Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«Московский государственный технический университет
Имени Н.Э. Баумана»

СОВРЕМЕННЫЕ КОМПЬЮТЕРНЫЕ СИСТЕМЫ И ТЕХНОЛОГИИ

Сборник трудов кафедры
«Компьютерные системы и сети»
МГТУ им. Н.Э. БАУМАНА

НИИ Радиоэлектроники и лазерной техники

Москва 2014

УДК 004.2/9(048.2)
ББК 32.973
С56

Современные компьютерные системы и технологии: сборник трудов
С56 кафедры «Компьютерные системы и сети» МГТУ им. Н.Э. Баумана. –М.: Изд-во НИИ Радиоэлектроники и лазерной техники, 2014, - 180 с.

ISBN 978-5-4384-0023-3

Сборник трудов преподавателей, аспирантов, студентов и молодых ученых кафедры «Компьютерные системы и сети» традиционно охватывает материалы двух научно-технических проводимых кафедрой ежегодно в ноябре и апреле месяцах. В настоящем сборнике представлены статьи, подготовленные по результатам докладов на конференциях, состоявшихся в 2013/2014 учебном году.

Тематика вошедших в сборник статей охватывает широкий спектр вопросов, относящихся к современным направлениям развития информатики, компьютерной техники, сетей, технологий обработки информации, разработки аппаратных средств и программных систем. Публикуемые материалы отражают научные интересы профессорско-преподавательского коллектива кафедры, обучающихся на ней студентов и аспирантов и могут быть полезны читателям, специализирующимся в области современной компьютерной техники и информационных технологий.

Статьи печатаются в авторской редакции.

УДК 004.2/9(048.2)
ББК 32.973

ISBN 978-5-4384-0023-3

УДК 004.942

ОСОБЕННОСТИ АВТОМАТИЗАЦИИ ПРИНЯТИЯ РЕШЕНИЙ В ПОДСИСТЕМЕ НАЧАЛЬНЫХ УСЛОВИЙ ПУСКА РАКЕТ С РАЗЛИЧНЫХ ВИДОВ БАЗИРОВАНИЯ

Ю.Г. Буланова, Т.Н. Ничушкина

Ключевые слова: *автоматизированное принятие решений, система управления стрельбой, начальные условия пуска, крылатые ракеты*

Key words:. *automated decisions, system of fire control, the initial starting conditions, cruise missiles*

Аннотация: *Данная работа посвящена анализу особенностей, проявляющихся при принятии решений о выборе начальных значений параметров пуска противокорабельных ракет в универсальной корабельной системе управления стрельбой в условиях сложной реальной обстановки. В статье определены задачи, которые необходимо решить, чтобы автоматизировать процесс принятия решения о значениях конкретных начальных параметров при различных условиях пуска на основе анализа этих условий. Выполнено исследование возможностей и способов решения этих задач, а также их формальная постановка.*

Современные оригинальные научно-технические решения позволили противокорабельным комплексам с крылатыми ракетами, созданными в ОАО «ВПК «НПО машиностроения», оставаться наиболее эффективным оружием борьбы как с надводными кораблями, так и наземными целями, а также быть конкурентоспособными на мировом рынке вооружений, что очень важно в настоящее время.

Противокорабельные ракеты (ПКР) ОАО «ВПК «НПО машиностроения» большой, средней и малой дальности, обладающие надводным и подводным стартом, минимальной длительностью режима предстартовой подготовки (ПП) примерно 115-120 секунд, обычно размещаются на надводных кораблях и подводных лодках. Указанными противокорабельными ракетами вооружаются корабли различных классов - от ракетных катеров до надводных и подводных ракетных крейсеров, включая наземные ракетные комплексы.

Анализ зарубежной технической информации систем управления огнем (СУО) различных видов базирования военно-морских сил стран НАТО, а также технической документации о подобных системах ОАО «ВПК «НПО машиностроения», показал: в условиях быстро изменяющейся целевой обстановки в театре военных действий (ТВД),

резкого увеличения потоков информации и сокращения времени на ее обработку принятие оптимального решения на применение ПКР возможно только при условии автоматизации процесса принятия решений.

В пользу автоматизации процесса принятия решений на применение различных ПКР в универсальной корабельной системе управления стрельбой (УКСУС) говорит и тот факт, что при большой дальности полета различных ПКР ОАО «ВПК «НПО машиностроения» такая автоматизация даст возможность кораблю-носителю занять наиболее удобную позицию для атаки, первому нанести удар и оторваться от противника после решения боевой задачи с учетом всех заложенных в полетное задание тактических маневров ракеты по траектории. Большая скорость полета ПКР позволяет обходить закладываемые в полетные задания зоны противодействия цели, что дополнительно затрудняет перехват ПКР в полете. Кроме того, в полетное задание ракеты можно рассчитывать и закладывать различные маневры по курсу, в том числе, при необходимости, противозенитный маневр типа «Спираль» перед целью. А так как в УКСУС ведется залповый метод стрельбы с переменным интервалом пуска различных ПКР, это позволяет осуществлять их рациональное взаимное построение на траектории полета.

Исходя из вышеизложенного, целью предлагаемой работы является формулировка основных принципов автоматизации принятия решений на примере алгоритмизации различных задач, включая контроль начальных условий пуска (НУП) ПКР, в подсистеме автоматизации принимаемых решений в УКСУС надводного корабля (НК).

Анализ процесса принятия решений показал, что подсистема автоматизации принимаемых решений (АПР) УКСУС должна обеспечивать:

- оценку вероятности попадания цели в зону досягаемости целей;
- оценку вероятности попадания цели в зону обзора головки самонаведения (ГСН);
- расчет рекомендуемого курса для маневра НК с выходом на допустимую дистанцию стрельбы в кратчайшее время;
- расчет координат точек прицеливания при маневре выхода НК на допустимый рубеж атаки в кратчайшее время;
- выход по углу послестартового разворота ПКР на цель;
- расчет и контроль на скользящем интервале времени режима ПП количества ограничительных нарушений по текущим значениям контролируемых величин;
- дискретизацию измеряемых на НК текущих параметров движения носителя, которые используются при контроле НУП;

- определение и контроль текущих динамических параметров НК в течение ПП;
- формирование необходимых рекомендаций на ПУРО для окончательного принятия решения о сигналах «В зоне пуска» или «Не в зоне пуска».

Так как подсистема АПР УКСУС, по замыслу разработчиков, будет входить в состав функциональных задач УКСУС НК, то ее основное назначение – формирование общего признака «В зоне пуска» или «Не в зоне пуска» и разрешения пуска ракет от одного до полного боекомплекта при выполнении критериев, обеспечивающих безопасность корабля-носителя и ракет по динамике движения в пусковой установке и при выходе из нее.

Выбор основных принципов реализации подсистемы АПР УКСУС вместе с алгоритмом контроля НУП ПКР должен базироваться на анализе:

- физических явлений, связанных с интенсивным воздействием на корабль-носитель морского нерегулярного волнения моря в процессе ПП и старта ПКР;
- возможности и целесообразности изменения параметров движения корабля-носителя, например, таких как скорость хода и курса корабля-носителя в подсистеме АПР УКСУС в ходе текущего решения в НУП в тех случаях, когда это изменение способствует приведению параметров алгоритмов или функционалов НУП в область допустимых значений.

Для размещаемых на НК противокорабельных ракет наиболее важным является контроль обеспечения допустимости поперечных нагрузок безударности при выходе каждой ПКР из универсальной вертикальной пусковой установки (УВПУ).

При качке корабля-носителя на морском нерегулярном волнении моря на движущуюся ракету в транспортно-пусковом стакане (ТПС) при старте воздействуют:

- поперечные перегрузки, вызванные линейными и угловыми ускорениями качки в сочетании с высокой скоростью ПКР относительно ТПС.
- ударные ходовые перегрузки (удары корпуса корабля-носителя от встречи с волной).

Такие параметры движения корабля-носителя, как скорость хода и угол курса (точнее угол между направлением движения корабля-носителя и генеральным направлением распространения волн) оказывают существенное влияние на величины ускорений, вызванных качкой, и на ударные ходовые перегрузки. Изменение этого угла влияет на величину и направления действия ударных ходовых перегрузок, а также на величины амплитуд и частот параметров качки.

С увеличением скорости хода корабля-носителя увеличиваются амплитуды и частоты параметров качки, а также интенсивность ударных ходовых перегрузок и частота

их появления. Поэтому в подсистеме АПР УКСУС необходимо повысить надежность и безопасность старта ПКР при автоматизированном принятии решения. Для этого в подсистеме АПР УКСУС и в процессе решения указанных задач в режимах «Боевой», «Дежурный» и «АВР» (аварийный) необходимо осуществлять целенаправленное управление изменением различных параметров, в том числе изменениями скорости хода корабля-носителя и угла курса.

На основе данных о метрологической обстановке в районе пуска и в районе цели, поступающих с обеспечивающих систем корабля-носителя, будет рассчитываться начальное значение скорости хода носителя к моменту начала режимов «Боевой», «Дежурный» и «АВР» или к моменту начала проведения ПП и пуска ПКР в подсистеме АПР УКСУС.

Поэтому в будущей подсистеме АПР УКСУС необходимо заложить алгоритмы для получения надежной оценки от обеспечивающих систем корабля-носителя, о степени волнения моря в период проведения ПП и пуска ПКР. При этом скорость хода носителя будет регулироваться подсистемой АПР УКСУС через рекомендации на отображении пульта управления ракетным оружием (ПУРО) УКСУС НК как до начала ПП, так и в процессе ПП с учетом необходимости обеспечения возможностей маневрирования корабля-носителя по «Змейке».

В дальнейшем, с целью приведения величины функционалов НУП в рамки допустимых диапазонов в подсистему АПР УКСУС исходя из результатов, полученных при решении задачи контроля НУП в режимах «Боевой», «Дежурный» и «АВР», скорость может быть рекомендована к уменьшению с учетом изменений по курсу.

С учетом возможного выполнения маневра «Змейка», необходимость которого определяется по результатам решения задачи о досягаемой дальности цели, изменение угла курса в режимах «Боевой», «Дежурный» и «АВР» будет зависеть от допустимости изменения боевого курса корабля-носителя.

Для проверки правильности функционирования разрабатываемой подсистемы АПР УКСУС, в том числе выбора ограничительных констант, алгоритмов контроля НУП и других сопутствующих алгоритмов, необходимо будет провести исследования на различных моделирующих стендах: в частности, на исследовательском стенде алгоритмов УКСУС НК и исследовательском моделирующем стенде полета ПКР на начальном участке. Целью такого моделирования будет подтверждение правильности функционирования подсистемы АПР УКСУС, алгоритмов контроля НУП, определение правильности выбора ограничительных констант НУП, при которых будет обеспечиваться

выполнение требований по формированию общего признака «В зоне пуска» или «Не в зоне пуска» и разрешения пуска ракет с заданным уровнем вероятности.

Для проведения вышеуказанного моделирования необходимо создать математическую модель динамики движения НК в условиях морского нерегулярного волнения моря от 0 до 7 баллов для всего диапазона скоростей хода НК и углов курса между направлением движения НК и генеральным направлением движения волн.

Особое внимание в исследовательских работах, связанных с разработкой подсистемы АПР УКСУС для пускового контура различных ПКР типа «Оникс», «Яхонт» и других, будет уделено:

- исследованию всего пускового контура функционирования подсистемы АПР УКСУС в различных условиях применения;
- отработке взаимодействия с базовым ПО УКСУС НК различных ПКР типа «Оникс» и «Яхонт»;
- внедрению разработанных способов управления в базовое ПО УКСУС НК различных ПКР типа «Оникс» и «Яхонт», включая в ПО УКСУС различных видов базирования.

Вывод

В статье сформированы основные научно-технические подходы и особенности автоматизации принятия решений в подсистеме начальных условий пуска различных ракет на примере универсальной корабельной системы управления стрельбой для надводного корабля, которую можно будет адаптировать для применения ПКР с аналогичных систем управления для различных видов базирования. Указанная статья отличается новизной подхода в указанной проблематике.

Литература

1. О возможности оснащения иностранных кораблей носителей ПКР комплексом РО «Яхонт». Технический отчет. НПО машиностроения, 1997.

2. Аванпроект. Противокорабельный ракетный комплекс «Бастион». Комплекс наземного базирования. Глава 8.6. Комплексы средств автоматизации информационно-расчетного тракта (ИРТ) СБУ на самоходных пусковых установках и машинах боевого управления. НПО машиностроения, 1997. С. 89-154. (инв. 046360).

3. Аванпроект. Противокорабельный ракетный комплекс «Бастион». Особенности комплекса морского базирования. Глава 4. Корабельная аппаратура системы управления. НПО машиностроения, 1997. С. 29-70. (инв.046381).

4. Аванпроект. Противокорабельный ракетный комплекс «Бастион». Особенности комплекса авиационного базирования. Глава 5. Самолётная аппаратура предстартовой подготовки и пуск ПКР. НПО машиностроения, 1997. С. 18-57, (инв.046380).

УДК 004.054/[[004.023+004.8]/[005:[62::811]]]

СРАВНЕНИЕ ЭФФЕКТИВНОСТИ НЕКОТОРЫХ СТАТИСТИЧЕСКИХ МЕТОДОВ КЛАССИФИКАЦИИ НА ПРИМЕРЕ ТЕХНИЧЕСКИХ СТАТЕЙ

А.Г. Васнецов, Р.С. Самарев

Ключевые слова: наивный байесовский классификатор, пространство признаков, Евклидова метрика, расстояние Махаланобиса, косинусная мера.

Key words: naïve Bayes classifier, Feature vectors, Euclidean distance, Mahalanobis distance, Cosine similarity.

Аннотация: В статье рассмотрена возможность применения следующих статистических алгоритмов для классификации технических текстов: наивный байесовский классификатор, классификатор, основанный на нахождении ближайшего класса в пространстве признаков, метод k ближайших соседей. Проведены эксперименты с различными метриками: Евклидовой метрикой, метрикой Махаланобиса, Косинусной мерой. Сформулированы рекомендации по применению классификаторов для текстов технической тематики.

Введение

Классификация документов – одна из задач информационного поиска, заключающаяся в отнесении документа к одной из нескольких категорий на основании содержания документа. Классификация документов находит свое применение в таких областях как: ограничение области поиска в поисковых системах, автоматического составления аннотации, составление интернет-каталогов, фильтрация спама и т.д.

В данной работе исследовалась возможность применения некоторых наиболее популярных статистических алгоритмов классификации для классификации статей технического характера. Экспериментально оценена точность классификации при различных признаковых описаниях документов.

Задача классификации

Задача классификации определяется следующим образом. Имеется некоторое множество объектов, разделенное произвольным образом на непересекающиеся группы (классы). Для некоторого конечного подмножества объектов данного множества известно к каким классам они принадлежат. Это подмножество называется выборкой. Классовая принадлежность остальных объектов неизвестна. Задача заключается в построении алгоритма, способного классифицировать произвольный объект из исходного множества.

Подготовка выборки

В данной работе в качестве классифицируемого множества объектов рассматриваются статьи, взятые с Интернет-ресурса http://habrahabr.ru/. Особенностью этого ресурса является то, что каждая статья отнесена самим автором как минимум к одной из множества заранее определенных категорий. В данном случае под категорией понимается множество статей, объединенных общей тематикой. Такая особенность делает возможным автоматизировать процесс подготовки материала для тестирования классификаторов.

В качестве обучающей выборки было взято по 100 статей каждой из следующих пяти категорий:

1) программирование;
2) алгоритмы;
3) гаджеты;
4) информационная безопасность;
5) DIY или сделай сам.

В связи с тем, что указанный Интернет-ресурс позволяет отнести каждую статью к нескольким категориям, а задача классификации подразумевает назначение единственного класса для статьи, то возникает неоднозначность при ее классификации. В этом случае можно говорить о погрешности выборки. Оценим вероятность появления погрешности при классификации. Для этого посчитаем количество статей в выборке, которые относятся более чем к одной категории:

$$c = \sum_{a \in A : \left| h_a \cap H \right| > 1} 1,$$

где A – выборка;

h_a – множество категорий, ассоциированных со статьей $a, a \in A$;

H – множество всех рассматриваемых категорий (в данном случае 5 перечисленных выше).

Так как в выборке присутствуют статьи, относящиеся одновременно к нескольким рассматриваемым категориям, то количество уникальных статей в выборке будет меньше чем суммарное количество статей в каждой рассматриваемой категории. Оценим долю погрешности в выборке как отношение количества статей, относящихся более чем к одному классу c к общему числу уникальных статей $|A_u| = 451$:

$$E = \frac{c}{|A_u|} = 0.07317.$$

Эксперимент заключается в применении алгоритмов классификации к некоторым элементам исходного множества, для которых известна их классовая принадлежность, но которые не присутствуют в обучающей выборке. После применения алгоритмов сравнивается их результат с заранее определенными классами элементов. Считается доля правильно классифицированных элементов. Для проведения эксперимента было взято по 20 статей из каждой категории.

Описание объектов

Для описания объектов-статей использовалось признаковое описание. Признаковое описание – это вектор, составленный из значений фиксированного набора признаков данного объекта. Признаки в общем случае могут иметь различные типы.

Далее не будет делаться различия между объектом и его признаковым описанием. В данной работе были произведены эксперименты с несколькими вариантами признакового описания V :

1) наличие слова в тексте статьи $V^A = (V_i^A), V_i^A \in \{0,1\}, i = \overline{1..N}$, где N – количества уникальных слов;

2) количество слов в тексте статьи $V^B = (V_i^B), V_i^B \in , i = \overline{1..N}$, где N – количества уникальных слов;

3) наличие в тексте статьи множества часто совместно встречающихся слов, полученных алгоритмом FPGrowth [2] $V^C = (V_i^C), V_i^C \in \{0,1\}, i = \overline{1..N}$, где N – количество множеств часто совместно встречающихся слов;

4) признаки TF-IDF [4] $V^D = (V_i^D), V_i^D \in , i = \overline{1..N}$, где N – количества уникальных слов;

5) признаки IDF $V^F = (V_i^F), V_i^F \in , i = \overline{1..N}$, где N – количества уникальных слов.

Приведение слов к исходной форме

При подготовке текстов статей к классификации слова приводились к начальной форме с помощью программы mystem от Yandex [3]. Это делалось для исключения ситуаций, когда одно и то же слово в разных формах считалось бы классификатором разными словами, например:

«Классифицирует» → «Классифицировать»

Если слово невозможно привести к начальной форме, то слово оставлялось в исходной форме, например:

«return» → «return»

Наивный байесовский классификатор

Наивный байесовский классификатор – простой вероятностный классификатор, основанный на применении Теоремы Байеса со строгими (наивными) предположениями о независимости. Классификация с помощью байесовской модели заключается в максимизации вероятности $p\langle C|V_{1,}..,V_n\rangle$ где C – зависимая переменная класса, V – вектор признаков классифицируемого объекта.

