемая система	0,9119	0,9592	0,9350	15,23	0,9288	0,7405	0,8241	17,68
Только детектор [6]	0,9910	0,7687	0,8658	42,46	0,9995	0,6000	0,7500	23,71

ноты, *F*-мера объединяет эти показатели в одно значение;

- время работы алгоритма в секундах (T).

Сравнение предлагаемой системы и системы без трекера. Введение трекера в систему обусловлено вычислительной сложностью процесса детектирования. Рассмотрим, как влияет добавление трекера на качество сопровождения и время работы (табл. 4).

Точность предлагаемой системы несколько ниже вследствие ошибок на краях кадра — выходящий из кадра (видимый не полностью) пешеход не будет обнаружен детектором, однако трекер продолжит "цепляться" за цель и сгенерирует ложное обнаружение объекта. Следует заметить, что такой дефект можно устранить, введя механизм удаления цели в связи с ее выходом из зоны наблюдения (а не только в силу потери точек). Однако такой подход требует реализации алгоритма прогнозирования траектории объектов, что является объектом дальнейших исследований. Можно также заметить, что добавление трекера значительно увеличивает полноту обнаружения, так как теперь объект обнаруживается не только тогда, когда на текущем кадре его обнаружил детектор, но и когда он был обнаружен на предыдущих кадрах и сопровожден трекером. Как и предполагалось, комбинация детектора и трекера гораздо быстрее постоянно работающего детектора.

Производительность. Для оценки производительности измерялось не только общее время работы системы наблюдения, но и отдельных этапов с тем, чтобы понять, какие этапы следует оптимизировать для ускорения работы системы (табл. 5).

Значительную часть времени занимают операции расширения активных областей (функции Matlab *imdilate u imerode*) и построение прямоугольников вокруг этих областей (функция *regionprops*). Следует однако заметить, что эта часть системы имеет существенный потенциал для ускорения за счет применения более специализированных реализаций.

Работа детектора также занимает существенную часть времени. Ускорение самого детектора представляет дополнительные трудности, так как требует улучшения уже хорошо оптимизированной реализации. Однако при этом можно существенно сократить размер данных, передаваемых детектору для обработки. Заметим, что в случае видео с плотным потоком пешеходов работа детектора занимает почти половину времени работы системы. Этот факт объясняется частыми перекрытиями объектов друг другом, которые вызывают потери объектов трекером, что влечет необходимость восстанавливать сопровождение при помощи запусков детектора. Таким образом, необходим механизм восстановления сопровождения объекта без использования детектора, что является одной из задач последующей работы.

Таблица 5

Время работы этапов предлагаемой системы

Операция	Видео 1		Видео 2	
	T, с	*T*, %	*T*, с	*T*, %
Получение активных областей	2,191	11	1,236	6
Подавление распознанных активных областей	0,106	1	0,114	1
Расширение нераспознанных активных областей	6,778	33	3,703	17
Построение прямоугольников вокруг активных областей	4,559	22	4,898	22
Обнаружение объектов внутри прямоугольников (детектор)	5,032	25	9,590	43
Сопровождение обнаруженных объектов (трекер)	1,162	6	2,412	11
Прочие операции	0,543	3	0,267	01
Всего	20,371	100	22,22	100

Устойчивость к шумам камеры. Были проведены эксперименты с добавлением к входным данным гауссовского шума и шума типа "соль и перец". Система видеонаблюдения оказалась достаточно чувствительной к шумам этих типов. Так, например, при добавлении гауссовского шума с нулевым средним и вариацией (*var*) 0,01 система перестает функционировать (нет ни одного обнаружения объекта). После применения сглаживающего фильтра работа системы восстанавливается, но с несколько меньшими показателями точности (табл. 6).

Шумы типа "соль и перец" также могут привести к остановке работы системы: при 5 % зашумленных пикселей (*density*) объекты перестают обнаруживаться. Применение медианного фильтра восстанавливает работу (табл. 7).

Устойчивость к изменениям изображения, вызванным погодными условиями. Был проведен ряд экспериментов, имитирующих погодные условия, затрудняющие работу системы. В качестве тестовых условий

Таблица 6

Влияние гауссовского шума на качество работы системы

Файлы	Фильтр	Без шума			Шум *var* = 0,001			Шум *var* = 0,01		
		P	*R*	*F*	*P*	*R*	*F*	*P*	*R*	*F*
Видео 1	С фильтром	0,98	0,86	0,92	0,98	0,85	0,91	0,95	0,89	0,92
	Без фильтра	0,91	0,96	0,93	0,46	0,21	0,29	Не работает		
Видео 2	С фильтром	1,00	0,68	0,81	1,00	0,65	0,79	0,95	0,72	0,82
	Без фильтра	0,93	0,74	0,82	0,70	0,18	0,29	Не работает		

Таблица 7

Влияние шума типа "соль и перец" на качество работы системы

Файлы	Фильтр	Без шума			Шум *density* = 0,01			Шум *density* = 0,05		
		P	*R*	*F*	*P*	*R*	*F*	*P*	*R*	*F*
Видео 1	С фильтром	0,98	0,95	0,96	0,97	0,95	0,96	0,94	0,95	0,94
	Без фильтра	0,91	0,96	0,93	0,84	0,44	0,57	Не работает		
Видео 2	С фильтром	0,92	0,70	0,80	0,93	0,73	0,82	0,91	0,71	0,80
	Без фильтра	0,93	0,74	0,82	0,95	0,24	0,38	Не работает		

Таблица 8

Влияние тумана на качество работы системы

Файлы	Без тумана			Туман 1			Туман 2			Туман 3		
	P	*R*	*F*	*P*	*R*	*F*	*P*	*R*	*F*	*P*	*R*	*F*
Видео 1	0,91	0,96	0,93	0,94	0,91	0,93	1,00	0,45	0,62	Не работает		
Видео 2	0,93	0,74	0,82	0,92	0,71	0,80	0,98	0,40	0,57	Не работает		

Таблица 9

Влияние неравномерного освещения на качество работы системы

Файлы	Обычное освещение			Освещение 1			Освещение 2			Освещение 3		
	P	*R*	*F*	*P*	*R*	*F*	*P*	*R*	*F*	*P*	*R*	*F*
Видео 1	0,91	0,96	0,93	0,98	0,78	0,87	0,95	0,59	0,72	0,96	0,35	0,51
Видео 2	0,93	0,74	0,82	0,90	0,59	0,72	0,91	0,48	0,63	0,87	0,11	0,20

Таблица 10

Влияние изменений, вызванных дождем, на качество работы системы

Файлы	Без дождя			Дождь 1			Дождь 2			Дождь 3		
	P	*R*	*F*	*P*	*R*	*F*	*P*	*R*	*F*	*P*	*R*	*F*
Видео 1	0,91	0,96	0,93	0,95	0,83	0,89	1,00	0,13	0,23	1,00	0,01	0,01
Видео 2	0,93	0,74	0,82	0,96	0,65	0,77	0,99	0,20	0,33	1,00	0,02	0,03

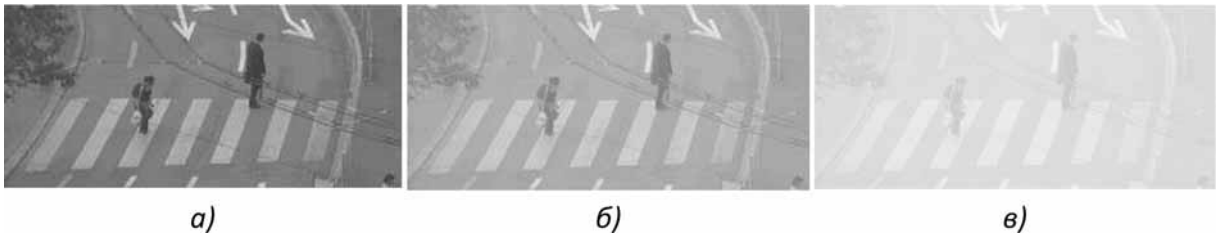


Рис. 7. Демонстрация погодных условий (см. табл. 8):
а — туман 1; *б* — туман 2, *в* — туман 3

Рис. 8. Демонстрация погодных условий (см. табл. 9):
а — освещение 1; *б* — освещение 2; *в* — освещение 3

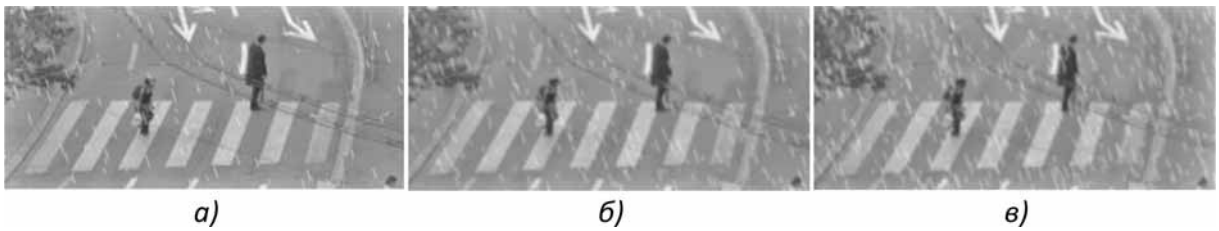


Рис. 9. Демонстрация погодных условий (см. табл. 10):
а — дождь 1; *б* — дождь 2; *в* — дождь 3

были выбраны туман, неравномерность освещения и дождь (табл. 8—10). На рис. 7—9 продемонстрированы условия, при которых проводились соответствующие эксперименты. Эксперименты показали необходимость дополнительной обработки входных данных для нормальной работы системы наблюдения в сложных погодных условиях.

Заключение

В представленной статье рассмотрена система обнаружения и сопровождения пешеходов для решения задачи видеонаблюдения за пешеходными переходами. Проанализированы особенности решаемой задачи, спроектирована архитектура системы, позволяющая непрерывно обновлять список сопровождаемых объектов и восстанавливать потерянные объекты. На основе анализа возможности применения различных подходов к решаемой задаче сделан выбор алгоритма трекинга и подобрана конкретная реализация. Проведен ряд экспериментов, измеряющих качество, скорость и устойчивость работы системы.

Результаты экспериментов выявили ряд задач для последующего исследования, среди которых разработка механизма автоматического восстановления после перекрытий, присоединение к системе блока нейтрализации помех, вызванных погодными условиями.

Список литературы

1. **Li X., Hu W., Shen C.** et al. A survey of appearance models in visual object tracking // ACM Transactions on Intelligent Systems and Technology (TIST). 2013. Vol. 4, N 4. Article № 58.

2. **Lalonde M., Foucher S., Gagnon L.** et al. A system to automatically track humans and vehicles with a PTZ camera // Proceedings of SPIE of Defense & Security: Visual Information Processing XVI. Orlando, USA. April 2007. Vol. 6575. Article N 657502.

3. **Kim I. S., Choi H. S., Yi K. M.** et al. Intelligent visual surveillance — A survey // International Journal of Control, Automation and Systems. 2010. Vol. 8, N 5. P. 926—939.

4. **Elarbi-Boudihir M.** Intelligent video surveillance system architecture for abnormal activity detection // The International Conference on Informatics and Applications (ICIA2012). Malaysia. June 2012. P. 102—111.

5. **Jalal A. S., Singh V.** The State-of-the-Art in Visual Object Tracking // Informatica Slovenia. 2012. Vol. 3, N 1/2. P. 52—74.

6. **Колосовский М. А.** Система обнаружения объектов интереса в задаче видеонаблюдения за пешеходными переходами // Ползуновский вестник. 2014. № 2. (в печати).

7. **Maggio E., Cavallaro A.** Video Tracking: Theory and Practice. London. Wiley, 2011. 280 p.

8. **Shi J., Tomasi C.** Good Features to Track // 1994 IEEE Conference on Computer Vision and Pattern Recognition (CVPR'94). Seatle, USA. June 1994. P. 593—600.

9. **Lucas B., Kanade N.** An Iterative Image Registration Technique with an Application to Stereo Vision // 7th International Joint Conference on Artificial Intelligence (IJCAI'81). Vancouver, Canada. August 1981. Vol. 2. P. 674—679.