Исходя из предположения о независимости V_i и V_j для каждого из рассматриваемых пространств признаков, при $i \neq j$ вытекает следующее выражение для классификатора:

$$classify(V_{1,}\dots,V_n)=\underset{C}{argmax}\left(p(C=c)\prod_{i=1}^{n}p(V_i=v_i|C=c)\right),$$

где $p\langle V_i = v_i \,|\, C = c\rangle$ – вероятность того, что признак объекта V_i примет значение v_i при условии принадлежности объекта классу c .

Так как применение байесовского классификатора для признаков не бинарного типа неэффективно, то его применение для пространства признаков на основе количества стол в тексте рассматриваться не будет. В случае бинарных признаков значение вероятностей $p\langle V_i = v_i \,|\, C = c\rangle$ определяется следующим выражением:

$$p\langle V_i = v_i \,|\, C = c\rangle = \left\{ \begin{array}{l} \dfrac{\sum v_{aj}}{|A_c|}, v_i = 1 \\ 1 - \dfrac{\sum v_{aj}}{|A_c|}, v_i = 0 \end{array} \right\},$$

где A_c – множество объектов, относящихся к классу $C, a_j \in A_c$;

v_{aj} – значение признака V_i у объекта a_j. Более подробное описание классификатора можно найти в [1].

Классификация на основе определения ближайшего класса в пространстве признаков

Следующий вид классификации, рассмотренный в данной работе - классификация на основе определения ближайшего класса в некоторой метрике. При этом каждому классу ставится в соответствие точка в пространстве признаков, координаты которой равны математическому ожиданию значения данного признака у объекта, принадлежащего рассматриваемому классу $F: C \to \overline{X}, X_i = M[V_i], V \in C, i = \overline{1 \ldots |V|}$. Классифицируемому объекту также ставится в соответствие точка в этом пространстве, координаты которой равны значению соответствующих параметров этого объекта. Затем рассчитывается расстояние от точки объекта до всех точек классов в какой-либо метрике. Объект считается принадлежащим к тому классу, расстояние до точки которого оказалось минимальным.

Евклидова метрика. При использовании Евклидовой метрики расстояние между точками *x, y* определяется с помощью формулы:

$$d(\overline{x}, \overline{y}) = \sqrt{\sum_{i=1}^{|V|} (y_i - x_i)^2} .$$

Расстояние Махаланобиса **–** мера расстояния между векторами случайных величин, обобщающая понятие евклидова расстояния. С помощью расстояния Махаланобиса можно определять сходство неизвестной и известной выборки. Оно отличается от расстояния Евклида тем, что учитывает корреляции между переменными и инвариантно к масштабу. Расстояние между точками определяется с помощью формулы:

$$d(\overline{x}, \overline{y}) = \sqrt{(\overline{x} - \overline{y}) S^{-1} (\overline{x} - \overline{y})},$$

где S – ковариационная матрица.

Косинусная мера. Расстояние между точками определяется как 1 минус косинус угла между векторами, идущими из начала координат в данные точки.

$$d(\overline{x}, \overline{y}) = 1 - \frac{\overline{x}\,\overline{y}}{|\overline{x}||\overline{y}|} .$$

Метод k ближайших соседей

Данный метод классификации схож с предыдущим, его описание можно найти в [5]. Эксперименты проводились с различными параметрами k в диапазоне от 1 до 30 с шагом 5. Лучший результат был получен при $k = 15$, он будет показан ниже (рисунок 5).

В качестве метрики использовалась косинусная мера, как показавшая наилучший результат для предыдущего метода.

Результаты экспериментов

В ходе эксперимента оценивалась эффективность классификации, где под эффективностью понимается отношение правильно классифицированных тестовых статей, к общему числу тестовых статей. Экспериментальные результаты оценки эффективности использования байесовского классификатора приведены на рисунке 1.

На данной и последующих диаграммах введены следующие обозначения:

«**Эталон»** – максимально возможная эффективность на данной выборке, определяется как 1-*E*, где *E* – процент погрешности выборки полученный выше.

«**1»** – Классификация с использованием пространства признаков основанного на наличие комплексов слов, сгенерированных FPGrowth.

«**2»** – Пространство признаков на основе наличия слова.

«**3»** – Пространство признаков на основе количества слова.

«**4**» – Пространство признаков на основе TFIDF.

«**5**» – Пространство признаков на основе IDF.

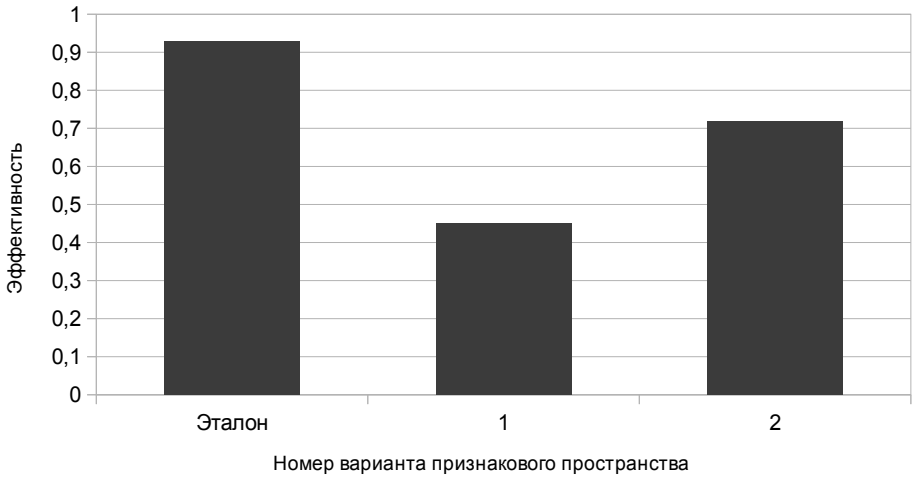


Рисунок 1. Диаграмма результатов тестирования байесовского классификатора

Диаграммы результатов тестирования классификатора на основе определения ближайшего класса в пространстве признаков приведены на рисунках 2, 3, 4.

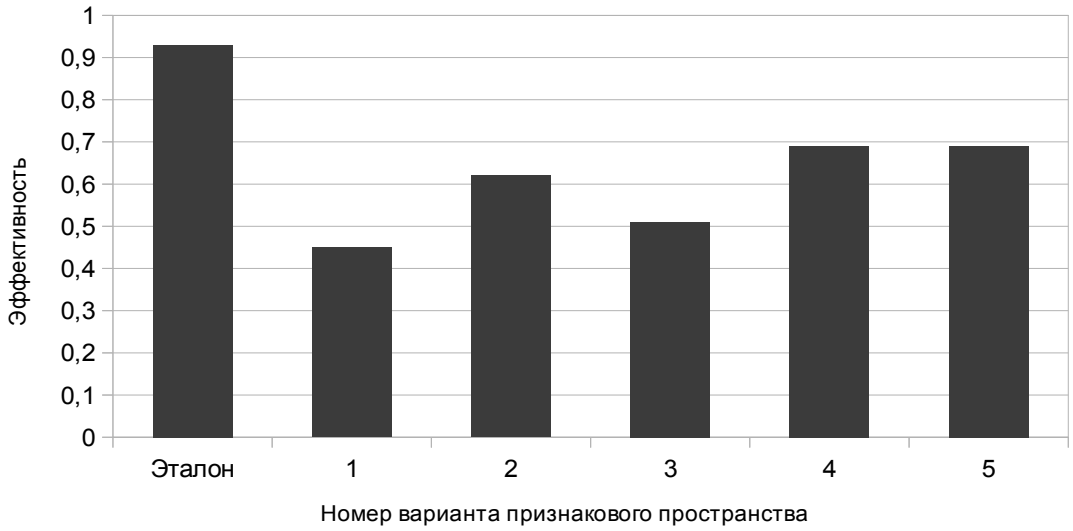


Рисунок 2. Диаграммы результатов тестирования классификатора на основе определения ближайшего класса в Евклидовой метрика

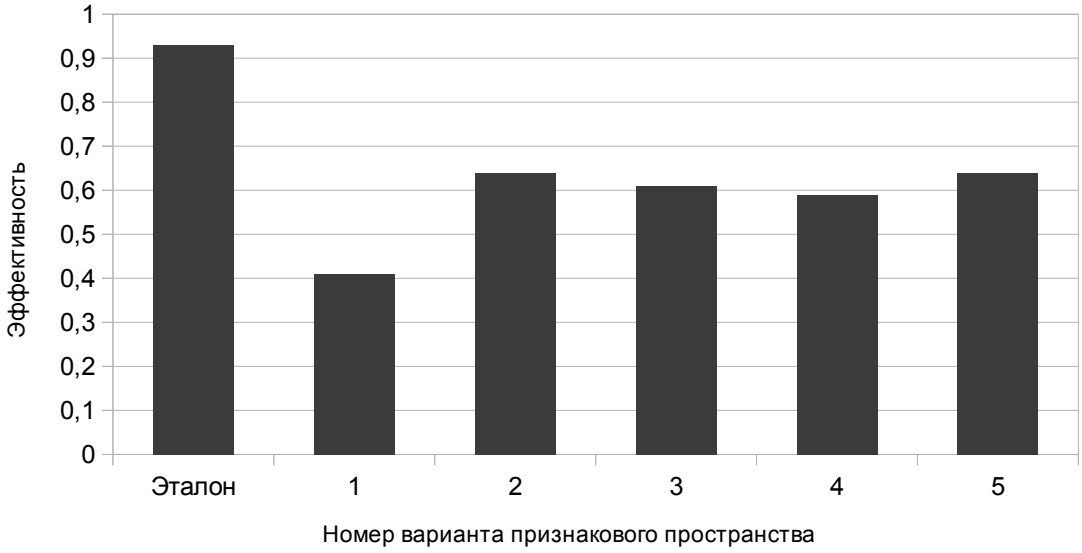


Рисунок 2. Диаграммы результатов тестирования классификатора на основе определения ближайшего класса в метрике Махаланобиса

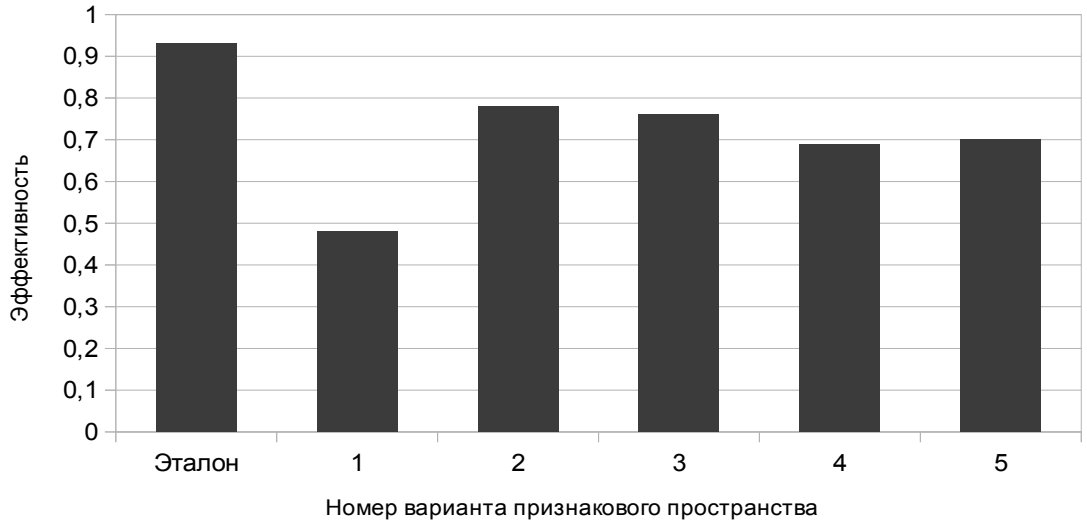


Рисунок 4. Диаграммы результатов тестирования классификатора на основе определения ближайшего класса в косинусной мере

Результаты тестирования алгоритмов классификации Таблица

Пространство признаков	Байесовский классификатор	Классификатор на основе поиска ближайшего класса в пространстве признаков			K ближайших соседей. Косинусная мера. $K = 15$
		Евклидова метрика	Метрика Махаланобиса	Косинус-ная мера	
FPGrowth	0,45	0,45	0,41	0,48	0,4
Наличие слова	0,72	0,62	0,64	0,78	0,72
Кол-во слов	-	0,51	0,61	0,76	0,75
TFIDF	-	0,69	0,59	0,69	0,71
IDF	-	0,69	0,64	0,7	0,79

Заключение

Экспериментально показана неэффективность использования пространства признаков на основе множества часто встречающихся слов по сравнению с пространством признаков, основанным на наличии слова.

Также показано, что наибольшую точность из рассмотренных вариантов классификаторов имеют классификатор на основе поиска ближайшего класса в пространстве признаков, основанным на наличии слова, в косинусной метрике, и классификатор на основе метода k ближайших соседей в пространстве признаков IDF с косинусной метрикой, при $k = 15$.

Литература

1. Воронцов К.В. Машинное обучение: курс лекций // Профессиональный информационно-аналитический ресурс, посвященный машинному обучению, распознаванию образов и интеллектуальному анализу данных. 2011 [Электронный ресурс]. Систем. требования: Программа для просмотра PDF. - URL: http://www.machinelearning.ru/wiki/images/6/6d/Voron-ML-1.pdf (дата обращения: 7.12.2013).
2. Florian Verhein., Frequent Pattern Growth (FP-Growth) Algorithm Outline, 2008. – 10 с.
3. О программе mystem. [Электронный ресурс]. – URL: http://company.yandex.ru/technologies/mystem/ (дата обращения: 7.12.2013).
4. TF-IDF // Википедия, свободная энциклопедия. 2013 [Электронный ресурс]. – URL: http://ru.wikipedia.org/wiki/TF-IDF (дата обращения: 7.12.2013).
5. k-nearest neighbors algorithm // Википедия, свободная энциклопедия. 2013 [Электронный ресурс]. – URL: http://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm (дата обращения: 7.12.2013).

УДК 519.168

АЛГОРИТМИЗАЦИЯ ОПТИМАЛЬНОГО ЦЕЛЕРАСПРЕДЕЛЕНИЯ МЕТОДАМИ ОБЪЕКТНО-ОРИЕНТИРОВАННОГО ПРОГРАММИРОВАНИЯ

С.А. Демиденко, Р.К. Колотов, А.П. Хабаров, С.Л. Старчак

Ключевые слова: *целочисленная оптимизация, метод ветвей и границ, целевая функция, объектно-ориентированный подход.*

Key words: *integer optimization, branch and bound method, objective function, object-oriented approach.*

Аннотация: *Предложено решение прагматической задачи, состоящей в разработке комбинаторного алгоритма и его программной реализации методами ООП. Приведена математическая постановка оптимизационной задачи, решение которой было обеспечено в рамках метода ветвей и границ, а программная реализация выполнена процедурным методом. Дано описание алгоритма неполного перебора, адаптированного*

под особенности объектно-ориентированного подхода. Описаны классы объектов, синтезирована диаграмма классов для эффективной реализации комбинаторного алгоритма. Определены направления совершенствования комбинаторного алгоритма оптимизации в интересах полного учета дифференциации огневых средств по канальности и пропускной способности.

При исследовании эффективности функционирования системы, включающей комплексы высокоточного оружия, возникает необходимость учитывать качество планирования их применения по множеству целей. При прочих равных условиях максимальная эффективность системы достигается в случае, когда в результате планирования обеспечивается оптимальное целераспределение [1-4]. Применительно к рассматриваемой системе целераспределение осуществляется на основе результатов априорной оценки боевых возможностей комплексов по поражению множества целей. Результаты оценки, в обобщенном виде, представляют множество возможных моментов обстрелов целей [1]:

$$M = \{(k, i_k, j_k, p_k, t_k, b_k)\},$$

где k – номер элемента массива боевых возможностей (номер момента обстрела),

$k \in K_s$, $K_s = \{1, \ldots, K\}$;

i_k – номерцели, $i_k \in I_s$, $I_s = \{1, \ldots, I\}$;

j_k – номер используемого для обстрела огневого средства, $j_k \in J_s$, $J_s = \{1, \ldots, J\}$;

p_k – вероятность поражения цели i_k;

t_k – момент поражения цели i_k;

b_k – количество боеприпасов, расходуемых огневым средством j_k при атаке цели i_k.

Целераспределение [2, 3] состоит в определение плана X обстрела множества целей:

$$X = (x_1, \ldots, x_K),$$

где x_k – переменная, принимающая значение 1, если момент обстрела с номером k включается в план, и 0, если не включается.

Для построения плана, с учетом задачи исследований, необходимо оптимизировать одну из следующих целевых функций:

взвешенное математическое ожидание числа поражаемых целей [2, 3]

, (1)

или взвешенное математическое ожидание числа целей, которые уничтожены и их состояние подтверждено по результатам оценки факта поражения [4]

, (2)

где K_{Bi} – коэффициент важности цели i;

$p_{\text{офпi}}$ – вероятность правильной оценки факта поражения цели i.

Оптимизация выполняется при следующих условиях.

Ограничение на боекомплект

$$\forall j \in J_S \sum_{\forall k \in K_S : j_k = j} b_k x_k \leq S_j \quad , \tag{3}$$

где S_j – боекомплект огневого средства j.

Ограничение на время: для любых $k \in K_s$ и $k' \in K_s$, удовлетворяющих следующим условиям

$$k \neq k', x_k \neq 0, x_k' \neq 0, \tag{4}$$

должно выполняться хотя бы одно из следующих неравенств

$$j_k \neq j_k', \tag{5}$$

$$|t_k' - t_k| > \Delta t, \tag{6}$$

где Δt – интервал, необходимый для восстановления готовности огневого средства к очередному обстрелу.

Ограничение на количество обстрелов одной и той же цели

$$\forall i \in I_S \sum_{\forall k \in K_S : i_k = i} x_k \le \mu_i \quad , \tag{7}$$

где μ_i – максимальное количество обстрелов цели i.

Задача (1)-(7) относятся к классу задач дискретной оптимизации [2-4], причем их решение представляется целесообразным в рамках комбинаторных методов [2-4, 6]. Комбинаторная оптимизация – отыскание среди структурированного конечного множества альтернатив наилучшего (относительно заданного критерия) подмножества объектов. Структурные особенности заданного конечного множества служат исходной предпосылкой для построения некоторого системно упорядоченного метода решения задачи, который используется вместо прямого перебора и сравнения всех альтернативных вариантов [2-3].