10. **Bircheld S.** Elliptical Head Tracking Using Intensity Gradients and Color Histograms // 1998 IEEE Conference on Computer Vision and Pattern Recognition. Santa Barbara, USA. June 1998. P. 232—237.

11. **Dokladal P., Enficiaud R., Dejnozkova E.** Contour-based object tracking with gradient-based contour attraction field // 2004 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '04). May 2004. Vol. 3. P. 17—21.

12. **Boudoukh G., Leichter I., Rivlin E.** Visual tracking of object silhouettes // 16th IEEE International Conference on Image Processing (ICIP). Cairo, Egypt. November 2009. P. 3625—3628.

13. **Guan L., Franco J., Pollefeys M.** Multi-Object Shape Estimation and Tracking from Silhouette Cues // 2008 IEEE Conference on Computer Vision and Pattern Recognition (CVPR'08). Anchorage, USA. June 2008. P. 1—8.

14. **Zhao P., Zhu H., Li H., Shibata T.** A Directional-Edge-Based Real-Time Object Tracking System Employing Multiple Candidate-Location Generation // IEEE Transactions on Circuits and Systems for Video Technology. 2013. Vol. 23, N 3. P. 503—517.

15. **Zhu G., Zeng Q., Wang C.** Efficient edge-based object tracking // Journal of Pattern Recognition. 2006. Vol. 39, N 11. P. 2223—2226.

16. **Mitiche A., Bouthemy P.** Computation and analysis of image motion: A synopsis of current problems and methods // International Journal of Computer Vision. 1996. N 19 (1). P. 29—55.

17. **Barron J. L., Fleet D. J., Beauchemin S. S.** Performance of optical flow techniques // International Journal of Computer Vision. 1994. N 12 (1). P. 43—77.

18. **Kloihofer W., Kampel M.** Interest Point Based Tracking // 20th International Conference on Pattern Recognition (ICPR'10). Istanbul, Turkey. August 2010. P. 3549—3552.

19. **Gauglitz S., Höllerer T., Turk M.** Evaluation of Interest Point Detectors and Feature Descriptors for Visual Tracking // International Journal of Computer Vision. 2011. Vol. 94, N 3. P. 335—360.

20. **Belagiannis V., Schubert F., Navab N., Ilic S.** Segmentation Based Particle Filtering for Real-Time 2D Object Tracking // 12th European Conference on Computer Vision (ECCV'12). Florence, Italy. October 2012. Vol. 4. P. 842—855.

21. **Papoutsakis K., Argyros A.** Object Tracking and Segmentation in a Closed Loop // 6th International Symposium on Visual Computing (ISVC 2010). Las Vegas, USA. November 2010. Vol. 1. P. 405—416.

22. **Ling T., Meng L., Kuan L., Kadim Z., Al-Deen A.** Colour-based Object Tracking in Surveillance Application // The International MultiConference of Engineers and Computer Scientists (IMECS 2009). Hong Kong, China. March 2009. Vol. 1. P. 459—464.

23. **Wang D., Lu H., Xiao Z., Chen Y.** Fast and effective color-based object tracking by boosted color distribution // Journal of Pattern Analysis and Applications. 2013. Vol. 16, N 4. P. 647—661.

24. **Колосовский М. А., Крючкова Е. Н.** Настройка параметров алгоритма сегментации изображений QuickShift // Программная инженерия. 2013. № 5. С. 11—20.

25. **Wiggin E.** Lucas-Kanade Tracker with pyramid and iteration. MATLAB Central File Exchange. 2012. URL: http://www.mathworks.com/matlabcentral/fileexchange/30822

26. **Birchfield S.** KLT: An Implementation of the Kanade-Lucas-Tomasi Feature Tracker. URL: http://www.ces.clemson.edu/~stb/klt/

27. **OpenCV** library. URL: http://opencv.org

28. **Leibe B., Schindler K., Van Gool L.** Coupled Detection and Trajectory Estimation for Multi-Object Tracking // 2007 IEEE 11th International Conference on Computer Vision. Rio de Janeiro, Brazil. 2007. P. 1—8.

ИНФОРМАЦИЯ

УДК 004.05

Н. С. Левшин, специалист по информационной безопасности, ООО "Азитос", г. Ростов-на-Дону,
e-mail:admin@biznes-informatika.ru

О проблемной ситуации, сложившейся на рынке систем, предназначенных для организации Интернет-магазинов

Анализируются вопросы, связанные с беспрепятственным выходом низкокачественных программных продуктов на рынок информационных систем, обеспечивающих процессы электронной коммерции. В качестве подтверждения актуальности рассматриваемой темы проводится обзор новой коммерческой системы управления контентом "Shop-Script 5", предназначенной для создания Интернет-магазинов.

Ключевые слова: *информационные системы, электронная коммерция, интернет-магазин, Shop-Script 5, Schema.org, Joomla 2.5, CMS, система управления контентом, поисковая оптимизация*

N. S. Levshin

On the Problematic Situation on the Market of Systems for Online Stores Organization

In this article raises the problem of the low-quality software products and their unhindered access to the market of information systems ensuring the processes of e-commerce. To prove the relevance of this issue the article features a review of a new paid content-management system — "Shop-Script 5", intended for building online stores.

Keywords: *information systems, e-commerce, online store, Shop-Script 5, Schema.org, Joomla 2.5, CMS, content management system, search engine optimization*

Экспоненциальный рост объемов рынка электронной коммерции неизбежно влечет за собой рост объемов поставок информационных систем, обеспечивающих его техническую поддержку. При этом причиной появления новых систем может являться не столько качественное развитие технологий, сколько стремление того или иного разработчика программного обеспечения занять свою нишу в перспективном сегменте. Желая как можно быстрее реализовать свои планы по внедрению на этот рынок, компании нередко предоставляют своим клиентам программные решения, содержащие различные недоработки и уязвимости.