Большую часть комбинаторных методов образуют методы типа ветвей и границ. Одним из перспективных направлений в их развитии является именно использование специфики решаемых задач. Наиболее перспективными считаются гибридные методы, основанные на использовании различных подходов. Универсальность МВГ и их априорная алгоритмическая неопределенность допускают различные модификации, модернизации, комбинации с другими методами, подходами, приемами [2-4].

Общеизвестно, что почти всегда найдется специальный метод решения, который окажется для данной задачи более эффективным, чем общий метод решения. Фактически же общий метод, адаптированный к структурным особенностям задачи, обычно представляет пример более эффективного специального метода, а эффективность конкретного алгоритма зависит от его эвристической насыщенности [2-3].

Применительно к рассматриваемой задаче (1)-(7) правила ветвления, основное и дополнительное правила отсечения, а так же комбинаторные алгоритмы, реализующие поиск экстремума неполным перебором, изложены в [2-4]. Особенностью их программной реализации является использование процедурного подхода. В процессе исследований и разработки программной реализации имитационной модели [1] возникла необходимость реализации решения задачи с использованием технологий ООП [7].

Основные построения модернизируемого комбинаторного алгоритма предполагают максимизацию целевой функции (1) проводить в два этапа.

На первом этапе производится выбор метода декомпозиции, т.е. способа формирования плана. Список целей L предварительно упорядочивается по убыванию коэффициента важности:

$$L = (l_1, \ldots, l_I),$$

$$l_1 \in I_s, \ldots, l_I \in I_s,$$

.

Процедура построения плана первым методом декомпозиции в качестве параметра использует номер $q \in I_s$ цели в списке L, с которой начнется построение плана, и предполагает следующие действия.

1) Инициализируются переменные, необходимые для учета расхода боекомплекта

$$\forall j \in J_S \; S_j^c = S_j .$$

Всем переменным x_k ($k \in K_s$) присваиваются нулевые значения, переменная q' принимает значение равное q.

2) Для текущей цели r ($r = l_{q'}$) предпринимается попытка включить в план максимальное число моментов обстрелов μ_r, причем их просмотр производится в порядке убывания вероятности поражения. Очередной просматриваемый момент обстрела d ($d \in K_s$, $x_d = 0$) проверяется на ограничения боекомплекта $S_{j_d}^c - b_d \geq 0$ и времени: для

любых $k' \in K_s$, удовлетворяющих условию $x_{k'} \neq 0$, должно выполняться хотя бы одно из

следующих неравенств: $j_d \neq j_{k'}$ или $|t_{k'} - t_d| > \Delta t$.

Если проверка ограничений оказалась успешной, то переменной x_d присваивается значение 1 (момент d включается в план) и учитывается расход боекомплекта

$$S^c_{j_d} = S^c_{j_d} - b_d .$$

3) Если просмотрены не все цели, то выбирается следующая цель из списка и производится переход на шаг 2. Номер следующей цели вычисляется по формуле

$$q' = (q' + 1) \bmod (I + 1).$$

Если просмотрены все цели, то вычисляется целевая функция по формуле (1) или (2) и работа процедуры заканчивается.

Изложенный алгоритм дает хорошее приближение при малой жесткости временных ограничений, заданных формулами (4), (5), (6), и существенно неравных коэффициентах важности. При увеличении жесткости временных ограничений возрастает погрешность приближенного решения, получаемого первым методом декомпозиции. Это обусловлено следующим фактом: чем сильнее ограничения, тем «дороже» каждое назначение обслуживания. Т.е. при включении обстрела текущей цели в план существенно ограничиваются возможности обстрелов следующих за текущей целей.

Чтобы повысить точность получаемого решения при высокой жесткости временных ограничений используется второй метод декомпозиции. Процедура построения плана вторым методом аналогична описанной выше. Отличие ее состоит в том, что выполняется множество обходов списка целей L, начиная с цели q. При этом в одном обходе производятся попытки назначить каждой цели только по одному моменту обстрела. Работа процедуры заканчивается, когда не остаётся моментов, которые еще можно добавить в план.

Выбор метода декомпозиции (первый этап) осуществляется следующим образом. При помощи каждой из процедур, описанных выше, строится план (параметр q равен 1 для обеих процедур). Затем сравниваются значения целевых функций. Выбирается тот метод декомпозиции, который обеспечил получение лучшего результата. План, полученный этим методом, фиксируется.

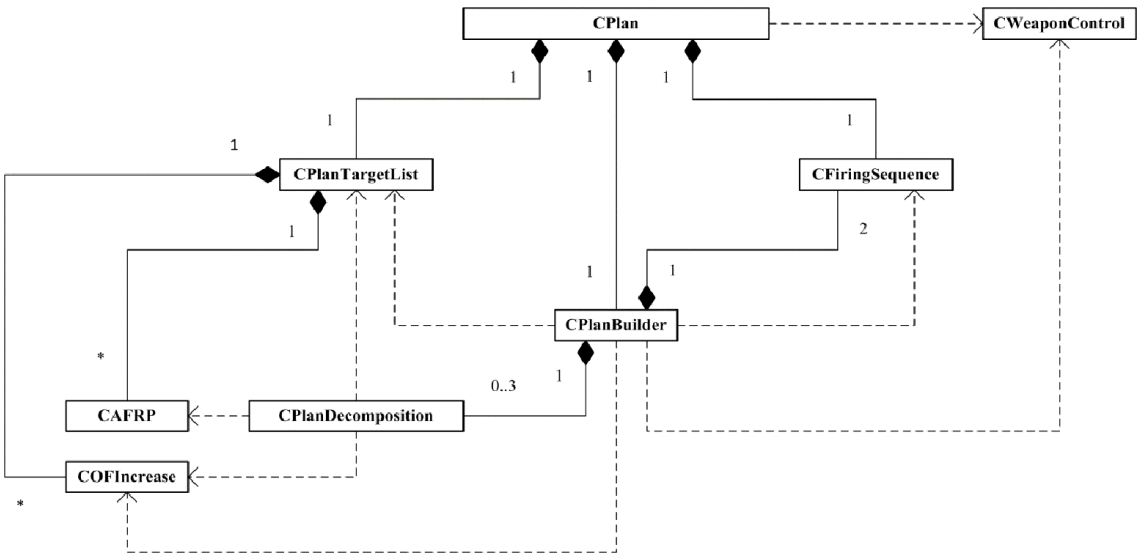
На втором этапе максимизации значения целевой функции происходит формирование выбранным методом декомпозиции планов для $q = 2, \ldots, I$. Из этих планов

и плана, зафиксированного на первом этапе, выбирается тот, значение целевой функции для которого максимально. Данный план и считается оптимальным.

Программная реализация алгоритма оптимального целераспределения выполнена в рамках объектно-ориентированного подхода. При проектировании системы классов учитывался тот факт, что задачи оценки состояния огневого средства, расчета оптимального плана и применения группировки огневых средств в соответствии с этим планом неразрывно связаны между собой и должны представлять собой одну подсистему – подсистему управления огневыми средствами.

На рисунке изображены классы, необходимые для решения задачи построения оптимального плана.

Класс CPlan представляет всю подсистему управления огневыми средствами. Класс CWeaponControl управляет огневыми средствами, выполняет оценку их состояния, формирует частные массивы боевых возможностей, которые содержат только моменты обстрелов для какой-либо одной цели. Список целей описан классом CPlanTargetList. Данный список хранит для каждой цели ее состояние (ожидает обстрела, находится в процессе обстрела или обстрел завершен) и частный массив боевых возможностей. Также каждой цели соответствует свой объект класса COFIncrease. Данный объект хранит вероятности поражения цели для каждого момента обстрела, включаемого в формируемый план. CPlanDecomposition управляет порядком обхода списка целей в зависимости от используемого метода декомпозиции. Классом CFiringSequence представлен план обстрелов. Класс CPlanBuilder является ключевым для подсистемы планирования. Его единственная функция – формирование оптимального плана.



Задача (1)-(7), реализация которой выполнена с помощью объектно-ориентированного программирования, не учитывает двух важных факторов. Первый фактор – зависимость времени восстановления от параметров движения цели и от типа огневого средства. В текущем варианте постановки задачи формула (6) задает одно и то же время восстановления для всех типов средств и для всех целей. Второй фактор – канальность огневого средства. Т.е. при формировании плана должна учитываться возможность обстрела одним огневым средством одновременно нескольких целей в зависимости от числа каналов.

Таким образом, дальнейшим направлением развития является совершенствование алгоритма в направлении учета указанных выше факторов. Это потребует постановки задачи в более общем виде, а также модификации системы классов.

Литература

1. Поисковые исследования методов обеспечения контроля за потоками информации в распределённых АСУ специального назначения. НТО о НИР «Скаут-АСУ». М.: ВИ МГТУ им. Н.Э. Баумана, 2012. 103 с.

2. Вениаминов С.С. Поиск и контроль высокоорбитальных космических объектов: Дис....док. техн. наук. Москва, 1987. 319 с.

3. Вениаминов С.С. Введение в теорию планирования поиска космического объекта по неточной априорной информации о его орбите. М.: ИКИ РАН, 2010. 140 с.

4. Старчак С.Л. Комплекс методов и моделей для обоснования состава и характеристик группировки: Дис....док. техн. наук. Москва, 2008. 348 с.

5. Теория вероятностей: Учеб. для вузов /А.В.Печинкин [и др.] М.: Изд-во МГТУ им. Н.Э. Баумана, 2004. 456 с.

6. Галкина В.А. Дискретная математика: комбинаторная оптимизация на графах.
М.: Гелиос АРВ, 2003. 232 с.

7. Буч Г., Рамбо Д., Якобсон И. Язык UML. Руководство пользователя. 2-е изд. М.: ДМК Пресс, 2006. 496 с.

УДК 004.415.25

API-ТЕХНОЛОГИЯ РАЗРАБОТКИ ПОЛЬЗОВАТЕЛЬСКИХ ИНТЕРФЕЙСОВ НА ПРИМЕРЕ API «В КОНТАКТЕ»

В.В. Кожушко, В.В. Гуренко

Ключевые слова: технология программирования, API-технология, интерфейс пользователя, мобильное приложение, запрос.

Key words: technology of programming, API technology, user interface, mobile application, request.

Аннотация: Проводится сравнительный анализ традиционного подхода к разработке прикладных приложений и технологии с использованием API. Предлагается система критериев для оценки технологий по функциональным характеристикам. Отмечены достоинства и недостатки API-технологии. Уделено внимание использованию методов API на примере сетевых приложений для социальной сети «ВКонтакте».

Введение

Стремительное развитие информационных технологий сопровождается постоянным обновлением существующих методов и средств разработки программных приложений, а также созданием новых, позволяющих существенно упростить процесс проектирования и детализации, заметно повысить производительность программного обеспечения, расширить его функциональные возможности, обеспечить многоплатформенные реализации. Одна из подобных технологий получила наименование API ([эй-пи-ай], от англ. Application Programming Interface – интерфейс программирования приложений или интерфейс прикладного программирования).

API представляет собой набор готовых классов и методов в виде процедур и функций, а также структур данных и констант, предоставляемых библиотекой или сервисом для применения во внешних программных продуктах [1]. Разработчику предлагается использовать наборы готовых кодов, позволяющих освободиться от рутинной работы по программированию сложных и громоздких промежуточных алгоритмов, а следовательно, сконцентрироваться на основной идее приложения, повысить его надежность, снизить трудоемкость и затраты на всех этапах разработки и тестирования.

Сравнительная оценка подходов к разработке программных интерфейсов

С точки зрения результативности процесса проектирования представляется целесообразным сопоставить традиционный подход и технологию с использованием API по следующим параметрам:

– сложность разработки;

– требуемые средства доступа к данным;

– сетевой трафик;

– затрачиваемые вычислительные ресурсы.

На перечисленные параметры существенное влияние оказывает скорость передачи данных. Для исключения связанных с ней ошибок необходимо использовать устойчивое сетевое соединение со скоростью не ниже 1 Мб/с. С целью получения в дальнейшем достоверных данных сравнительного анализа версий программного продукта, полученного при традиционном подходе и с использованием API, было произведено измерение скорости сетевой передачи при загрузке и выгрузке данных. Использовалось мобильное устройство Samsung Galaxy S-II GT-9100i с предустановленной производителем ОС Android версии 4.1.2 и сервисное приложение Speedtest версии 3.0.2 (URL: https://play.google.com/store/apps/details?id=org.zwanoo.android.speedtest), работающее под указанной ОС. Вид экрана при запущенном приложении показан на рисунке 1.

Рисунок 1. Тестирование скорости передачи данных (приложение Speedtest)

В ходе трех тестовых замеров были получены значения, обработанные путем расчета математического ожидания и дисперсии по формулам (1) и (2) соответственно:

$$M[X] = \frac{1}{n}\sum_{i=1}^{n} x_i \qquad (1)$$

$$D[X] = M[X^2] - (M[X])^2 \qquad (2)$$

Результаты сведены в таблицу 1.

Результаты измерения скорости передачи данных Таблица 1

	RTT*, мс	**Скорость загрузки, Мб/с**	**Скорость выгрузки, Мб/с**
Тест 1	14,00	9,51	12,16

Тест 2	12,00	8,98	12,42
Тест 3	11,00	10,34	12,17
***M*[*X*]**	12,33	9,61	12,25
***D*[*X*]**	1,64	0,31	0,015

* RTT, от англ. Round Trip Time – интервал времени между отправкой запроса и получением ответа при сетевом обмене пакетами данных.

Сопоставим подходы по *сложности разработки*. К примеру, метод получения информации с веб-страницы будет включать ряд шагов, различающихся в случае традиционного подхода и при использовании API-технологии (таблица 2).

Последовательность шагов метода получения данных с веб-страницы Таблица 2

Традиционный подход	**API-технология**
1. Загрузка веб-страницы.	1. Запрос серверу с указанием имени и параметров Get-метода.
2. Поиск объекта на странице.	2. Обработка запроса сервером.
3. Извлечение обработчика объекта.	3. Получение пакетов структурированных данных от сервера.
4. Выполнение кода обработчика объекта.	4. Обработка пакетов с извлечением данных, запрошенных пользователем.
5. Открытие страницы с искомой информацией.	
6. Извлечение требуемых данных.	

Как видно из таблицы, при использовании API количество шагов меньше и их сложность ниже. Следовательно, получение требуемой информации, а также ее обработка происходят за меньшее время.

Различия классического подхода и применения API с точки зрения *требуемых средств доступа к данным* отражены в таблице 3.

Средства получения доступа к данным Таблица 3

Традиционный подход	**API-технология**
1. Пользовательский интерфейс.	1. Пользовательский интерфейс.
2. Программный код извлечения данных для пользовательского интерфейса.	2. Код для отправки запросов к серверам и получения информации.
3. Программный код извлечения обработчиков объектов.	3. Код для обработки данных.
4. Программный код извлечения необходимых данных.	
5. Программный код обработки запроса.	
6. Программные средства, необходимые для конкретной платформы.	

Необходимые средства минимальны при использовании API, тогда как при традиционном подходе требуются объемные и сложные алгоритмы для извлечения данных с веб-страниц (парсинга), учитывающие особенности веб-страниц каждого типа. Следовательно, увеличивается время исполнения программного кода, а разработчик вынужден учитывать особенности программирования под определенную платформу.

Рассмотрим особенности подходов с точки зрения затрачиваемого *трафика*, что особенно важно при разработке сетевых приложений. Измерение трафика было произведено с помощью приложения Avast! Mobile security версии 3.0.6572 (https://play.google.com/store/apps/details?id=com.avast.android.mobilesecurity). Вид экрана при запущенном приложении показан на рисунке 2.

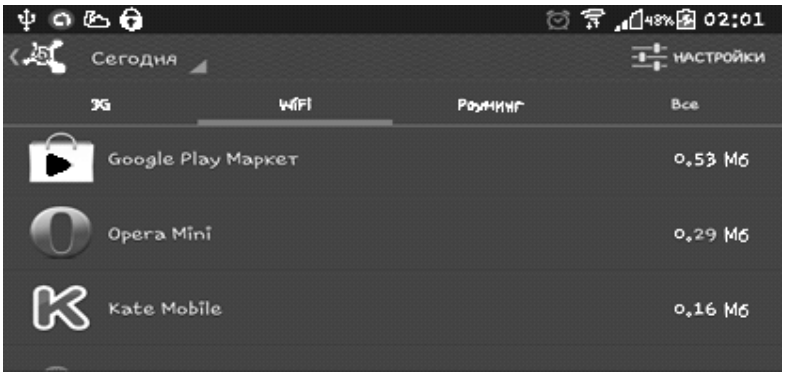


Рисунок 2. Измерение сетевого трафика в приложении Avast! Mobile security

Было отправлено по 3 сообщения («Тестовое сообщение 1.», «Тестовое сообщение 2.», «Тестовое сообщение 3.») через браузеры Opera Mini, Boat Browser и клиент Kate Mobile, использующий API. Учитывался трафик на загрузку профиля и новостей. Результаты измерений представлены в таблице 4.

Результаты измерений сетевого трафика Таблица 4

Приложение	**Версия сайта**	**Отправка сообщений**	**Объем трафика**
Opera Mini 7.5.3	мобильная	0.07 МБ	0,29 МБ
Boat Browser 6.4.1	полная	0.27 МБ	1,02 МБ
Kate Mobile 8.6 (API)	API	0.02 МБ	0,16 МБ

Результаты измерений наглядно показывают, что при использовании API расход трафика ощутимо снижается, поскольку имеют место только передача текста сообщения, идентификатора пользователя и ключа доступа и получение ответа от сервера [2]. В браузерных версиях, как мобильной, так и полной, дополнительно загружаются объекты интерфейса веб-страниц сайта.

Порядок отправки сообщений при двух рассматриваемых подходах можно охарактеризовать следующими этапами [3]:

– при традиционном подходе: а) открытие соединения, б) отправка запроса на получение страницы, в) загруза всей (!) страницы, г) закрытие соединения;

– при использовании API: а) открытие соединения, б) отправка POST- или GET-запроса, в) получение минимальных структурированных данных от сервера, г) закрытие соединения.

Видно, что этапы схожи, однако при традиционном подходе веб-страница загружается полностью для ее последующей обработки, в то время как при использовании API-технологии сервер присылает минимальный необходимый набор данных.

С точки зрения *затрат вычислительных ресурсов* следует отметить, что при традиционном подходе вся работа с информацией происходит на стороне клиента, что в некоторых случаях может занять продолжительное время. При использовании технологии API обработка запроса, сбор и структуризация данных происходят на стороне сервера, и устройство пользователя освобождается от самого ресурсоемкого этапа вычислений [2]. Как правило, серверное оборудование обладает высокой вычислительной мощностью и обеспечивает скоростную обработку данных. Следовательно, в случае API даже самым непроизводительным оконечным устройствам может быть предоставлена возможность удовлетворительной работы с большими объемами данных.

Преимущества API-технологии

Подведем итоги проведенному анализу по четырем наиболее значимым критериям. Он показывает, что применение интерфейса программирования приложений дает ряд преимуществ.

1. ***Простота.*** Если программное обеспечение разрабатывается по классической технологии, то приходится самостоятельно разрабатывать и отлаживать программный код, нередко алгоритмически громоздкий, для загрузки всех требуемых данных с веб-страниц. В случае API необходим лишь запрос на сервер с указанием имени метода и параметров. В ответ сервер предоставит запрошенные данные в структурированном виде.

2. ***Гибкость.*** Технология API универсальна. Она позволяет использовать свои методы на разном оборудовании и на разных системных платформах. От разработчика требуется создание только:

- пользовательского интерфейса;
- кода для отправки запросов;
- кода для загрузки полученных данных.

Итог работы методов API одинаков на разных платформах, причем как по затратам времени, так и по результатам, а при традиционном подходе может варьироваться из-за особенностей ОС.

3. ***Снижение времени разработки и отладки.*** Очевидно, поскольку объем программного кода при использовании API заметно сокращается, то и затраты времени на проектирование и отладку приложений соответственно уменьшаются.

4. ***Вычислительные ресурсы.*** Поскольку методы API выполняются со стороны сервера, то устройство, на котором запущено приложение, освобождается от части нагрузки. В случае высокопроизводительных серверных сервисов обработка данных ускоряется, повышается эффективность работы «слабых» устройств пользователей с большими массивами данных. Таким образом, возрастает производительность разработанных программных продуктов. Следует так же отметить, что при использовании мобильных устройств экономится энергия аккумуляторных батарей, и сами устройства могут дольше функционировать в автономном режиме.

5. ***Регулярные обновления.*** Методы и средства API регулярно обновляются и модифицируются. В результате появляются новые функции и методы, расширяются возможности существующих, исправляются ошибки, повышается стабильность работы готовых приложений.

Недостатки API-технологии

Применение всякой новой технологии не бывает лишено недостатков. Рассмотрим основные, относящиеся к API.

1. ***Зависимость от отказоустойчивости сервисного оборудования.*** В случае недостаточной отказоустойчивости или ненадежности программно-аппаратного резервировании оборудования серверов либо происходит потеря доступа к API, либо API работает некорректно. Несмотря на то, что вероятность сбоев в работе оборудования серверов мала, нельзя исключать из внимания возможность отказа сервиса в обслуживании API-запросов.

2. ***Закрытый исходный код***. Разработчикам практических приложений не всегда хватает возможностей, предоставляемых API. Однако исходные коды API недоступны, и только сотрудники сервиса имеют доступ к работе над улучшением интерфейса программирования приложений. Именно и только они могут вносить изменения и интегрировать в технологию новые средства. Пользователям сервиса API возможность его модифицирования не предоставляется.

Особенности использования API-технологии для доступа к социальной сети «ВКонтакте»

При работе с социальной сетью «ВКонтакте» для получения доступа к методам API, в том числе расширенным, необходимо выполнить следующую последовательность действий [2]:

1) зарегистрировать приложение на сайте *vk.com* и получить идентификатор *APP_ID*;

2) разработать несложный графический интерфейс;

3) разработать программные коды для отправки GET- и POST-запросов и получения данных от сервера;

4) разработать код для загрузки (парсинга) JSON-объекта;

5) пройти авторизацию в социальной сети.

Для вызова метода API необходимо отправить POST- или GET-запрос по протоколу HTTPS на следующий URL-адрес:

https://api.vk.com/method/'METH'?'PARAMS'&access_token='ACCESS_TOKEN',

где METH – имя метода API, PARAMS – его параметры, ACCESS_TOKEN – ключ доступа.

Рассмотрим пример отправки сообщения на так называемую "стену" сети «ВКонтакте». Ниже приводится текст программы универсальной функции для вызова любого метода API на языке Java 6 для ОС Android [4, 5].

```
public static JSONObject Post(final Context context, final String method,
                                                    final List<NameValuePair> params) {
        JSONObject json= null;
    final Handler handler = new Handler() {
        // заголовок ("хэндлер") необходим для обращения к UI из главного потока
        public void handleMessage(android.os.Message msg) {
            Toast.makeText(context, SendParser(json), 3).show();
            //вывод сообщения пользователю о результате
        }
    };
    new Thread(new Runnable() {//создание нового потока
        public void run() {
        UrlEncodedFormEntity entity = null;
        try {
            entity = new UrlEncodedFormEntity(params, "UTF-8");
            //кодировка параметров в UTF8
            HttpPost request = new HttpPost("https://api.vk.com/method/"
```

```
                                                                    +method);
                        //описание переменной POST-запроса
                request.setEntity(entity); //добавление параметров в зарпос
                HttpClient client = new DefaultHttpClient();
                //описание клиента, который работает по HTTP/HTTPS
                HttpResponse response = null;
                response = client.execute(request);
                //отправка запроса
                HttpEntity entry = response.getEntity();
                //получение данных от сервера
                String responseText = null;
                responseText = EntityUtils.toString(entry);
                JSONObject json = new JSONObject(responseText);
                //преобразование строки в JSON-объект
                //Исключения
                } catch (JSONException e) {
                        Log.e(TAG_Send_Error, e.toString());
                } catch (UnsupportedEncodingException e1) {
                        Log.e(TAG_Send_Error, e1.toString());
                } catch (ClientProtocolException e) {
                        Log.e(TAG_Send_Error, e.toString());
                } catch (IOException e) {
                        Log.e(TAG_Send_Error, e.toString());
                } catch (ParseException e) {
                        Log.e(TAG_Send_Error, e.toString());
                }
                handler.sendEmptyMessage(0);
                return json;//возврат полученного JSON-объекта
            }
        }).start(); //запуск потока
}
```

Поскольку функция принимает коллекцию List<NameValuePair>, требуется собрать все необходимые параметры. Для отправки сообщения на "стену":

```
List<NameValuePair> params = new ArrayList<NameValuePair>(4);
params.add(new BasicNameValuePair("owner_id", "0"));
params.add(new BasicNameValuePair("message", "sometext");
params.add(new BasicNameValuePair("v", "5.5"));
params.add(new BasicNameValuePair("access_token", "token"));
JSONObject js=Post(GetApplicationContext,"wall.post",params);
```

Минимальный необходимый набор параметров: **owner_id** – идентификатор адресата, **message** – текст сообщения, **v** – версия API, **access_token** – ключ доступа, полученный при авторизации [2].

Если в запросе отсутствуют ошибки, то сервер вернет JSON-объект с идентификатором сообщения (например, {"***response***": {"***post_id***": 4298}}), иначе возвращается код ошибки с подробным ее описанием [2].

Метод для загрузки полученного JSON-объекта:

```
public static String sendParser(JSONObject json) {
        String res = null;
        try {
            if (json.has("error")) {//есть ли ошибка?
                json = json.getJSONObject("error");
                int err = json.getInt("error_code");
                res = String.valueOf(err) + ": ";
                switch (err) {
                case 100:
                    res += "Пропущен один из параметров";
                    break;
                default:
                    res += json.getString("error_msg");
                    break;
                }
            } else {
                res = "Успешно!";
            }
        } catch (JSONException e) {//исключение
            e.printStackTrace();
        }
        return res;
    }
```

Для отправки личного сообщения указывается имя метода **messages.send**. Минимальный набор параметров остается таким же, как в случае отправки сообщения на «стену» (**wall.post**), однако имя параметра **owner_id** меняется на **user_id** [2]**.**

Выводы

Применение API-технологии позволяет существенно упростить разработку приложений и сделать их более надежными, снизить затраты времени на проектирование и отладку прикладных программ, разгрузить устройства, на которых работает программный продукт, от части вычислений, экономить сетевой трафик, что немаловажно для мобильных приложений. Технология универсальна, ее применение дает сходные результаты на различных платформах. Главным недостатком является недоступность и, как следствие, невозможность модификации предоставляемого программного кода, что отчасти компенсируется регулярными обновлениями сервиса API.

Литература

1. Интерфейс программирования приложений. // Википедия, свободная энциклопедия. [Электронный ресурс]. – URL: http://ru.wikipedia.org/wiki/Интерфейс_программирования_приложений (дата обращения: 12.02.2014).
2. Документация по API социальной сети «В контакте». [Электронный ресурс]. – URL: http://vk.com/dev/ (дата обращения: 12.02.2014).
3. Таненбаум Э. Компьютерные сети. Изд. 4-е. – СПб.: Питер, 2003. – 992 с.
4. Эккель Б. Философия Java. Библиотека программиста. Изд. 4-е. – СПб.: Питер, 2009. – 640 с.
5. Документация по операционной системе Android. [Электронный ресурс]. – URL: http://developer.android.com/index.html (дата обращения 12.02.2014).

УДК 004.72

ПРОГРАММНО-КОНФИГУРИРУЕМЫЕ СЕТИ SDN – ПРИНЦИПИАЛЬНО НОВЫЙ ПОДХОД К ПОСТРОЕНИЮ СЕТЕЙ

А.Е. Коломеец, Л.В. Сурков

Ключевые слова: сеть, ПКС, контроллер, фонд открытых сетевых технологий

Key words: network, SDN, controller, ONF, OpenFlow

Аннотация: Рассмотрены основные проблемы современных компьютерных сетей и пути их решения. Одним из таких подходов является развитие программно-конфигурируемых

SDN – сетей (software defined networks), основной принцип которых – разделение уровня управления от уровня данных. Для этого вся логика работы компьютерных сетей вынесена на отдельные устройства – контроллеры, а коммутаторы непосредственно используются для быстрого продвижения пакетов. В настоящее время для реализации данной технологии применяется разрабатываемый фондом открытых сетей (ONF) протокол OpenFlow. В статье описаны механизмы работы программно-конфигурируемых сетей (ПКС) на базе протокола OpenFlow, и их преимущества относительно архитектуры традиционных компьютерных сетей.

Введение

Компьютерные сети как основополагающая инфраструктура – стратегический фактор развития современных ИТ, однако архитектура сети, основы которой закладывались еще в конце 60-х годов, устарела и уже не всегда способна адекватно и эффективно реагировать на новые информационные потребности общества. Рост количества и разнообразия мобильных устройств, развитие различных технологий беспроводной связи привели к тому, что сегодня число их пользователей превысило число пользователей сетей с фиксированной связью. Однако рост мощности мобильных терминалов стимулирует увеличение вычислительной емкости приложений, что, в свою очередь, требует увеличения пропускной способности каналов связи. Объем мобильного трафика растет в геометрической прогрессии, а виды трафика становятся все более разнообразными. По данным ведущих производителей сетевого оборудования, трафик удваивается примерно каждые девять месяцев, что в ближайшие несколько лет приведет к увеличению нагрузки на несколько порядков. По прогнозам компании Cisco в ближайшие 5 лет объем трафика увеличится в 4 раза, причем, мобильный трафик будет удваиваться ежегодно. К 2015 г. количество устройств в сети, будет в два раза выше, чем население планеты [1].

До недавнего времени действующая архитектура сетей развивалась по методу «ласточкиного гнезда», т.е. по мере выявления проблем к стеку протоколов TCP/IP добавлялся новый, который эту проблему решал. Например, когда появились цифровая сеть с интеграцией служб, объединяющая передачу речи, данных и изображений (ISDN) возникла проблема передачи видеотрафика. Суть проблемы можно свести к следующему: при передаче видео по сетям возникают жесткие требования к управлению качеством сервиса, поскольку необходимо динамично продвигать очень высокоскоростной трафик, не допускающий задержек. Протокол RSVP (Resource Reservation Protocol) решил

проблему резервирования необходимых ресурсов под такой трафик и, соответственно, позволил обеспечить необходимый уровень качества услуг. Однако при использовании данного протокола встала проблема масштабирования сети.

Другой пример – протоколы DHCP и NAT были разработаны для сетей IPv4. Эти протоколы немного решили проблему ограниченности IP адресов в сетях IPv4, но в результате появилась другая проблема – назначение одинаковых IP адресов разному оборудованию в рамках одной сетевой инфраструктуры.

На сегодня число фактически (активно) используемых протоколов более 600, и эта цифра далеко не конечная. Итак, можно выделить следующие проблемы современных компьютерных сетей [1]:

- научно-технические – сегодня невозможно контролировать и надежно предвидеть поведение таких сложных объектов, как глобальные компьютерные сети;
- экономические – сети дороги, сложны и требуют для своего обслуживания высококвалифицированных специалистов;
- проблемы развития – в архитектуре современных сетей имеются существенные барьеры для экспериментирования и создания новых сервисов.

Ответом на кризис компьютерных сетей стало появление принципиально нового подхода к их построению – программно-конфигурируемых сетей (ПКС).

Архитектура ПКС

Концепцию новой сетевой архитектуры программно-конфигурируемых сетей предложили в 2007 году сотрудники Стэнфордского университета [2]. С тех пор сети ПКС развивались преимущественно в научной лаборатории Стэнфорда и Беркли и в промышленно значимых масштабах их еще никто не пробовал. Зарождение данной технологии было связано с несколькими моментами:

- сети традиционной архитектуры проприетарны, закрыты для исследований и практически любых изменений извне. Оборудование разных производителей часто между собой плохо совместимы;
- рост трафика в геометрической прогрессии и тезис о том, что сети нынешней архитектуры не смогут с ним справиться на необходимом уровнем качества;
- рост количества протоколов и их стеков в сети.

Исследователи из Стэнфорда и Беркли предположили, что в компьютерных сетях возможно разделить функции управления и передачи данных.

При передаче данных коммутатор Ethernet подает запрос к таблице коммутации (рисунок 1). Затем на основании полученной информации коммутационная матрица осуществляет дальнейшую обработку и передачу данных на целевой исходный порт.

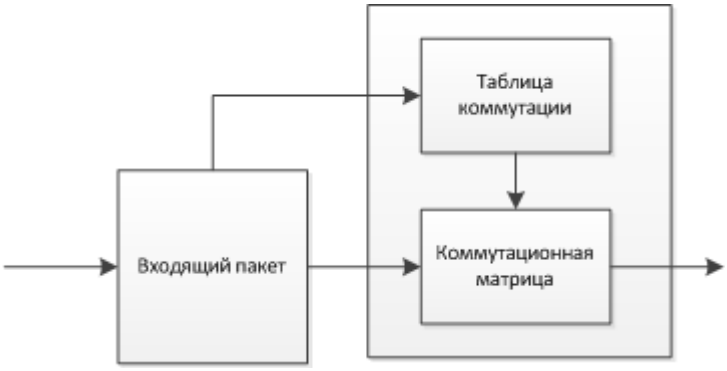


Рисунок 1. Обработка и передача данных в коммутаторе Ethernet

В обычном коммутаторе Ethernet одновременно реализуются и управление и передача данных. Уровень управления представлен встроенным контроллером, уровень передачи данных – таблицей коммутации и коммутационной матрицей. Контроллер обладает некоторыми интеллектуальными функциями, позволяющим ему самому принимать решения о передаче данных на основе информации о структуре сети. Но непосредственно управлять принятием решения нельзя – можно лишь конфигурировать контроллер, задавая определенные наборы правил и приоритетов. Это значительно ограничивает функциональность коммутатора и всей сети [3]. Например, организацию связей «каждый с каждым» в такой сети нельзя построить без сетевого устройства третьего уровня – маршрутизатора. Проблему разделения уровня управления и передачей данных исследователи Стэнфорда и Беркли предложили решить в рамках подхода, получившего название SDN (Software-defined networking).

В архитектуре SDN можно выделить три уровня:

- *инфраструктурный уровень*, предоставляющий набор сетевых устройств (коммутаторов и каналов передачи данных);
- *уровень управления*, включающий в себя сетевую операционную систему, которая обеспечивает приложениям сетевые сервисы и программный интерфейс для управления сетевыми устройствами и сетью;
- *уровень сетевых приложений* для гибкого и эффективного управления сетью.

Протокол OpenFlow

Идея SDN о создании унифицированного, независимого от производителя сетевого оборудования, программно-управляемого интерфейса между контроллером и транспортной средой сети нашла отражение в протоколе OpenFlow, позволяющем пользователям самим определять и контролировать, кто с кем, при каких условиях и с каким качеством может взаимодействовать в сети. До OpenFlow сетевую архитектуру

можно представить как телефонную связь в самом начале 20 века, когда коммутация осуществлялась вручную. Сегодня администратор также вручную настраивает оборудование по заданным параметрам, и любые дальнейшие изменения осуществляются преимущественно на аппаратном уровне. OpenFlow позволяет уйти от такого управления сетью, что положительно сказывается на ее масштабируемости.

В коммутаторе OpenFlow реализован только уровень передачи данных. Вместо контроллера используется гораздо более простое устройство, задача которого состоит в принятии поступающих данных, извлечение их адресов и, если адресат есть в таблице коммутации, немедленной передачи данных коммутационной матрице. Иначе коммутатор по защищенному каналу отправляет запрос на центральный контроллер сети, и на основании полученной от него информации, вносит необходимый изменения в таблицу коммутации, после чего осуществляется обработка полученных данных (оборудование не перенастраивается вручную, а перенастраивается с помощью ПО). Центральный контроллер имеет точную информацию о структуре и топологии сети. Это позволяет оптимизировать продвижение пакетов данных, и, в частности, прокладывать связи «каждый с каждым» на уровне L2, не прибегая к IP-маршрутизации (рисунок 2).

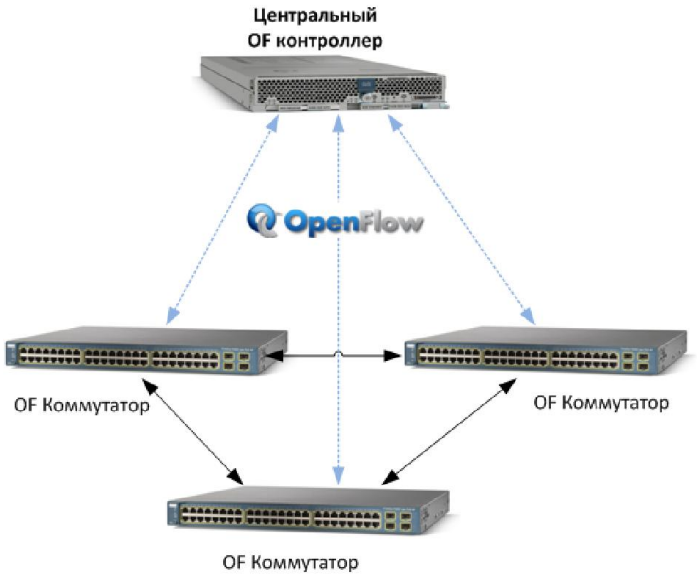


Рисунок 2. Основная топология SDN-сети

Также становится возможным резервировать каналы передачи данных на всем пути от источника до пункта назначения. В результате по сети передаются потоки данных, а не отдельные пакеты. Поэтому в терминологии ПКС такая таблица коммутации получила название FlowTable (таблица потоков). У каждого коммутатора своя уникальная таблица, которую он заполняет только на основании информации, полученной от центрального контроллера.