Сталкиваясь с необходимостью выбора платформы для будущего интернет-магазина организации и предприниматели, в первую очередь, руководствуются желанием выйти на рынок электронной коммерции с минимальными капиталовложениями, а не соображениями надежности и безопасности. Отдавая предпочтение более выгодным с их точки зрения решениям, представители малого и среднего бизнеса далеко не всегда осознают, к каким последствиям для их проектов может привести использование ненадежных информационных систем. Сложившееся положение дел открывает широкие возможности для беспрепятственного вывода непроверенных программных продуктов на рынок систем для электронной коммерции.

В случае обнаружения клиентом недоработок большинство разработчиков готово безоговорочно вернуть ему деньги, потраченные на приобретение системы. Однако, как правило, недостатки обнаруживают уже после того, как клиент вложил определенное

количество времени и средств в разработку интернет-магазина на базе конкретной системы. В такой ситуации полный отказ от ее использования ему просто не выгоден, так как вложенные им в ходе этого средства уже не будут компенсированы. То же относится и к ущербу, который владелец интернет-магазина может понести уже в процессе ведения активной коммерческой деятельности на основе ненадежной системы.

Таким образом, на рынке программного обеспечения сложилась ситуация, которая деструктивно влияет на рынок электронной коммерции. Она задерживает появление новых игроков и вносит помехи в работу уже вышедших на рынок. Очевидно, что для проектов малого и среднего бизнеса такое положение вещей может приводить к печальным, а иногда и вовсе фатальным последствиям. В связи с этим обретает особую актуальность необходимость разработки системы мер и методов для своевременного предотвращения появления и распространения ненадежных информационных систем.

Возможным выходом из сложившейся негативной для развития рынка электронной коммерции ситуации может послужить публикация независимыми экспертами аналитических обзоров информационных систем в авторитетных общедоступных источниках. В качестве примера такого обзора можно привести анализ новой CMS (от англ. *content management system* — система управления контентом) для создания интернет-магазинов. Эта система выпущена на рынок информационных технологий (IT-рынок) в начале 2013 г. российской компанией "WebAsyst", ведущей разработки в области систем для электронной коммерции уже более 10 лет.

История появления системы "Shop-Script 5" на рынке CMS

Появление CMS "Shop-Script 5" было анонсировано компанией-разработчиком "WebAsyst" 19 декабря 2012 г. На тот момент компанией уже было выпущено на рынок несколько хорошо зарекомендовавших себя решений для разработки сайтов различных типов (в том числе и интернет-магазинов). Своим названием система "Shop-Script 5" обязана именно череде предыдущих разработок компании в области систем для обеспечения процессов электронной коммерции, включая "Shop-Script 1.0" (2002 г.), "Shop-Script 2.0" (2003 г.), "Shop-Script Premium" (2005 г.) и "WebAsyst Shop-Script" (2008 г.).

Одновременно с анонсом новой CMS стартовал этап предварительного заказа, участникам которого разработчики обещали предоставить систему на месяц раньше официального релиза, запланированного на 12 марта 2013 г. Компания обещала вернуть покупателям деньги, если к этой дате продукт еще не будет окончательно готов. Согласно заверениям разработчиков, система "Shop-Script 5" должна была стать наиболее удобной из существующих на рынке платных CMS за счет воплощения в ней всех самых актуальных решений для электронной коммерции.

Заявленные разработчиками функциональные возможности "из коробки" (базовая комплектация системы) и хорошая репутация компании привлекли немало клиентов. Приведем некоторые цитаты из рекламных материалов, размещенных на официальном сайте системы "Shop-Script 5": "В "Shop-Script 5" есть все, что нужно для открытия интернет-магазина: более 250 полезных функций, которые позволят вам продавать больше"; "Shop-Script 5" является одним из самых функциональных скриптов интернет-магазина для малого и среднего бизнеса в рунете"; "На основе разных версий "Shop-Script" сегодня работают более 10 000 прибыльных интернет-магазинов по всему миру"; "Открыть интернет-магазин очень просто, и для этого не нужно искать программистов и технических специалистов" [1].

Перечислен на официальном сайте и более конкретный перечень возможностей, реализованных в новой системе управления контентом "Shop-Script 5". В числе прочих заявлены следующие функции: фильтрация товаров по характеристикам; отзывы о товарах; разметка витрины магазина согласно стандартам микроформатов описания Schema.org; оптимальный для SEO функционал; экспорт товаров в Яндекс.Маркет и авторизация через соцсети [1].

Однако уже в первые дни старта продаж CMS "Shop-Script 5" в адрес разработчиков начали поступать жалобы от клиентов о несоответствии содержимого рекламных материалов фактическим функциональным возможностям и состоянию системы. На устранение большей части отмеченных покупателями несоответствий у сотрудников компании "WebAsyst" ушло несколько месяцев. Однако к этому времени покупателями уже были выявлены другие недостатки, повлекшие за собой продолжение отладочных мероприятий. Далее приведен подробный анализ соответствия фактических функциональных возможностей системы "Shop-Script 5" их описаниям в рекламных материалах, размещенных на сайте компании-разработчика.

Шаблоны оформления в CMS "Shop-Script 5"

Согласно условиям распространения системы, в комплекте с CMS "Shop-Script 5" покупатель получает несколько бесплатных вариантов оформления витрины интернет-магазина. К достоинствам каждого

из них можно отнести адаптивный дизайн (подстраивающийся под экраны различных устройств), соответствующий современным стандартам оформления интернет-страниц, и частичное внедрение микроразметки Schema.org (которое будет рассмотрено более подробно в ходе оценки механизмов реализации характеристик товаров). При этом для защиты базовых шаблонов от изменений, которые вносят в процессе разработки магазина, в системе реализован специальный механизм резервного копирования шаблонов.

Механизм резервного копирования шаблонов "Shop-Script 5" подразумевает, что одновременно с внесением первых изменений в шаблон оформления теряется возможность обновлять этот шаблон автоматически (причем никаких предупреждений об этом в системе не предусмотрено). Таким образом, если разработчики CMS "Shop-Script 5" найдут ошибки или уязвимости в предыдущих версиях шаблона и выпустят исправляющее их обновление, то владельцу интернет-магазина или нанятому им специалисту придется искать внесенные разработчиками исправления и вносить их в код своей копии шаблона оформления вручную, или продолжать использовать ненадежный шаблон.

В то же время предлагаемые в базовой комплектации шаблоны витрин организованы таким образом, что необходимость их доработки предельно высока. Например, они не представляют возможности провести без вмешательства в код системы следующее: разместить логотип магазина; разместить в верхней части сайта телефон и e-mail для справок и заказов (за исключением нескольких шаблонов); добавить метатег для файла favicon.ico; разместить код, позволяющий посетителям оперативно делиться ссылками на сайт магазина. Другими словами, в предоставляемых с системой шаблонах оформления отсутствует довольно большое число необходимых для нормального функционирования интернет-магазина настроек.

Еще одной недоработкой шаблонов оформления в "Shop-Script 5" является реализация плагина "Брэнды", предназначенного для вывода в меню сайта полного списка производителей. Когда число производителей товаров, представленных в интернет-магазине, колеблется от 1 до 10, примитивность реализации плагина не особенно заметна. Однако при большем числе производителей (что в условиях рыночной экономики не такая уж редкость) вывод их названий в форме одноколоночного выпадающего списка представляет собой не самое рациональное с позиции удобства для посетителей решение.

Еще один важный момент, непосредственно касающийся шаблонов оформления, заключается в некорректности ряда слов и словосочетаний, используемых в дизайне. Например, при использовании плагина "Брэнды" отсутствуют настройки, которые позволяли бы сменить выводимое на витрине магазина слово "Брэнды" на понятное каждому русскоязычному покупателю "Производители". Провести данное изменение можно только путем редактирования программного кода плагина и эту процедуру приходится повторять после каждого его обновления. Кроме этих недостатков следует отметить, что в первых версиях шаблонов оформления витрины около полугода можно было наблюдать несогласованные сочетания существительных с числительными, например: "Посмотрите все 1 отзывов о ...".

Все текстовые элементы дизайна интернет-магазина на основе системы "Shop-Script 5" задаются непосредственно в местах их размещения в шаблонах или в коде системы. В реализации такого подхода присутствуют, как минимум, два минуса. Первый заключается в том, что создателю интернет-магазина приходится угадывать, в каком из файлов шаблона разработчики системы разместили ту или иную фразу. При этом документация по редактированию шаблонов оформления практически отсутствует. Второй минус заключается в том, что при необходимости сменить или обновить шаблон оформления в новом шаблоне все исправления системных текстов потребуется прописывать заново. При этом следует заметить, что на этапе разработки пожелания владельца по оформлению магазина могут неоднократно меняться, и необходимость при каждом изменении шаблона выполнять поиск и замену языковых переменных приведет к существенному увеличению сроков выполнения проекта.

Описанный механизм реализации текстов витрин интернет-магазина не заслуживал бы порицания, если бы на рынке CMS уже долгие годы не существовал более удачный пример реализации текстов оформления. Речь идет о бесплатной CMS "Joomla" [2], в которой вместо текста оформления в шаблонах используют так называемые языковые переменные, значения которых вынесены в отдельные файлы системы. Такой подход позволяет разработчикам менять шаблоны оформления без потери текстовых данных; быстро находить тексты, требующие изменения (так как все они сосредоточены в одном файле); обеспечивает простоту организации мультиязычных сайтов (для каждого языка создается отдельный файл со значениями переменных) [3].

Панель администрирования в CMS "Shop-Script 5" мало уступает аналогичным панелям более дорогостоящих систем управления контентом. Как и в большинстве современных систем сегмента электронной коммерции, панель администрирования "Shop-Script 5" имеет адаптивный дизайн, что позволяет менеджеру магазина комфортно обрабатывать заказы с сенсор-

ных экранов мобильных устройств. Таким образом, подключив платные SMS-уведомления о новых заказах, владелец магазина может оперативно реагировать на заказы покупателей из любой точки мира, находящейся в зоне доступа сотовой сети.

К недостаткам административной панели CMS "Shop-Script 5" можно отнести отсутствие нумерации строк в редакторе кода. Другой существенный недостаток на данный момент уже устранен, однако нельзя не отметить тот факт, что более полугода первые покупатели системы наблюдали частичное отсутствие перевода используемых в системе описаний и названий переменных и плагинов. При этом не каждый разработчик, даже прошедший вузовский курс технического английского, мог понять, что плагин с названием "Store pickup" предназначен для реализации самовывоза товаров покупателями. Другими словами, русскоязычным клиентам приходилось работать с системой в буквальном смысле "со словарем", что существенно замедляло процесс создания интернет-магазинов на ее основе.

Характеристики товаров в CMS "Shop-Script 5"

Согласно рекламным материалам в CMS "Shop-Script 5" реализованы колоссальные функциональные возможности для отображения и сортировки товаров с учетом их характеристик. Эти возможности по ряду заложенных создателями системы "Shop-Script 5" параметров должны существенно превосходить реализованные в других CMS для электронной коммерции. На практике же разработчику, пожелавшему этими возможностями воспользоваться, приходится столкнуться с целым рядом вопросов, трудностей и недоработок.

В системе "Shop-Script 5" заложено большое число разнообразных типов характеристик, призванных организовать процесс управления информацией о товарах максимально комфортно. Например, для упрощения ввода информации о производителях довольно удобно использовать характеристики типа "Выпадающий список:Текст", а для добавления в описание товара информации о его комплектации — "Чекбоксы (множественный выбор):Текст".

Однако разработчики системы "Shop-Script 5" реализовали самые необходимые типы характеристик таким образом, что они конфликтуют с серверными модулями безопасности, которые традиционно используют для обеспечения защиты на большинстве серверов. Например, при включенных на сервере модулях безопасности suhosin и mod_security попытка добавления характеристик определенных типов вызывает бесконечный цикл перезагрузки страницы с настройками характеристик. Чтобы обойти эту трудность и полноценно использовать функциональные возможности, заложенные в "Shop-Script 5", разработчики системы рекомендуют отключать модули безопасности, несмотря на то, что это подвергнет угрозе безопасность сервера и, как следствие, самого интернет-магазина.