Механизм работы коммутатора OpenFlow состоит в следующем [4]. У каждого пришедшего пакета «вырезается» заголовок (битовая строка определенной длины). Для этой битовой строки в таблицах потоков, начиная с первой, ищется правило, у которого поле признаков ближе всего соответствует (совпадает) заголовку пакета. При наличии совпадения, над пакетом и его заголовком выполняются преобразования, определяемые набором инструкций, указанных в найденном правиле. Инструкции, ассоциированные с каждой записью таблицы, описывают действия, связанные с пересылкой пакета, модификацией его заголовка, обработкой в таблице групп, обработкой в конвейере и пересылкой пакета на определенный порт коммутатора. Инструкции конвейера обработки позволяют пересылать пакеты в последующие таблицы для дальнейшей обработки и в виде метаданных передавать информацию между таблицами. Инструкции также определяют правила модификации счетчиков, которые могут быть использованы для сбора разнообразной статистики (рисунок 3).

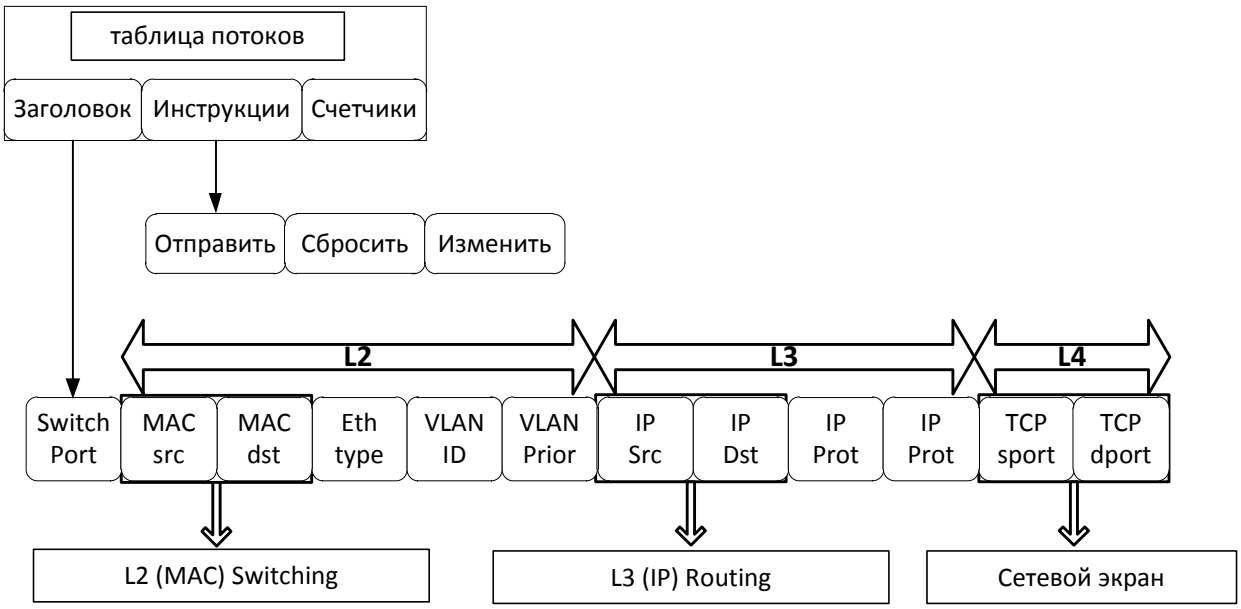


Рисунок 3. Таблица потоков OpenFlow

Если нужного правила в первой таблице не обнаружено, то пакет инкапсулируется и отправляется контроллеру, который формирует соответствующее правило для пакетов данного типа и устанавливает его на коммутаторе (или на наборе управляемых им коммутаторов), либо пакет может быть изменен или сброшен.

Запись о потоке может предписывать переслать пакет в определенный порт (обычный физический порт либо виртуальный, назначенный коммутатором, или зарезервированный виртуальный порт, установленный спецификацией протокола). Зарезервированные виртуальные порты могут определять общие действия пересылки: отправка контроллеру, широковещательная (лавинная) рассылка, пересылка без OpenFlow.

Виртуальные порты, определенные коммутатором, могут точно определять группы агрегирования каналов, туннели или интерфейсы с обратной связью.

Записи о потоках могут также указывать на группы, в которых определяется дополнительная обработка. Группы представляют собой наборы действий для широковещательной рассылки, а также наборы действий пересылки с более сложной семантикой, например быстрое изменение маршрута или агрегирование каналов. Механизм групп позволяет эффективно изменять общие выходные действия для потоков. Таблица групп содержит записи о группах, содержащие список контейнеров действий со специальной семантикой, зависящей от типа группы. Действия в одном или нескольких контейнерах действий применяются к пакетам, отправляемым в группу.

Разработчики коммутаторов могут быть свободны в реализации их внутренней начинки, однако процедура просмотра пакетов и семантика инструкций должны быть для всех одинаковы. Например, в то время как поток может использовать все группы для пересылки в некоторое множество портов, разработчик коммутатора может выбрать для реализации этого единую битовую маску внутри аппаратной таблицы маршрутизации. Другой пример – это процедура просмотра таблиц: конвейер физически может быть реализован с помощью различного количества аппаратных таблиц. Установка, обновление и удаление правил в таблицах потоков коммутатора осуществляются контроллером. Правила могут устанавливаться реактивно (в ответ на пришедшие пакеты) или проактивно (заранее, до прихода пакетов).

Управление данными в OpenFlow осуществляется не на уровне отдельных пакетов, а на уровне их потоков. Правило в коммутаторе OpenFlow устанавливается с участием контроллера только для первого пакета, а затем все остальные пакеты потока его используют.

Протокол OpenFlow поддерживает три типа сообщений [5]: контроллер-коммутатор, асинхронные и симметричные. Сообщения типа контроллер-коммутатор инициируются контроллером и используются для непосредственного управления и слежения за состоянием коммутатора. Сообщения данного типа могут использоваться контроллером для установки параметров конфигурации коммутатора, для сбора статистики, для добавления, удаления и модификации записей в таблицах потоков. Асинхронные сообщения инициируются коммутатором для оповещения контроллера о сетевых событиях (прибытии пакетов или удалении записи из таблицы по тайм-ауту) и изменениях состояния коммутатора или ошибках. Симметричные сообщения могут инициироваться коммутатором или контроллером без запроса и используются при

установлении соединения, а также при измерении задержек, пропускной способности соединения контроллер-коммутатор или для проверки живучести соединения.

В сетях новой архитектуры ПКС все маршрутизаторы и коммутаторы объединяются под управлением Сетевой Операционной Системы (СОС), которая обеспечивает приложениям доступ к управлению сетью и которая постоянно отслеживает конфигурацию средств сети.

В отличие от традиционного толкования термина СОС как операционной системы интегрированной со стеком сетевых протоколов, в данном случае под СОС понимается программная система, обеспечивающая мониторинг, доступ, управление, ресурсами всей сети, а не конкретного узла. СОС формирует данные о состоянии всех ресурсов сети и обеспечивает доступ к ним для приложений управления сетью. Эти приложения управляют разными аспектами функционирования сети, типа построения топологии, принятия маршрутизирующих решений, балансировки нагрузки и т.п.

Протокол OpenFlow решает также проблему зависимости от сетевого оборудования какого-либо конкретного поставщика, поскольку ПКС использует общие абстракции для пересылки пакетов, которые сетевая операционная система использует для управления сетевыми коммутаторами.

В 2011 году альянс компаний Deutsche Telekom, Facebook, Google, Microsoft, Verizon, и Yahoo образовали общественную организацию Open Networking Foundation (ONF), нацеленную на продвижение и стандартизацию программно-конфигурируемых сетей и координирующую развитие открытого сетевого протокола OpenFlow. О значимости нового движения говорит быстрорастущий состав ONF, в которую вошли уже больше 40 крупнейших компаний мира: в числе которых Brocade, Cisco, Citrix, Oracle, Dell, Ericsson, HP, IBM, Juniper, Marvell, NEC, Netgear, NTT, Riverbed и ряд других.

Одна из идей, активно развиваемая в рамках SDN, – это виртуализация сетей с целью более эффективного использования сетевых ресурсов. Под виртуализацией сети понимается изоляция сетевого трафика – группирование (мультиплексирование) нескольких потоков данных с различными характеристиками в рамках одной логической сети, которая может разделять единую физическую сеть с другими логическими сетями или сетевыми срезами (network slices). Каждый такой срез может использовать свою адресацию, свои алгоритмы маршрутизации, управления качеством сервисов и т. д. Виртуализация сети позволяет: повысить эффективность распределения сетевых ресурсов и сбалансировать нагрузку на них; изолировать потоки разных пользователей и приложений в рамках одной физической сети; администраторам разных срезов использовать свои политики маршрутизации и правила управления потоками данных;

проводить эксперименты в сети, используя реальную физическую сетевую инфраструктуру; использовать в каждом срезе только те сервисы, которые необходимы конкретным приложениям.

Одним из примеров виртуализации ресурсов SDN, разделения сети на срезы и управления ими является FlowVisor – программа-посредник (proxy), действующая на уровне между OpenFlow-коммутаторами и различными контроллерами SDN. Посредством FlowVisor можно создавать логические сегменты сети, использующие разные алгоритмы управления потоками данных, обеспечивая изоляцию данных сетей друг от друга. Это означает, что каждый контроллер управляет только своей логической сетью и не может оказывать влияния на функционирование других. Для контроллера, взаимодействующего с оборудованием OpenFlow через FlowVisor, весь обмен сообщениями выглядит так же, как если бы контроллер взаимодействовал с обычной сетью SDN. Всю необходимую модификацию сообщений, требующуюся для поддержки различных изолированных сегментов сети, выполняет FlowVisor. То есть для контроллера логической сети не требуется модификации – это может быть любой контроллер SDN, например сетевая операционная система NOX с произвольным набором программ.

Теоретически неограниченные возможности сетей SDN к расширению позволяют строить реальные облака, масштабируемые в зависимости от решаемых задач. При этом сеть обладает требуемой «интеллектуальностью», необходимой, в частности, для оркестровки работы обширных групп коммутаторов.

Заключение

Отметим потенциальные достоинства архитектуры SDN-сетей относительно традиционных современных сетей:

- уменьшение стоимости развертывания сетей за счет более дешевого оборудования;
- повышение утилизации ресурсов с использованием централизованных политик;
- удобство администрирования и отладки;
- возможность разработки и развития программных сетевых приложений и сервисов пользователем;
- открытость протокола OpenFlow, позволяющая не зависеть от производителей сетевых устройств;
- простота масштабирования.

Теоретически неограниченные возможности сетей SDN к расширению позволяют строить реальные облака, масштабируемые в зависимости от решаемых задач. При этом

сеть обладает требуемой интеллектуальностью, необходимой, для управления работой больших групп коммутаторов.

Литература

1. Смелянский Р.Л. Программно-конфигурируемые сети//Открытые системы [Электронный ресурс]. – URL: http://www.osp.ru/os/2012/09/13032491 (дата обращения: 07.03.2014).
2. Thomas D. Nadeau, Ken Gray, SDN: Software Defined Networks, O'Reilly, 2013.
3. Onix: A distributed control platform for large-scale production networks /T. Koponen [и др.], OSDI, 2010.
4. OpenFlow Tutorial [Электронный ресурс]. – URL: http://archive.openflow.org/wk/index.php/OpenFlow_Tutorial (дата обращения: 07.03.2014).
5. Open network foundation Specification [Электронный ресурс]. –URL: https://www.opennetworking.org/sdn-resources/onf-specifications (дата обращения: 07.03.2014).

УДК 004.424 +004.75+519.171.2

СПОСОБ ПРЕДСТАВЛЕНИЯ ГРАФОВ ДЛЯ РЕШЕНИЯ ЗАДАЧ СТРУКТУРНОГО СИНТЕЗА В ПАРАЛЛЕЛЬНЫХ И РАСПРЕДЕЛЕННЫХ СИСТЕМАХ

Ю.К. Петров, Г.С. Иванова

Ключевые слова: графы, параллелизм, распределенные вычисления, модель акторов, виртуальные машины, легковесные процессы, межпроцессное взаимодействие.

Key words: graphs, parallelism, concurrency, distributed computing, actors model, virtual machines, lightweight processes, green threads, IPC.

Аннотация: В статье описывается новая модель представления графов в программах. Целью являлось получение модели, обеспечивающей высокий уровень параллелизма на всех этапах обработки для ускорения операций, которые производятся над графами. Проведено исследование, которое показало возможность реализации предлагаемой модели с помощью современных технологий виртуализации, в частности – виртуальных машин, использующихся в ряде языков программирования. Сравнительный анализ

посредством временных диаграмм функционирования предлагаемой модели и базовой традиционной показал эффективность разработанной модели.

Введение

Графы используются при решении широкого спектра задач, связанных с анализом и синтезом структур объектов. Так, графы могут быть использованы для реализации древовидных структур в поисковых запросах, семантическом анализе текстов, моделировании сложных устройств, представимых в виде множества состояний.

В обрабатывающей программе графы обычно представляют, таблично задавая отношения смежности или инцидентности вершин и/или ребер. Такой подход был эффективен для однопроцессорных вычислительных систем. В настоящее время в большинстве случаев используются многопроцессорные системы, зачастую распределённые, для которых представление таблицами смежности или инцидентности может оказаться неэффективным.

В [1-3] предлагается представление, рассчитанное на параллельную обработку. При этом:

1. Вычислительная среда предполагается гомогенной;
2. Вершины графа представляются некоторыми пассивными структурами данных;
3. Ответственность за распределение вершин графа лежит на основном (управляющем) узле вычислительной установки.

Таким образом, предлагаемому в [1-3] представлению свойственно централизованное управление структурой графа, что обуславливает синхронность исполняемых над графом операций и не позволяет реализовать алгоритмы, гарантирующие параллелизм на всех стадиях обработки данных, или даже эффективно использовать вычислительные мощности. Исходя из основного типа используемого способа межпроцессного взаимодействия, данное представление далее будем называть *синхронным*.

Для обеспечения более эффективного использования потенциала распределённых систем требуется изменить подход, распределив управление по специализированным процессам, что снимет нагрузку с основного управляющего процесса, обеспечив минимум блокировок.

Предлагаемый в настоящей работе способ представления графов ориентирован на параллельные и распределённые системы. Он основывается на том, что вершины графа представлены легковесными процессами внутри виртуальной машины (в данной работе

для этого была использована нотация языка Erlang [4]). Идея, лежащая в основе данного представления, аналогична идее, заложенной в модели акторов: существует некоторое количество относительно автономных сущностей (в данном случае – вершин), которые могут действовать одновременно и обмениваться информацией друг с другом в процессе решения задачи.

Постановка задачи

Требуется разработать модель представления графов, удовлетворяющую следующим условиям:

1. Обеспечение высокой степени параллелизма при выполнении операций над графом.
2. Применимость к разным типам параллельных систем (SMP, NUMA…).
3. Возможность применения в гетерогенных средах.

Основные концепции предлагаемой модели представления графа

Предлагаемая модель подразумевает концепции, отличные от описанных выше:

1. Ответственность за распределение вершин графа распределена между различными процессами, отвечающими за первичное размещение, групповую коммуникацию, балансировку;

2. Вершины графа являются легковесными процессами внутри виртуальной машины.

На рисунке 1 изображена простейшая схема графа, представленного с помощью данной модели.

В модели используется терминология, принятая в языке программирования Erlang:

- Supervisor – процесс-наблюдатель. Он может не только выполнять вычисления, но и отвечает за отказоустойчивость порождённых им процессов, т.е. процессов, располагающихся ниже его в иерархии данной модели.
- Worker – обычный рабочий процесс, этими процессами представляются вершины графа.
- Node – представляет собой узел, в котором запущены легковесные процессы, реализующие модель. Таким образом, узлом является экземпляр виртуальной машины, т.е. на одном многоядерном процессоре или даже процессорном ядре могут работать несколько узлов.

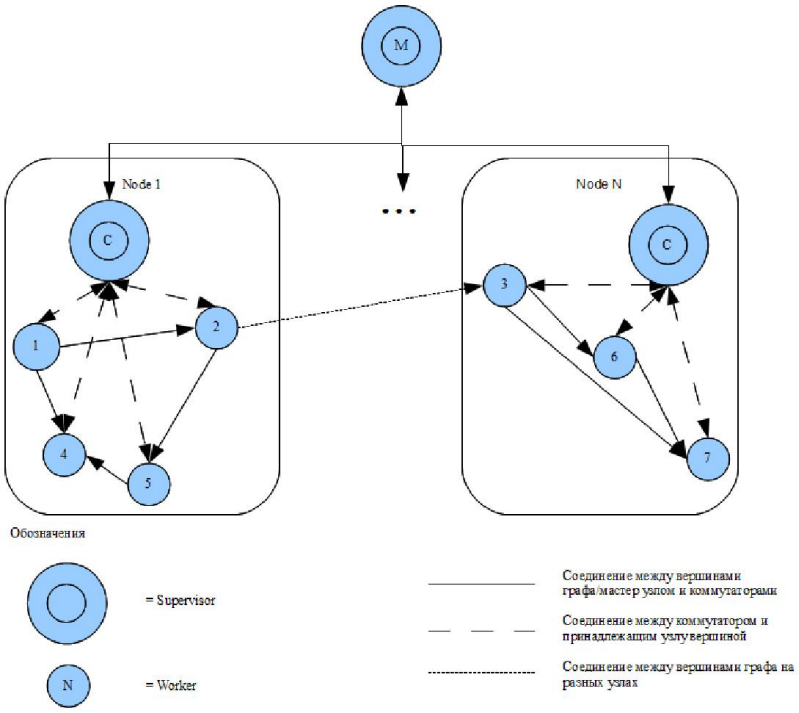


Рисунок 1. Простейшая модель распределённого графа

Иерархия процессов в модели

Для обеспечения максимальной возможности распараллеливания организуем (реализуем) следующие типы процессов:

1. Мастер Master (M) – существует всегда только один мастер, отвечает за взаимодействие пользователя с графом, добавление вершин, первичное размещение графа (при помощи поисковых агентов – см. далее), адресацию узлов, адресацию вершин по номерам узлов. Не может быть завершён, так как является корнем иерархии;

2. Поддерживающий мастер Support Master (SM) – процессы этого типа порождаются мастером, могут существовать на различных виртуальных машинах. Выполняют те же задачи, что и мастер, за исключением первичного размещения узлов, обеспечения отказоустойчивости коммуникаторов и порождения новых поддерживающих мастеров. Может быть завершён;

3. Поисковый агент Search Agent (SA) – порождается мастером для каждого запроса от коммуникатора на выделение группы вершин, работает в виртуальной машине мастера, завершается после выполнения задачи;

4. Коммуникатор Communicator (C) – существующий на каждом узле в единственном экземпляре процесс, отвечает за размещение и адресацию процессов вершин внутри узла, используется для доступа к ним. Может быть завершён только при удалении узла и только после перераспределения принадлежащих узлу вершин. Отвечает за отказоустойчивость вершин графа, в случае аварийного завершения

вершины (или всего узла) коммуникатор обеспечивает восстановление данных, обращаясь за информацией к главному узлу;

5. Вершина графа Vertex (V) – рабочий процесс вершины графа. Содержит в своём состоянии адрес коммуникатора, информацию о вершинах, указывающих на вершину и вершинах, на которые указывает вершина.