Каждой характеристике товара в системе "Shop-Script 5" присваивается уникальный идентификатор, формирующийся автоматически, путем транслитерации названия характеристики, вводимого разработчиком. Таким образом, характеристика с названием "производитель" по-умолчанию получит идентификатор "proizvoditel". При этом всегда существует возможность отредактировать сгенерированный системой идентификатор вручную. Однако мало кто задумывается о необходимости предоставления этой возможности. Дело в том, что ни на официальном сайте "Shop-Script 5", ни в самой системе не упоминается, что идентификаторы характеристик, помимо прочего, участвуют в формировании разметки Schema.org.

Как известно, в стандарте Schema.org отсутствует свойство "proizvoditel" [4]. Соответственно, элемент разметки, содержащий такой идентификатор, не будет правильно восприниматься поисковыми системами. Для корректной работы микроразметки необходимо, чтобы элемент разметки, отражающий название производителя товара, имел идентификатор "brand". Таким образом, необходимо, чтобы и характеристика, отвечающая за вывод списка производителей, имела соответствующий идентификатор. Однако ни панель настроек, ни официальный сайт "Shop-Script 5" не содержат никаких предупреждений об этом. Таким образом, если создатель Интернет-магазина на основе "Shop-Script 5" не проверит корректность микроразметки, присутствующая уязвимость рискует остаться незамеченной.

Отдельное внимание следует уделить возможностям, которые представляет система для сортировки товаров. Сортировка по их характеристикам настраивается в опциях конкретных категорий товаров. Порядок вывода параметров сортировки можно изменить только путем вмешательства в код системы. Сортировка товаров по характеристикам с главной страницы сайта не предусмотрена. Примечательно, что для некоторых типов характеристик возможность сортировки по ним вообще отсутствует (как и предупреждения об этом). Это обстоятельство может привести к тому, что разработчик магазина сначала заполнит описания и свойства для определенного числа товаров и лишь потом столкнется с необходимостью переделывать работу ради удобства покупателей. Разработчики "Shop-Script 5" поясняют, что такая ситуация стала возможна ввиду того, что они расширяют ассор-

тимент типов характеристик постепенно и не успевают внедрять механизмы сортировки для новых типов.

CMS "Shop-Script 5" и инструменты для поисковой оптимизации

Поисковые системы являются хорошим, а главное бесплатным источником посетителей (потенциальных клиентов интернет-магазина). Для того чтобы поисковые системы показывали сайт на более выгодных местах нежели сайты конкурентов, часто применяется комплекс мер, получивший название "поисковая оптимизация" (англ. *search engine optimization*, SEO) [5]. При этом основная задача SEO заключается в том, чтобы сделать страницы сайта более привлекательными для поисковых систем и тем самым способствовать увеличению числа потенциальных покупателей.

На настоящее время при распределении результатов поиска по запросам пользователей поисковые системы руководствуются несколькими десятками (а иногда, в зависимости от масштабов поисковой системы, даже сотнями) параметров, которые принято называть метриками. Часть из этих метрик зависит от действий, осуществляемых создателем и персоналом интернет-магазина в процессе его разработки и наполнения. Для упрощения этих благоприятных для поисковой оптимизации действий разработчики CMS, как правило, стараются заложить в свои системы специальные функции.

Так, из необходимых для SEO решений в CMS "Shop-Script 5" реализованы: удобная система автоматического формирования человекопонятных адресов страниц товаров; возможность заполнить для каждой страницы магазина метатеги keywords и description; внедрение микроразметки Schema.org (с рассмотренными выше оговорками); широкие базовые возможности для перелинковки страниц; возможность задать атрибут title для каждой картинки. Таким образом, базовые функции системы "Shop-Script 5" гарантированно перекрывают 5—6 из актуальных для SEO метрик. В то же время другие существующие на рынке программные продукты показывают, что в функциональные возможности CMS можно было бы заложить существенно больше готовых решений для поисковой оптимизации.

Начнем с того, что ни в одной из тем оформления "Shop-Script 5" не прописан метатег, указывающий поисковым роботам путь к файлу favicon.ico (например, <link href = "favicon.ico" rel = "shortcut icon" type = "image/x-icon" />). По своему характеру этот недостаток относится к недостаткам шаблонов оформления, однако последствия он имеет только для результатов поисковой выдачи. Отсутствие этого тега приводит к тому, что в сниппете (форме отображения сайта в результатах поискового запроса) отсутствует миниатюрный логотип сайта, в силу чего сниппет выглядит менее привлекательно для потенциальных посетителей.

Карта сайта в CMS "Shop-Script 5" формируется автоматически. Однако нигде не упоминается ее адрес (его можно лишь предположить, изучив немногочисленные инструкции на официальном сайте). Согласно действующим стандартам взаимодействия с поисковыми системами, адрес карты сайта требуется указать в файле robots.txt. Учитывая тот факт, что адрес расположения карты сайта постоянен и заранее известен разработчикам системы "Shop-Script 5", так как ими же и задается, они могли предусмотрительно заложить это в базовые функции системы.

В качестве существенного минуса механизма формирования карты сайта можно отметить тот факт, что в нее не включаются страницы, создаваемые обособлено от каталога товаров. Речь о страницах, на которых могут быть перечислены условия работы магазина, условия доставки и гарантийного обслуживания, схемы проезда к пунктам выдачи товаров, условия оферты, условия скидок. Необходимость наличия таких страниц для сайтов, ориентированных на электронную коммерцию, очевидна и невнимание разработчиков системы "Shop-Script 5" к этому вопросу удивительно.

Наиболее существенным недостатком CMS "Shop-Script 5" в аспекте вопросов поисковой оптимизации является отсутствие функциональных возможностей для автоматического экранирования внешних ссылок (скрытия ссылок на чужие сайты с использованием специальных тегов, параметров и атрибутов ссылок: noindex, nofollow, go-redirect). Пользуясь этим, мошенники могут легко оставлять ссылки на свои сайты через систему отзывов, маскируя их под отзывы обычных покупателей. Реализация таких действий со временем приведет к понижению позиций интернет-магазина в поисковой выдаче [6].

Примечательно, что в качестве подтверждения наличия функций облегчающих SEO, в рекламных материалах на официальном сайте "Shop-Script 5" приводится сертификат от компании "Корпорация РБС" ("BDBD Group"), которая является одним из лидеров по числу клиентов в российском сегменте SEO-услуг. В сертификате утверждается, что "CMS "Shop-Script 5" оптимальна для SEO" [1]. Однако представленные выше результаты анализа несколько разнятся с этим заявлением и вынуждают усомниться либо в компетентности инспектирующей компании, либо в объективности выдачи сертификата.

Дополнительные возможности CMS "Shop-Script 5"

Рекламные материалы на официальном сайте CMS "Shop-Script 5" обещают покупателям системы и ряд функций, которые стали актуальными в электронной коммерции сравнительно недавно или имеют сугубо российскую специфику, ввиду чего до сих пор не входят в набор базовых функций популярных CMS. Таким образом, наличие этих функций в базовом наборе системы "Shop-Script 5" должно выгодно выделять ее на фоне других систем управления контентом [1].

Наиболее значимой для осуществления электронной коммерции функцией представляется возможность экспорта товаров в "Яндекс.Маркет" (российскую агрегированную витрину интернет-магазинов, обладающую колоссальной аудиторией) [7]. Рекламные материалы официального сайта подразумевают, что плагин, предназначенный для реализации экспорта данных о товарах в систему "Яндекс.Маркет", успешно работает в "Shop-Script 5" с мая 2013 г. Однако по факту данный плагин более полугода находился в состоянии перманентной доработки, что делало его использование затруднительным.

Функциональные возможности для авторизации покупателей через социальные сети, реализованные в CMS "Shop-Script 5", выгодно отличаются от большинства других функций системы наличием подробных пояснений по настройке. Однако и с ними связан целый ряд различных проблемных ситуаций, к числу которых относятся перечисленные далее.

- По состоянию на начало 2014 г. регистрация пользователей через API mail.ru продолжает проходить некорректно, хотя создатели системы "Shop-Script 5" были уведомлены об этой проблеме еще в сентябре 2013 г. Вопрос заключается в том, что посетитель магазина успешно авторизуется через данный сервис, однако его учетная запись в базе данных магазина не заполняется контактной информацией из аккаунта в базе mail.ru, а получает название <без имени>, затрудняющее дальнейшую идентификацию этого посетителя.
- В качестве логотипа для авторизации через социальную сеть Google Plus долгое время использовался общий логотип компании Google, а не иконка конкретного сервиса. Это обстоятельство вводило посетителей магазина в заблуждение относительно возможностей авторизации, так как они могли быть не знакомы с общим логотипом и принимать его за логотип другой системы.
- Для кнопки каждой социальной сети загружается отдельное изображение-логотип, в то время как современными стандартами веб-разметки рекомендуется загружать одно изображение, содержащее логотипы всех сервисов, и посредством технологии CSS задавать изображения для отдельных сервисов, указывая координаты области на этом изображении. Более того, изображения для некоторых сервисов загружаются со сторонних сайтов, что приводит к дополнительным задержкам при загрузке страниц интернет-магазина.

Следует также отметить, что механизм интеграции с социальными сетями, реализованный в "Shop-Script 5", требует от разработчика, занимающегося построением интернет-магазина на ее основе, создания в каждой сети учетной записи и получения для каждого сайта ряда идентификаторов. В то же время в сегменте авторизации через социальные сети уже несколько лет существуют такие сервисы, как uLogin и Loginza, бесплатно предоставляющие API для авторизации пользователей. Благодаря им для организации авторизации на популярных CMS, например "Joomla", разработчику достаточно установить и настроить соответствующий плагин (средние затраты на это составляют 2—3 мин). При этом посетителям предоставляется возможность авторизации через существенно больший (в сравнении с представленным в "Shop-Script 5") перечень социальных сетей (22 сервиса при использовании API от uLogin) и разработчику не требуется для каждой из них проводить какие-либо дополнительные манипуляции по настройке.

Для учета предложений клиентов по расширению функциональных возможностей и устранению недочетов CMS "Shop-Script 5" был открыт специальный сервис. Этот сервис располагается по адресу http://webasyst.reformal.ru и позволяет оставить предложения и замечания по развитию системы. Однако отвечая на некоторые предложения и пожелания, разработчики нередко категорично заявляют об отсутствии намерений по внедрению той или иной функции, не удосуживаясь при этом обосновать свою позицию. В большинстве случаев это означает, что необходимые дополнительные функциональные возможности клиенты компании смогут получить только за отдельную плату — получив готовое решение (если оно уже существует на рынке), либо путем привлечения разработчика для решения конкретной задачи.

В качестве примера такой ситуации можно рассмотреть отсутствие в базовом наборе функций системы "Shop-Script 5" возможности для удобного резервного копирования интернет-магазина, в то время как создание резервных копий очень важно для минимизации рисков любого интернет-проекта. Для решения этой задачи один из сторонних разработчиков предлагает плагин Q-Backup, который он реализует по цене в 1000 руб. В то же время для многих других CMS можно найти бесплатные решения, реализующие аналогичные возможности (например, расширение Akeeba BackUp для "Joomla").

Выводы

Из представленного выше анализа соответствия фактического состояния рассматриваемой системы описанному в рекламных материалах следует, что несмотря на заверения разработчиков, создание конкурентоспособного интернет-магазина на базе системы "Shop-Script 5" без навыков веб-разработки невозможно. Фактический набор функциональных возможностей системы также далеко не во всем соответствует заявленному в рекламных материалах. При этом те или иные проблемные ситуации наблюдаются в реализации практически каждой функции системы, что не позволяет выделить существенных ее преимуществ в сравнении с другими CMS.