Перечисленные процессы могут быть разделены на 3 группы:

1. Глобальные управляющие процессы – процессы, взаимодействующие с графом в целом;
2. Локальные управляющие процессы – процессы, взаимодействующие только с частью графа на узле;
3. Рабочие процессы – вершины.

Полученная иерархия схематично представлена на рисунке 2.

Мастер-узел, реализованный легковесным процессом, в рабочем состоянии содержит две таблицы. Первая таблица содержит информацию об управляемых узлах: номер главного процесса узла и идентификатор соответствующего процесса внутри конкретной виртуальной машины pid. Главным процессом узла может являться поддерживающий мастер или коммуникатор. Вторая таблица отображает распределение вершин по узлам.

Теоретически, эти таблицы могут быть сведены в одну, однако при этом усложнится обеспечение отказоустойчивсти. В случае экстренного завершения узла он может быть просто перезапущен, вершины графа, размещенные на нём ранее, должны быть переразмещены, однако таблица с размещением вершин не изменится. Итого надо внести одно изменение. Если таблица будет всего одна, потребуется исправить каждую строку, где записан pid коммуникатора отказавшего узла.

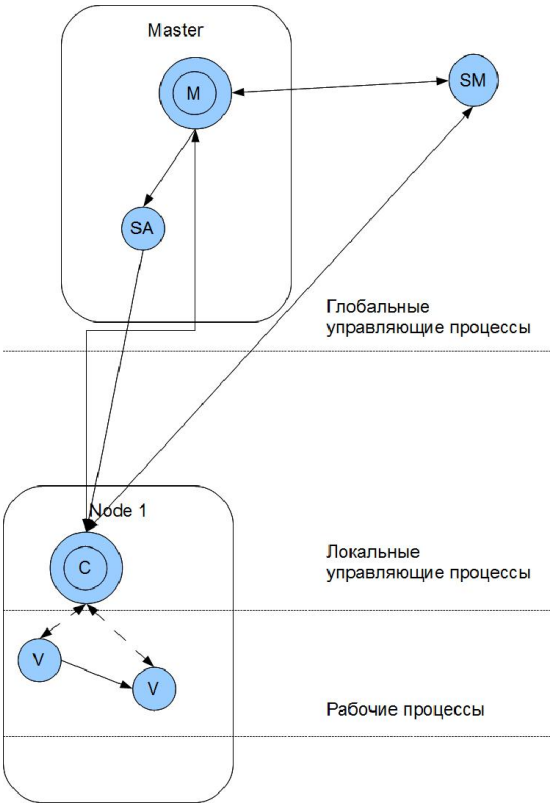


Рисунок 2. Иерархия процессов модели

Поддерживающие мастер-узлы обладают теми же данными, что и мастер-узел в модели без поддерживающих узлов. Поддерживающие мастер-узлы не имеют информации друг о друге, т.к. в этом нет необходимости. Они могут использоваться как удалённые консоли управления, добавляя, удаляя вершины графа, запуская вычисления над графом. При этом отказ одного из поддерживающих узлов никак не отражается на работоспособности: мастер-узел может просто перезапустить отказавший поддерживающий мастер-узел, благодаря асинхронной передаче сообщений отправка сообщения по недействительному адресу (например, от коммуникатора к отказавшему поддерживающему мастер-узлу) не привносит практически никаких накладных расходов.

С точки зрения мастер-узла поддерживающие мастер-узлы почти не отличаются от процессов коммуникаторов, поэтому помещены с ними в одну таблицу. Это обеспечивает быстрое восстановление в случае отказа поддерживающего мастер-узла: мастер-узел ищет pid отказавшего узла (полученный из сообщения экстренного завершения) в таблице узлов, перезапускает процесс, отправляет ему требуемые данные (в зависимости от типа узла) и обновляет информацию в таблице узлов.

Реализация коммуникатора в виде легковесного процесса предполагает всего одну таблицу, отображающую соответствие номеров вершин графа, размещенных на данном узле, pid рабочих процессов, ассоциированных с этими вершинами, количество связей с вершинами данного узла (графа Local) и других узлов (Node2...NodeN для Node1).

Процесс также обладает параметром State, под которым понимается внутреннее состояние процесса: в число переменных внутреннего состояния процесса всегда входит pid мастер-узла, хранящийся в переменной типа список. Другой вариант представляет собой реализацию с помощью таблицы, однако гарантированно она будет содержать лишь одно значение, что нецелесообразно. Во внутреннее состояние коммуникатора так же входят pid поддерживающих мастер-узлов, они хранятся в том же списке, что и pid мастер-узла, но всегда после него.

Коммуникатор является ответственным за обновление информации о размещении вершин: при переразмещении вершины он отправляет соответствующую информацию мастер-узлу и всем поддерживающим мастер-узлам.

Вершина, представленная в виде процесса виртуальной машины, не обладает никакими таблицами, т.к. в модели графа с десятками тысяч вершин реализация потребляла бы слишком много памяти, вся необходимая информация содержится в состоянии процесса вершины, описываемом параметрами ID, Pid of comm, List of neighbors, List of connections.

Под параметром ID понимается постоянный уникальный идентификатор вершины (её номер). Pid of comm – pid коммуникатора узла, которому принадлежит вершина. Под списком соседей List of neighbors понимается список вершин, представленных кортежами вида {ID, pid}, которые указывают на эту вершину. Список соединений List of connections представляется списком всех вершин, на которые указывает вершина, запись представляется кортежем вида {ID, pid}.

Возможный вариант единичной записи для списков List of neighbors/List of connections: {ID, pid}. Достоинством такой записи является что:

1. Многие операции выполнимы без участия в них коммуникатора;
2. Упрощена операция переразмещения вершин в процессе балансировки нагрузки за счёт децентрализации хранения данных: в случае появления новых связей или удаления старых не требуется обращаться к глобальной базе данных, тем самым блокируя её.

Однако при этом необходимо обновление состояния при каждом изменении pid вершины из любого списка, что также следует учитывать.

В качестве механизма межпроцессного взаимодействия используем асинхронные сообщения. Это значит, что каждый процесс обладает «почтовым ящиком» (mailbox), из которого он может выбирать сообщения в порядке поступления или произвольно, по тегам, например, сообщения от определённого отправителя. Следует заметить, что при этом операции получения сообщения и его чтения могут быть разделены во времени.

Базовыми операциями в реализациях существующих моделей, как правило, являются синхронные операции чтения/записи, в предлагаемой – асинхронная передача сообщения, где сообщение представляет собой запрос или ответ.

Сравним предлагаемую асинхронную модель с синхронной. При этом предположим, что синхронная модель содержит аналог коммуникаторов (которые далее называются просто коммуникаторами), а каждый шаг алгоритма занимает 1 условную единицу времени – условный такт, за который выполняется одна элементарная базовая операция в моделях. Данная временная единица в описываемых моделях полностью совпадает для обращений к локальной памяти, условно совпадает при доступе к распределённой памяти для синхронной модели и отправки или чтении сообщений для предлагаемой модели. Условия совпадения: нет очереди на доступ к распределённой памяти (синхронная модель), нет очереди сообщений (предлагаемая модель).

Пусть каждая вершина, помимо своего номера, обладает некоторым весом. Будем рассматривать операции, которые встречаются в большинстве алгоритмов, решающих задачи над графами – чтение веса, запись веса, добавление вершины.

Операция получения веса вершины при традиционном представлении графов включает:

- обращение к управляющей программе;
- обращение к коммуникатору (синхронное);
- обращение к вершине (синхронное);
- возврат значения коммуникатору;
- возврат значения управляющей программе.

Временные диаграммы выполнения этой операции представлены на рисунке 3. (Здесь и далее предполагается, что состояние 0 означает «процесс/ресурс свободен», 1 – «процесс/ресурс занят»). Операции передачи и приёма на диаграммах выглядят как синхронные, но не являются таковыми: используется асинхронная передача сообщений («почтовый ящик»), в данном сравнении предполагается, что нет никаких других процессов, которые могут вызвать один или несколько тактов ожидания между отправкой сообщения и его чтением.

На диаграмме можно выделить следующие основные события:

- Синхронное обращение управляющей программы к коммуникатору (такт 2) – после успешного выполнения данной операции управляющая программа остаётся в заблокированном состоянии до получения ответа от коммуникатора (такт 5);
- Синхронное обращение коммуникатора к вершине (такты 3 и 4) – коммуникатор блокируется при обращении к вершине, таким образом, в случае

недоступности вершины, управляющая программа будет заблокирована на время блокировки вершины плюс 2 такта, необходимые коммуникатору на взаимодействие с управляющей программой. Таким образом, управляющая программа может быть заблокирована с такта 2 на неопределённое время.

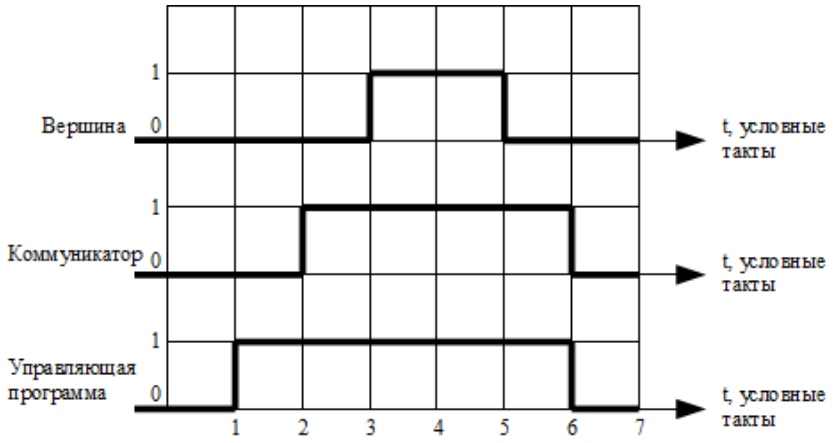


Рисунок 3. Временные диаграммы процесса чтения веса вершины для синхронной модели

В отличие от этого получение веса вершины при предлагаемом представлении включает:

- обращение к мастер-узлу (асинхронное);
- обращение к коммуникатору (асинхронное);
- обращение к вершине (асинхронное);
- возврат значения мастер-узлу.

Соответствующие временные диаграммы представлены на рисунке 4.

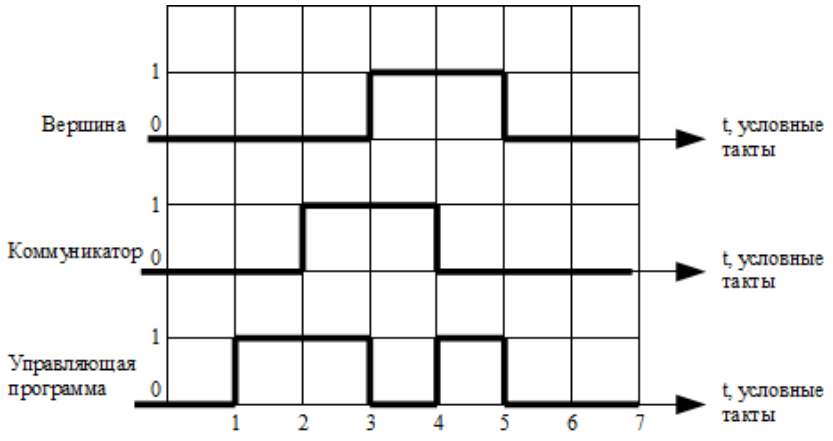


Рисунок 4. Временные диаграммы процесса чтения значения вершины для предлагаемой модели представления графа

Диаграмма включает в себя ряд отличий от предыдущей:

- Управляющая программа гарантированно находится в занятом состоянии 3 условных такта вместо минимум 5 для синхронной модели, что обеспечивает более высокий коэффициент готовности системы;
- Коммуникатор и вершина гарантированно находятся в заблокированном состоянии 2 условных такта, что также обеспечивает более удобный доступ к ресурсам в параллельной системе.

Таким образом, при выполнении операции чтения гарантируется её более быстрое выполнение и более высокий коэффициент готовности системы.

Выполнение операции записи веса вершины при использовании существующего представления включает:

- обращение к управляющей программе;
- обращение к коммуникатору (синхронное);
- обращение к вершине (синхронное) и запись значения.

Временные диаграммы данного алгоритма представлены на рисунке 5. Предполагается, что информация о завершении операции передаётся моментально.

Из диаграммы видно, что на такте 2 происходит синхронное обращение управляющей программы к коммуникатору, после чего коммуникатором выполняется операция синхронного записи веса вершины. Аналогично случаю чтения значения вершины, управляющая программа блокируется на такте 2 и ожидает ответа от коммуникатора, в данном случае время блокировки управляющей программы представляется как время ожидания получения доступа к вершине плюс 1 условный такт записи данных в коммуникатор. Предполагается, что после записи коммуникатор не производит никаких операций, кроме передачи сигнала разблокировки управляющей программе.

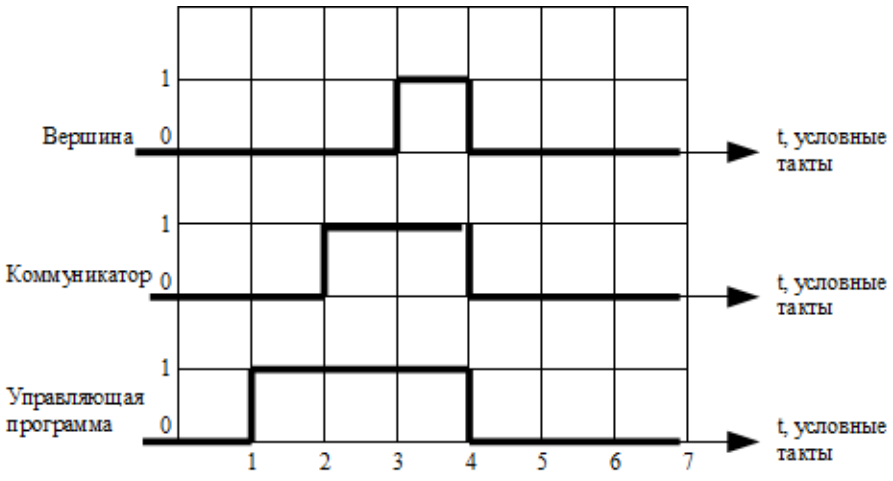


Рисунок 5. Временные диаграммы процесса записи значения вершины для существующей модели

Выполнение этой же операции записи веса вершины в предлагаемой модели включает:

- обращение к мастер-узлу (асинхронное);
- обращение к коммуникатору (асинхронное);
- обращение к вершине (асинхронное) и запись значения.

Временные диаграммы выполнения операции представлены на рисунке 6.

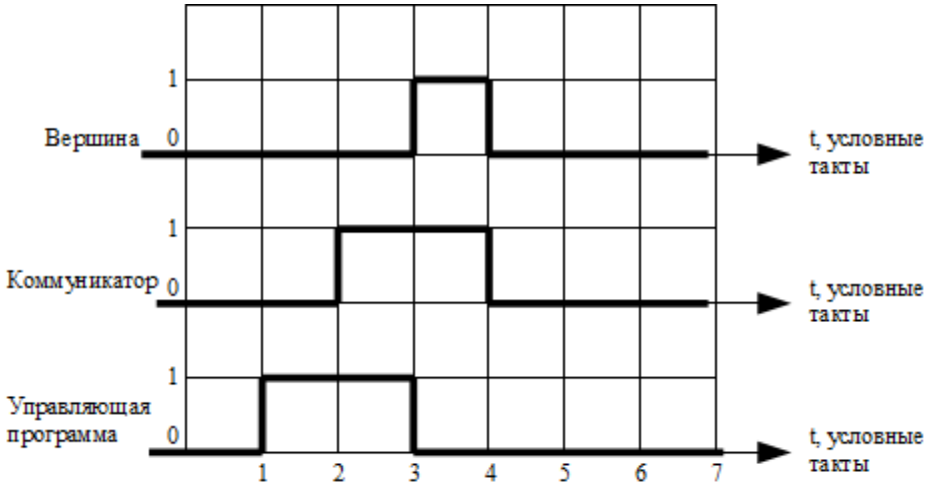


Рисунок 6. Временные диаграммы процесса записи веса вершины для предлагаемой модели

В отличие от синхронной модели, управляющая программа блокируется гарантированно только на 2 условных такта: такт обращения к самой управляющей программе и такт отправки сообщения коммуникатору. Хотя в идеальных условиях операция при обоих способах представления графа и занимает по 4 такта, но при реализации графа посредством предлагаемой модели выше коэффициент готовности системы.

И наконец, выполнение операции добавление вершины при существующем представлении графов включает:

- обращение к управляющей программе;
- поиск узла для вершины (осуществляется управляющей программой);
- обращение к коммуникатору выбранного узла (синхронное);
- создание вершины и запись значения.

Соответствующие временные диаграммы данного алгоритма представлены на рисунке 7.

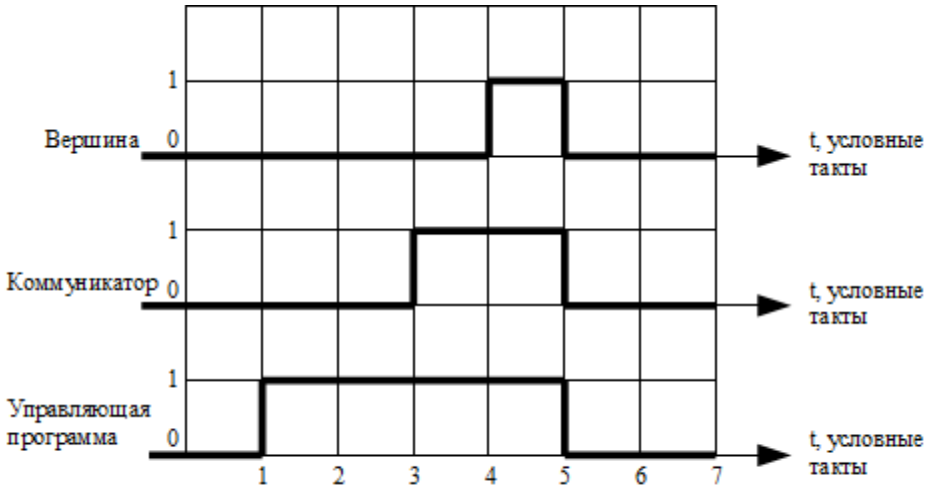


Рисунок 7. Временные диаграммы процесса добавления вершины для существующей модели

Для рассматриваемой операции в синхронной модели время блокировки управляющей программы зависит от реализации вершин, однако составляет не менее 4 тактов.