Следует также понимать, что многие из обнаруженных за время распространения системы недоработок не получили отражения в данном обзоре по причине их непродолжительного, в сравнении с рассмотренными выше, существования. Тем не менее нельзя не упомянуть о том, что наличие этих ошибок доставляло и доставляет покупателям системы немало хлопот. Подтверждением этого служит страница по адресу www.shop-script.ru/changelog, на которой можно

Сравнение CMS "Shop-Script 5" и CMS "Joomla 2.5" (в сочетании с расширением "Virtuemart 2")

Параметр сравнения	CMS "Shop-Script 5"	CMS "Joomla 2.5" + "Virtuemart 2"
Стоимость	9999 руб.	Бесплатно
Возможность организации полноценного Интернет-магазина лицом, не обладающим навыками HTML, CSS и программирования	Нет	
Наличие документации	Слабо документирована	Множество подробных инструкций, однако русскоязычные инструкции рассредоточены по разным сайтам
Проблемы с безопасностью	Возможны, так как использование системы создает необходимость отключить серверные модули безопасности для корректной работы системы	Отсутствуют при достаточной опытности разработчика [8]
Число бесплатных шаблонов оформления витрины с адаптивным дизайном	7 в базовой комплектации	0 в базовой комплектации, несколько десятков на сторонних сайтах
Ошибки и огрехи в шаблонах оформления витрины магазина	Встречаются, устранение требует специальных навыков	
Адаптивный шаблон панели управления	Да	Нет
Перевод настроек на русский язык	Не всегда	
Настройки для реализации мультиязычности	Не предусмотрены	Присутствуют в базовых настройках системы
Сортировка по характеристикам	Возможны проблемы с настройкой и поиском решений	
Экранирование внешних ссылок	Не предусмотрено	Достаточно установить и настроить бесплатное расширение
Дополнительные функциональные возможности для SEO	Наиболее востребованные доступны в базовом комплекте	
Авторизация пользователей через социальные сети	Авторизация через 5 сервисов (требует около получаса настройки)	Авторизация через 22 сервиса (2...3 минуты установки и настройки расширения)
Экспорт товаров в "Яндекс.Маркет"	Установка и настройка бесплатного плагина	
Возможность добавить форум для общения покупателей	Нет	Возможно установить и настроить бесплатный компонент (например, "Kunena")
Возможность для организации службы поддержки покупателей	Платный компонент "Поддержка", 5999 руб.	Различные расширения, например, бесплатное "Live Support Chat"
Создание резервных копий сайта из панели управления CMS	Платный компонент "Q-Backup", 1000 руб.	Бесплатный компонент "Akeeba BackUp"

наблюдать пополняющийся перечень исправлений, которые вносятся в систему разработчиками, и официальный форум компании "WebAsyst", расположенный по адресу http://forum.webasyst.ru, также регулярно пополняющийся жалобами клиентов компании на некорректную работу системы "Shop-Script 5".

Принято считать, что специализированные системы управления контентом, распространяемые на коммерческой основе, существенно превосходят системы с открытым исходным кодом. Однако, как показывает проведенное исследование, даже среди неспециализированных систем управления контентом можно найти решения для электронной коммерции более удобные, чем CMS "Shop-Script 5". В качестве подтверждения данного вывода приведено сравнение системы "Shop-Script 5" с решением для организации интернет-магазина на базе бесплатной CMS "Joomla 2.5" и ее расширения "Virtuemart 2". Ключевые критерии их сравнения приведены таблице.

Анализируя таблицу можно прийти к выводу, что при прочих равных условиях решение о создании полноценного интернет-магазина средствами бесплатного программного тандема систем "Joomla 2.5" и "Virtuemart 2" окажется гораздо более логичным для представителей малого и среднего бизнеса, нежели решение об использовании рассмотренного в данной работе коммерческого продукта от компании "WebAsyst".

Резюмируя получившийся обзор, можно констатировать, что компания "WebAsyst" уже около года предоставляет клиентам продукт, не в полной мере соответствующий описанию, заявленному в рекламных материалах. В результате чего покупатели системы вынуждены выступать в роли ее бета-тестеров, теряя время и средства на устранение недостатков системы и их последствий.

Следует особо отметить и тот факт, что речь идет о продукте компании, занимающейся разработкой решений для электронной коммерции уже более десяти лет, и сумевшей, благодаря этому, привлечь большое число клиентов. На официальном сайте "Shop-Script 5" утверждается, что на базе этой новой системы создано уже "более 1500 *работающих* интернет-магазинов" [1]. Такой прецедент лишний раз подчеркивает актуальность и масштабность затронутого в данной работе проблемного вопроса беспрепятственного выхода низкокачественных информационных систем на рынок программного обеспечения для электронной коммерции, и необходимость привлечения внимания общественности к сложившейся на этом рынке ситуации.

Список литературы

1. **Официальный** сайт CMS "Shop-Script 5". URL: http://www.shop-script.ru (дата обращения: 29.01.2014).

2. **Есиков А. В.** Сравнительный анализ систем управления контентом (Content Management System — CMS) // Перспективы развития информационных технологий. 2013. № 13. С. 62—65.

3. **Анашкин Р. В., Беневоленский С. Б., Кирьянов А. А.** Особенности интеграции системы автоматизированного перевода в CM Services для управления локализацией // Современные проблемы науки и образования. 2013. № 2. С. 181.

4. **Organization** — Schema.org. URL: http://schema.org/Organization (дата обращения: 29.01.2014).

5. **Степаненко Г. А.** Моделирование поисковой оптимизации в электронной коммерции // Вестник ИНЖЭКОНа: Серия Экономика. 2013. № 6. С. 166—168.

6. **Лысенко Д. С., Гридина Е. Г.** Факторы формул ранжирования поисковых систем Яндекс и Google // Качество. Инновации. Образование. 2011. № 3. С. 55—60.

7. **Бебрис А. О.** Маркетинговая составляющая повышения конкурентоспособности Интернет-магазинов // Проблемы современной экономики (Новосибирск). 2013. № 14. С. 105—107.

8. **Гольчевский Ю. В., Северин П. А.** О безопасности Интернет-сайтов под управлением системы управления контентом Joomla // Вопросы защиты информации. 2012. № 3. С. 44—49.

ООО "Издательство "Новые технологии". 107076, Москва, Стромынский пер., 4

Дизайнер *Т. Н. Погорелова*. Технический редактор *Е. М. Патрушева*. Корректор *М. Г. Джавадян*

Сдано в набор 05.03.2014 г. Подписано в печать 21.04.2014 г. Формат 60×88 1/8. Заказ PI514
Цена свободная.

Оригинал-макет ООО "Адвансед солюшнз". Отпечатано в ООО "Адвансед солюшнз".
119071, г. Москва, Ленинский пр-т, д. 19, стр. 1.