Выполнение операции добавления вершины в предлагаемой модели предполагает:

- обращение к мастер-узлу;
- поиск узла (с помощью поискового агента);
- обращение к коммуникатору выбранного узла;
- создание вершины, запись значения.

Временные диаграммы данного алгоритма представлены на рисунке 8. При этом следует отметить, что процесс «вершина» не существует до такта 4, потому, хотя его состояние до этого момента и показано как «свободен», обращение к нему невозможно. Также невозможно обращение к поисковому агенту после такта 4, на котором он обращается к коммуникатору выбранного узла, после чего завершает свою работу. Поисковых агентов в системе может быть более одного.

Как видно из диаграммы, время блокировки управляющей программы гарантированно составляет 2 условных такта. Время от запроса до размещения вершины варьируется в зависимости от графа, однако при этом управляющая программа и все её коммуникаторы остаются свободными и готовы для обработки новых запросов. Наличие специализированных поисковых агентов позволяет добиться более полного использования потенциала параллельных систем и более равномерного распределения нагрузки: при добавлении сразу множества вершин каждый поисковый агент может искать наиболее удачное размещение для одной или нескольких вершин. Так как поисковые агенты являются независимыми процессами, то потенциально они могут быть размещены вне управляющего узла, что позволит использовать ресурсы управляемых (вычислительных)

узлов. В случае традиционной модели такое невозможно: масштабирование алгоритма ограничено возможностями управляющей программы, т.е. теми потоками (программными или аппаратными), которыми она может управлять.

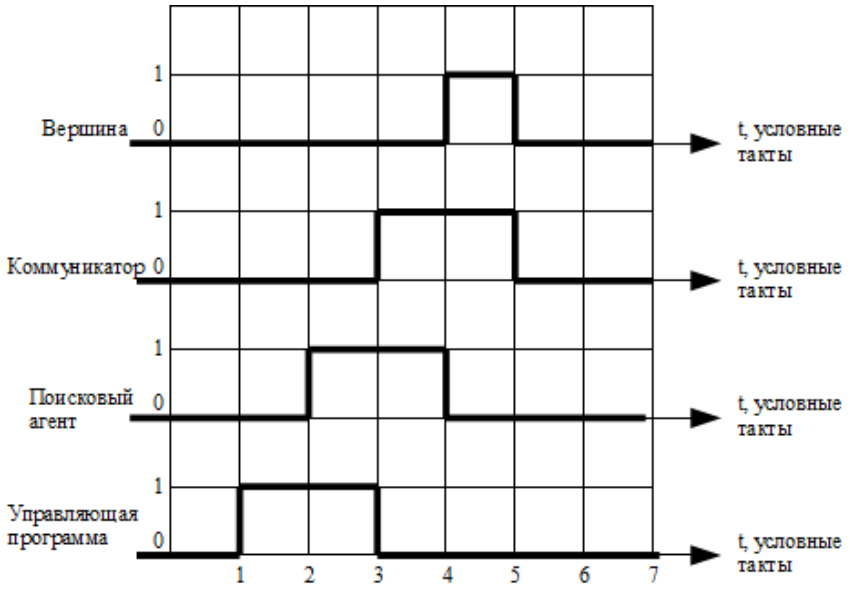


Рисунок 8. Временные диаграммы процесса добавления вершины для предлагаемой модели

Таким образом, анализ временных диаграмм показывает, что предлагаемая модель позволяет более эффективно использовать ресурсы многопроцессорной вычислительной системы за счёт делегирования обязанностей специализированным процессам и асинхронных сообщений как средства межпроцессного взаимодействия. При этом управляющая программа чаще находится в состоянии готовности, в то время как в существующих моделях для начала обработки нового запроса требуется сначала завершить обработку текущего, что может заметно снизить эффективность использования ресурсов. Подразумевается возможность реализации синхронных операций на базе асинхронных благодаря возможности ожидания сообщения – управляющая программа ожидает сообщение с определённым тегом, при этом другие сообщения временно игнорируются, но сохраняются в очереди.

При работе с графом, представленным с помощью предлагаемой асинхронной модели, могут проявиться следующие особенности:

1. Большее потребление памяти за счёт использования дополнительных структур данных реализации, поскольку потребуется дополнительная память для хранения таких данных, как pid вершин, размещение вершин по узлам, также оперативная память для поддержания работы управляющих процессов глобального и локального уровней;

2. Представление вершин активными процессами, по сути, подразумевает предварительное выполнение операций (представление всех вершин активными вместо выбора одной вершины или набора вершин для активизации). Это означает, что в некоторых ситуациях реализация данной модели может проиграть предлагаемой ранее. Например, построение большого графа и его удаление без выполнения каких-либо других операций.

Выводы

Выполненный анализ позволяет сделать вывод о том, что предлагаемая асинхронная модель представления графа в обрабатывающей программе обеспечивает лучшие возможности распараллеливания операций, выполняемых над графами при решении задач структурного анализа и синтеза, что позволяет уменьшить время решения таких задач на ЭВМ. Причем, помимо обозначенных в статье операций, также возможно заметное ускорение балансировки нагрузки в условиях динамически изменяющейся структуры графа, а также решение ряда проблем, описанных в [3], в частности: применение модели для гетерогенных систем, метакомпьютинг.

С учетом того, что многие задачи структурного анализа и синтеза являются NP-полными, представление графов подобными асинхронными моделями представляется перспективным.

Литература

1. Гергель В.П. Теория и практика параллельных вычислений. М.: Интернет-Университет Информационных Технологий; БИНОМ. Лаборатория знаний, 2013. – 423 с.

2. М.В. Якобовский. Введение в параллельные методы решения задач. М.: Изд-во Московского университета, 2013. – 328 с.

3. Shloegel K., Karypis G., Kumar V. Graph Partitioning for High Performance Scientific Simulations. University of Minnesota, 2000.

4. Чезарини Ф., Томпсон С. Программирование в Erlang. М.: изд-во «ДМК», 2012. – 488 с.

УДК 002.53

МЕДИЦИНСКИЕ ИНФОРМАЦИОННЫЕ СИСТЕМЫ В ЛЕЧЕБНЫХ УЧРЕЖДЕНИЯХ РОССИИ: ПРОБЛЕМЫ СРАВНЕНИЯ И ОЦЕНКИ

А.В. Пронин, Е.В. Смирнова

Ключевые слова: медицинская информационная система, лечебно-профилактическое учреждение, оценка, критерии сравнения МИС.

Key words: Medical Information System, health care institution, evaluation, comparison criteria MIS.

Аннотация: В статье рассматривается уровень обеспеченности лечебных учреждений медицинскими информационными системами и их технологическая развитость. Описываются особенности внедрения медицинских информационных систем в специализированных лечебно-профилактических учреждениях. Предложены обобщенные критерии сравнения информационных систем.

Введение

В настоящее время информатизация здравоохранения является важнейшим направлением модернизации, повышения эффективности управления и качества медицинского обслуживания. Развитию этой области уделяется самое пристальное внимание – причем не только со стороны практического звена и научного общества, но и со стороны государства.

Как показывают данные статистики в работе [1], основными заказчиками МИС являются государственные лечебно-профилактическими учреждениями (ЛПУ): 70% всех используемых информационных систем приходятся именно на эти учреждения. На текущем этапе все внедрения МИС производятся самыми разными фирмами от небольших групп разработчиков, использующих открытые решения, до крупных компаний, таких как: 1С, IBS, Интерин и других, их число достигает 300. Существующие решения, а на сегодня насчитывается более 600 продуктов, сильно разнятся между собой – начиная с того, что продукты для информатизации здравоохранения устанавливаются на разные операционные системы (Windows, Unix, Mac), имеют разные ограничения по количеству автоматизированных рабочих мест, по масштабируемости и заканчивая возможностями интеграции с другими системами, такими как аптечные системы, системы кадрового учета, бухгалтерские системы.

Многообразие решений, децентрализованность данных и несовершенство решений подвигли Правительство России начать процесс создания Единой государственной

информационной медицинской системы, одно из первых упоминаний о которой прозвучало в апреле 2010 г. от Председателя Правительства РФ В.В. Путина в его выступлении в Государственной Думе с отчетом о деятельности Правительства [2]. Единая государственная информационная медицинская система должна централизовать и упорядочить обработку медицинских данных в масштабах страны, что позволит решить множество проблем, существующих сегодня в процессе медицинского обслуживания. Свидетельством ускоренного движения в этом направлении является также документ от 11.11.2013 г. «Основные разделы электронной медицинской карты» за подписью министра здравоохранения РФ, размещенный на сайте Минздрава РФ [3].

Но даже при масштабной централизации данных, эта единая государственная информационная медицинская система не сможет удовлетворить специфические потребности каждого ЛПУ. Будет организован сбор данных о пациентах, для возможности доступа к ним из использующих эту систему ЛПУ, а все частные задачи и административные проблемы конкретной клиники будут по-прежнему решаться с помощью коммерческих продуктов. Можно провести аналогию с бухгалтерским учетом на предприятиях и в банках, где отчетность сдается в соответствующий государственный орган в едином унифицированном формате, а ведение бухгалтерии происходит в коммерческих продуктах.

Применимость и значение медицинской информационной системы каждый понимает по-своему. Для небольших клиник может оказаться достаточным ведение журнала пациентов, диагнозов и оказываемых услуг. Крупные ЛПУ нуждаются в полной автоматизации и централизованной обработке информации во всех сферах деятельности учреждения, к которым относятся:

−история болезни пациента (группа записей в формализованной форме обо всех событиях, происходящих с пациентом);

−учет аптечных средств (поставка, складирование, распределение, утилизация лекарственных форм в ЛПУ);

−расписание (полная информация о режиме работы для врача и для пациента);

−запись на прием и выписка (интерфейс пользователя, посредством которого пациент самостоятельно или сотрудник ЛПУ записываются на визит к специалисту);

−статистика (медицинские статистические срезы, отчетные формы, заключения, оценки медицинских показателей и другое).

Комплексность и полнота решаемых задач медицинской информационной системы может быть достигнута использованием современных технологий обработки и хранения данных. Постоянно меняющиеся условия работы и отчетности медицинских учреждений

накладывают на информационную систему требования гибкости и простоты модернизации.

Критерии сравнения МИС

Функциональность. Часть организаций автоматизируют лишь несколько компьютеров, кто-то старается автоматизировать весь процесс обслуживания пациентов. К сожалению, очень большая часть составляют малые внедрения с небольшим количеством рабочих мест. После экономического кризиса 2008 года эти показатели стали улучшаться и как следствие, совершенствуются технические решения. Можно разделить существующие решения по следующим уровням [4]:

−автоматизация отдельных рабочих мест;

−наличие сети и единой базы данных в ЛПУ;

−интегрированная МИС (совместно с существующими в ЛПУ информационными системами Аптека, Бухгалтерия, Кадры и т.д.);

−наличие статистической обработки данных, вывод статистических отчетов;

−появление элементов контроля (выдача информации пользователю МИС на основе текущей информации о пациенте и другое);

−прогнозирование (автоматизация закупок лекарств и другое).

Используемые системой сетевые технологии. Для создания гибкой и модернизируемой системы очевидным является переход от технологий «толстого» клиента к «тонкому». Решения, базирующиеся на технологиях толстого клиента проще и дешевле в разработке, имеют более высокий функционал на клиентской стороне и возможно работы без связи с сервером, вследствие чего более популярны. Но обладают существенными минусами [5]:

−большой размер дистрибутива;

−многое в работе клиента зависит от того, для какой платформы он разрабатывался;

−при работе с ним возникают проблемы с удаленным доступом к данным;

−довольно сложный процесс установки и настройки;

−сложность обновления и связанная с ней неактуальность данных;

−наличие бизнес-логики.

Решения, базирующиеся на тонких клиентах, проще в эксплуатации, требуют менее мощной аппаратной части, требуют меньших затрат при масштабируемости. Недостатками тонкого клиента являются: меньшая производительность вычислений на стороне клиента, теоретически большая уязвимость и более высокие требования к локальной сети ЛПУ. На сегодняшнем этапе развития информационных технологий

тонкие клиенты становятся все более популярны и применимы. Но сейчас 60% существующих систем МИС используют толстый клиент, 25% – смешанный подход и лишь 15 % – полностью реализованы на базе тонкого клиента [1].

Веб-технологии и облачные вычисления. Необходимо отметить, что, несмотря на призывы активнее использовать «облачные вычисления» (для которых работа в браузере, фактически, необходимое условие), большинство существующих МИС этот тренд не поддерживают. Ситуация при этом меняется, но крайне медленно. В основном процесс носит эволюционный характер – в крупных системах появляются отдельные модули, реализованные как «тонкий клиент», но массовой поддержки работы МИС в браузере на текущий момент нет.

Особенности МИС в специализированных ЛПУ

Существенные изменения происходят в распределении проектов внедрения МИС в зависимости от типа ЛПУ. Наибольшую область занимают медицинские центры и поликлиники. Количество внедрений в стационары, специальные ЛПУ (диспансеры, стоматологические поликлиники) и санатории уменьшилось в 2012 году почти в 2 раза. Хотя именно стационары и диспансеры оказывают наиболее полный спектр медицинских услуг и нуждаются в более совершенной, с технической точки зрения, автоматизированной системе.

Основное влияние на архитектуру и работу информационных систем для стационаров и специальных ЛПУ оказывает внутренний регламент учреждений и специфические особенности. Такие ЛПУ работают на плановой основе, и главным источником финансирования для них является государство, а средства выделяются на основе точности выполнения плана. Это означает, что система должна иметь в себе особые модули, показывающие текущее выполнение плана стационаром, статистические срезы по большому спектру параметров и элементы прогнозирования на основе полученной статистики. Наиболее сложной в реализации такого рода модулей является задача получения упорядоченной информации в масштабе одной ЛПУ для достижения достоверности данных. Данные должны быть формализованы и стандартизованы, что в дальнейшем позволит их обработать, отсортировать или использовать для проведения медицинских исследований. Весь путь лечения пациента внутри ЛПУ должен фиксироваться, начиная от поступления пациента и предоставления им своих персональных данных, до его выписки с выдачей листка нетрудоспособности. Имея полные данные обо всех пациентах, система на их основе сможет строить статистические выборки, например, статистику по группам диагнозов, установленных пациентам, потреблению лекарств или по выполнение плана тем или иным подразделением ЛПУ. Все

это может быть достигнуто только полной автоматизацией рабочих мест ЛПУ, что зачастую является колоссально сложным процессом для построения МИС с нуля, в силу необходимости больших вложений и обучения персонала.

Выводы

Разработка медицинских информационных систем является на сегодня востребованным и перспективным направлением. Для централизации данных обо всех жителях страны создается Единая Медицинская Система [4], но она не может удовлетворить всем потребностям отдельно взятого предприятия, вследствие чего на рынке предлагается большое количество разноуровневых МИС, отличающихся по своим техническим и функциональным характеристикам. Отсутствие ориентированности на тонкий клиент, возможности перехода на облачные вычисления и необходимость удовлетворения потребностей специализированных ЛПУ являются слабыми местами этого направления, что дает большие перспективы развития в данном направлении.

Литература

1. Гусев А.В. Медицинские информационные системы в России: текущее состояние, актуальные проблемы и тенденции развития [Электронный ресурс]. – URL: http://www.gosbook.ru/node/24926 (дата обращения: 20.12.2013).
2. Отчет о деятельности Правительства РФ в 2009 году [Электронный ресурс]. – URL: http://medportal.ru/mednovosti/news/2010/04/20/hosp/ (дата обращения: 20.12.2013).
3. Структура электронной медицинской карты сайт Минздрава России [Электронный ресурс]. – URL: http://www.rosminzdrav.ru/health/it/57 (дата обращения: 20.12.2013).
4. ГОСТ Р52636-2006. Электронная история болезни, общие положения, 2008 год [Электронный ресурс]. – URL: http://resortsoft.ru/publications/gost52636.html (дата обращения: 20.12.2013).
5. Википедия, Толстый клиент [Электронный ресурс]. – URL: http://en.wikipedia.org/wiki/Fat_client (дата обращения: 20.12.2013).

УДК 004.722.4

АКТУАЛЬНОСТЬ ВНЕДРЕНИЯ ТЕХНОЛОГИИ LTE-ADVANCED

Д.Э. Трибушков, А.А. Кочетков, О.Ю. Еремин

Ключевые слова: технология LTE-Advanced, агрегация частот, координированная передача/приём данных, ретрансляция.

Keywords: LTE-Advanced Technology, Carrier Aggregation, Coordinated Multiple Transmission/Reception, Relaying

Аннотация: В статье выполнен сравнительный анализ двух технологий беспроводной связи LTE и LTE-Advanced. Проанализированы качественные и принципиальные улучшения нового стандарта в сравнении с нынешним, описаны новые методы построения сетей и организации обмена данными, способствующие удовлетворению требованиям нового стандарта связи. В статье рассмотрены преимущества метода агрегации частот, основные изменения в беспроводных сетях для обеспечения многоантенной передачи данных, принцип взаимодействия терминалов с базовыми станциями при координированных передаче и приеме (CoMP), а также метод ретрансляции сигнала. Результаты анализа подтвердили целесообразность внедрения новой технологии беспроводной связи LTE-Advanced.

Появление новых услуг и совершенствование технических возможностей мобильных устройств связи привело к существенному росту объемов трафика в сотовых сетях и увеличению потребительского спроса на высокоскоростную передачу данных. Так, трафик данных в сетях мобильной широкополосной связи демонстрирует практически экспоненциальный рост. Уже в 2009 году объемы мобильного трафика данных превысили объем трафика голосовой связи [1]. То есть передача данных стала преобладающим видом трафика в мобильных сетях.

Система 3GPP Long Term Evolution (LTE) была разработана для того, чтобы обеспечить высокую скорость передачи данных, низкую задержку, увеличить площадь покрытия. LTE-Advanced обеспечивает обратную совместимость и может функционировать в спектре, в котором работает LTE, без влияния на существующие устройства.

Значения пиковой (максимальной) скорости передачи данных в настоящее время составляют 100 Мбит/с в нисходящем канале и 50 Мбит/с в восходящем канале. Общая задержка передачи сигнального трафика, соответствующая времени передачи на участке радиоинтерфейса и опорной сети в условиях малой нагрузки, равна 50 мс и 5 мс для

сигнальных и пользовательских данных соответственно. Время, необходимое на переключение мобильной станции из холостого состояния в активное, составляет около 100 мс [2].

Параметры LTE уже не удовлетворяют запросам быстро растущей аудитории пользователей мобильных сетей, поэтому становится актуальным вопрос о внедрении нового стандарта беспроводной связи.

Новая система LTE-Advanced способна поддержать до 300 активных пользователей при ширине канала в 5 МГц. Целевые значения для пиковой спектральной эффективности при нисходящей передаче – 30 бит/с/Гц, а при восходящей передаче – 14 бит/с/ Гц.

Система обеспечивает более высокое значение средней спектральной эффективности при разумной сложности самой системы. Максимальное целевое значение для средней спектральной эффективности для нисходящего канала равно 3,7 бит/с/Гц/сектор (при конфигурации 4x4, т.е. 4 передающие и 4 приемные антенны), а для восходящего канала – 2,0 бит/с/Гц/сектор (при конфигурации 2x4).

LTE-A поддерживает работу с мобильными пользователями, которые могут двигаться со скоростью до 350 км/ч (или даже до 500 км/ч, в зависимости от используемых частот). Производительность системы улучшена при работе с пользователями, которые перемещаются со скоростью от 0 до 10 км/ч.

LTE-Advanced поддерживает работу с различными размерами частотных диапазонов, в том числе и с более широкими диапазонами (например до 100 МГц), чем указанные в предыдущей версии LTE, для того, чтобы обеспечить более высокую производительность и целевую пиковую спектральную эффективность. Для достижения этих результатов, пришлось совершить несколько улучшений на физическом уровне, такие как: агрегация каналов, координирование множественных передач, мультиплексирование восходящих и нисходящих линий связи до 4 и 8 антенн соответственно.

Несколько спектров могут быть объединены при сохранении обратной совместимости (с устаревшими пользовательскими устройствами). На рисунке 1 показано, как пять спектров на частоте 20 МГц могут быть объединены, чтобы сформировать один на частоте 100 МГц. После внедрения LTE-A, будет задействован новый спектр частот, при этом сохранится возможность использования старых полос. Например, можно развернуть систему на частоте 40 МГц с помощью 2 x 20 МГц UMTS носителей в диапазоне 2.6 ГГц.

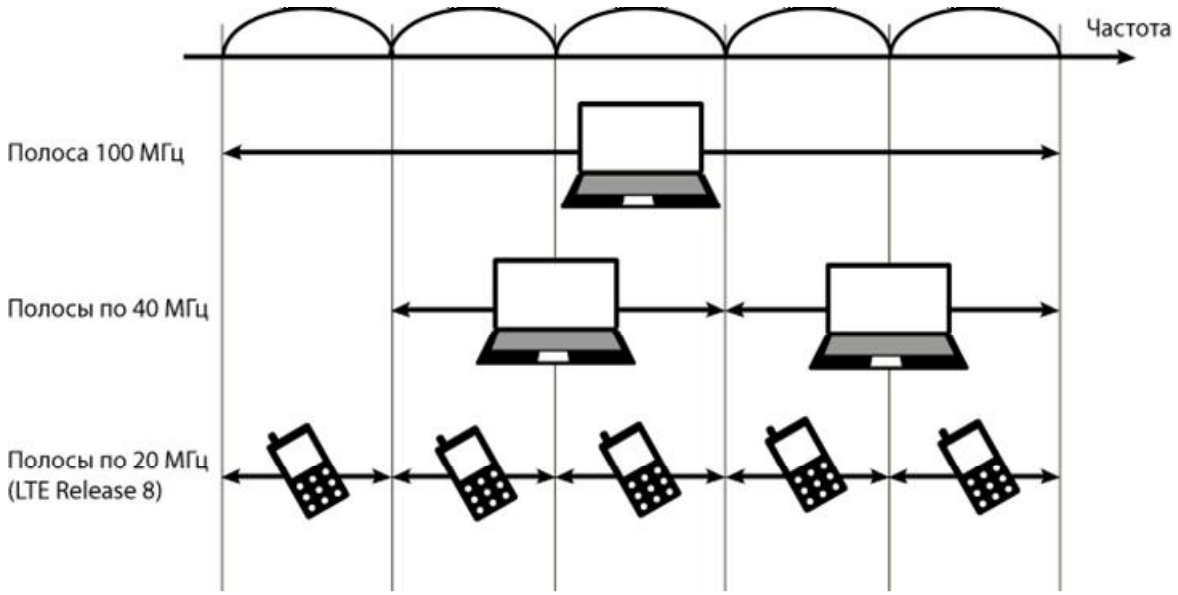


Рисунок 1. Агрегация частот

Многоантенные решения играют важную роль в увеличении спектральной эффективности. В LTE-A возможности многоантенной передачи по нисходящему каналу расширены за счет поддержки пространственного мультиплексирования до восьми передающих антенн и, соответственно, восьми передающих трактов. В сочетании с расширением полосы частот до 100 МГц за счет агрегации частот это позволяет достичь пиковых скоростей передачи данных порядка 3 Гбит/с, или 30 бит/с/Гц.

В LTE-A также поддерживается многоантенная передача по восходящему каналу с мультиплексированием до четырех передающих антенн и, соответственно, четырех передающих трактов. При этом достигается пиковая скорость передачи данных по восходящему каналу порядка 1,5 Гбит/с в полосе 100 МГц или 15 бит/с/Гц.

Специальный шаблон сигнала демодуляции, установленный в системах LTE, должен быть изменен с целью поддержки до 8 антенн в LTE-Advanced системах. Также возникает необходимость изменить информацию о состоянии канала передачи сигналов (CSI-PTC) и обратной связи в архитектуре таблицы маршрутизации. Должны быть проведены эквивалентные изменения системы управления нисходящей линии связи, чтобы удовлетворить требованиям стандарта LTE-Advanced [3].

Одним из новых методов, который используется в стандарте LTE-A, является координированная передача и прием (Coordinated Multipoint, CoMP). Иными словами, реализовано обслуживание одного абонентского устройства несколькими базовыми станциями, принцип работы которого отображен на рисунке 2. Координированная передача и прием рассматриваются как способ, с помощью которого можно увеличить пропускную способность на границах секторов. При этом повышение пропускной

способности в нисходящем канале (DL) достигается за счет уменьшения уровня интерференции (Inter Cell Interference). А при восходящей передаче (UL) – за счет обработки принятого сигнала на нескольких базовых станциях.

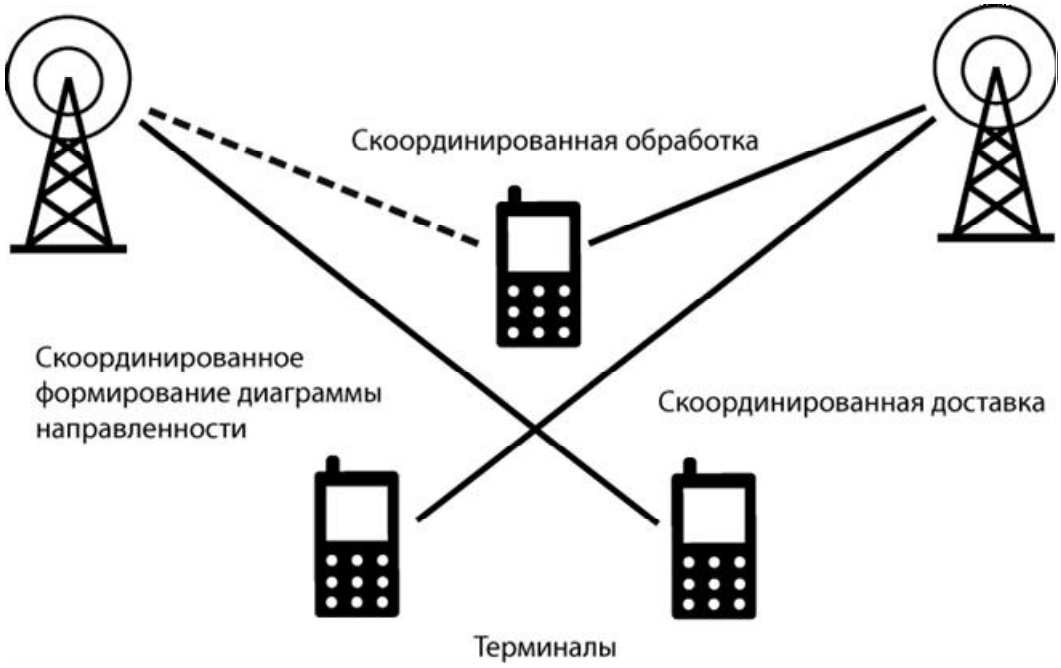


Рисунок 2. Координированная передача и прием

Рассмотрим возможные варианты организации координированной передачи в нисходящем канале. Среди основных вариантов можно выделить следующие: совместная передача (joint processing, JP) и координированное планирование (coordinated scheduling, CS). В случае совместной передачи передаваемые данные доступны на всех базовых станциях (БС), с которых ведется передача. Однако, существует два различных варианта реализации этого подхода. В первом варианте осуществляется одновременная передача с нескольких БС. А во втором варианте БС, которая осуществляет передачу данных, выбирается динамически. То есть передача осуществляется только с одной БС в каждый момент времени (при этом данные для передачи доступны на нескольких БС). В случае координированного планирования передач данные всегда передаются только с одной БС, при этом решение о расписании передач делается с учетом информации от нескольких БС. Ниже описанные варианты организации координированной передачи в нисходящем канале представлены в таблице.

Координированная передача и планирование Таблица

Координированная передача		Координированное планирование
Совместная	Динамический выбор	

передача	сектора	
Данные доступны на всех БС, которые осуществляют передачу	Данные доступны на всех БС, которые могут осуществлять передачу	Данные доступны только на той БС, которая осуществляет передачу
Множество БС одновременно передают одни и те же данные	Только выбранная БС передает данные	Только выбранная БС передает данные. Однако во время планирования передач учитывается информация от множества БС

При координированном приеме данных (т.е. при восходящей передаче) можно выделить так же два различных варианта ее организации. Первый вариант – это прием сигнала от МС на нескольких БС (совместный прием, joint reception, JR). И второй вариант – это, так же как и при нисходящей передаче, координированное планирование передач с целью уменьшения или полного исчезновения интерференции. Кроме этого, возможна комбинация обоих названных вариантов.

В зависимости от архитектуры системы и пропускной способности линий передачи данных, которые имеются у оператора связи, существуют различные варианты реализации координированного приема данных. Например, конечные узлы (БС или радио модули) могут отправлять принятые данные в некий центральный узел, который будет производить их обработку. В этом случае достижим максимально-возможный выигрыш от использования координированного приема. Однако, для его реализации нужны высокоскоростные линии передачи данных. Другой вариант – оставить часть обработки принятого сигнала на стороне конечного узла и только после этой обработки отправлять принятые данные в центральный узел для дальнейшей их обработки. Чем больше функций обработки остается в конечном узле, тем меньшее количество данных необходимо отправлять в центральный узел и, соответственно, тем менее скоростные линии требуются. Однако, чем больше обработки принятого сигнала осуществляется в конечных узлах, тем выигрыш от использования координированного приема становится меньше [4].

Для функционирования координированного приема необходимо, чтобы все БС, осуществляющие прием данных, были синхронизированы по частоте и времени. А кроме этого должны соблюдаться требования к задержке передачи данных между БС (в случае распределенного решения) или центральным узлом и конечными узлами (в случае централизованного решения).

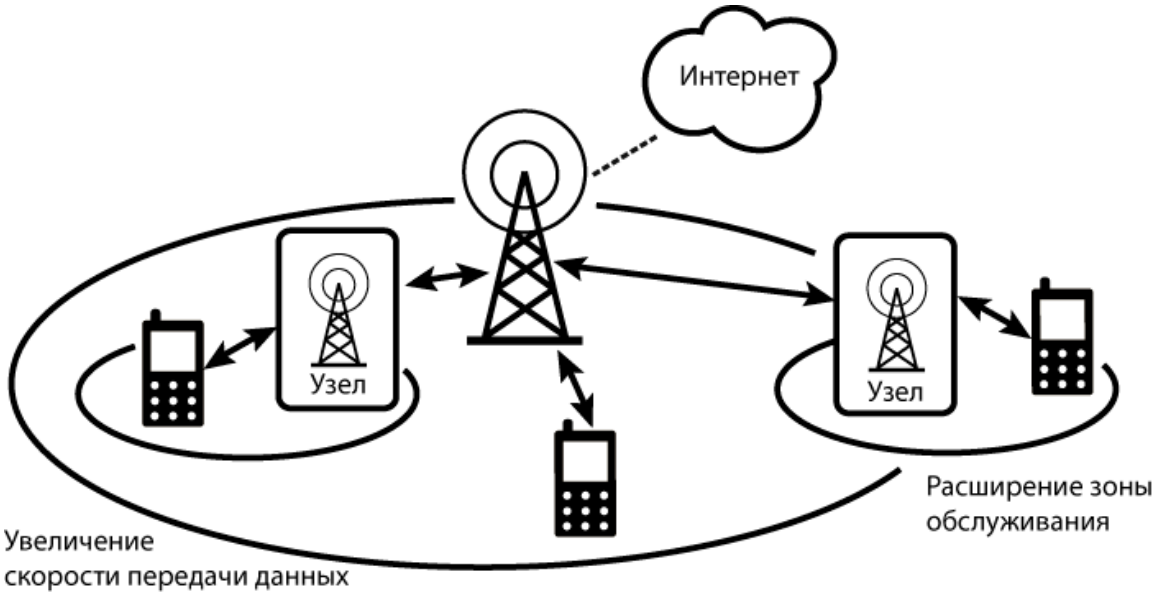


Рисунок 3. Узлы ретрансляции

В LTE-A поддерживается функция ретрансляции, что позволяет мобильным терминалам обмениваться данными с сетью через узел ретрансляции, соединенный по беспроводной связи с основным узлом, с использованием технологии радиодоступа и спектра LTE, как показано на рисунке 3. С точки зрения терминала узел ретрансляции представляется «обычной» базовой станцией. Это означает, что устаревшие пользовательские устройства также могут подключаться к сети через узел ретрансляции. Функция ретрансляции может стать одним из способов быстрого и экономически эффективного расширения покрытия сети LTE. Сюда входят как расширение зоны обслуживания, так и увеличение скорости передачи данных [5].

Выводы

Итак, наиболее перспективной технологией беспроводной связи, которая сможет заменить нынешние, является LTE-Advanced. При относительной простоте построения новых сетей, скорость передачи сигнальных данных в них возрастет в несколько раз, существенно расширится зона покрытия и увеличится производительность сетей в целом. В силу своих преимуществ технология LTE-Advanced, возможно, станет самым распространенным видом мобильных сетей в ближайшем будущем.

Литература

1. LTE-Advanced: Next generation wireless broadband technology / A.Ghosh [и др.]. // IEEE Wireless Comm., 2010.

2. LTE-Advanced – Evolving LTE towards IMT-Advanced / S.Parkvall [и др.]. // VTC 2008-Fall, Calgary, Canada, 2008.

3. Effect of Relaying on Capacity Improvement in WLAN / A. So [и др.]. // WCNC, New Orleans, Louisiana, USA, 2005.

4. Coordinated Multipoint Transmission/Reception Techniques For LTE-A / M. Sawahashi [и др.]. // IEEE Wireless Comm., 2010.

5. IEEE 802.16j Relay-Based Wireless Access Networks: An Overview / V. Genc, S. Murphy, Y. Yu, J. Murphy [и др.]. // IEEE Wireless Comm., 2008.

УДК 004.4'23

АВТОМАТИЗАЦИЯ НЕПРЕРЫВНОГО РАЗВЕРТЫВАНИЯ ПРИЛОЖЕНИЙ

Г. П. Гаджиев, Е. А. Авраменко, Т.Н. Ничушкина

Ключевые слова: *веб-сервис, непрерывная интеграция, эксплуатация ПО.*

Key words: *web-service, continuous integration, deployment.*

Аннотация: *В статье рассмотрен цикл разработки программного продукта, предполагающий автоматизацию тестирования и частые выпуски обновлений. Описано создание программного обеспечения, осуществляющего как автоматическое, так и ручное управление развертыванием версий приложений на серверах.*

Распространение веб-приложений позволило сократить время доставки новых версий ПО конечным пользователям до пары минут. Вследствие этого, классический

подход к разработке, предполагающий периодические крупные выпуски, сильно уступил свои позиции в не критичных к сбоям, но требовательным к скорости развития областях.

Команда *GitHub* (крупнейший веб-сервис для совместной разработки) запускает сотни новых версий в неделю, что позволяет им оперативно добавлять новый функционал, проверять его на реальных пользователях и откатывать изменения в случае регресса.

Подобный подход к разработке невозможно представить без покрытия кодовой базы тестами, которые будут сигнализировать о возникновении ошибок в различных частях приложения, а также отслеживать падения производительности, вызванные последними изменениями.

Небольшие тестовые наборы можно запускать локально на компьютере разработчика после каждого изменения, однако с ростом количества проверок, время прохождения тестов может достигать нескольких часов, в связи с чем, для постоянного тестирования часто используются так называемые серверы непрерывной интеграции (*continuous integration*). Используемый в предлагаемой работе сервис *Travis-CI.org,* бесплатен для проектов с открытым исходным кодом и может быть гибко настроен для проверки приложения с различными версиями интерпретаторов и компиляторов.

Цель работы

Основной задачей предлагаемой работы являлось создание ПО, позволяющего автоматически разворачивать новые версии приложения после прохождения ими тестового набора, а также при необходимости управлять этим процессом вручную. В качестве тестируемого проекта использовался веб-сервис с активной пользовательской базой, разрабатываемый одним из авторов данной работы.

Использованные технологии

Технически, разработанное ПО, состоит из двух частей: сервиса, отвечающего за обработку результатов тестирования и интерфейса пользователя.

Серверная часть

Сервис разработан на языке программирования *JavaScript* с использованием платформы *Node.js*, идеально подходящей для реализации не критичной к отказам системы мягкого реального времени. В предлагаемой реализации задачей серверной части является обработка входящего запроса от сервера *Travis-CI* и, в случае положительного результата прохождения тестов, запуск новой версии.

Сам процесс развертывания обновлений реализован с помощью *Capistrano,* набора скриптов, упрощающих задачи обновления кодовой базы, подготовки связанных с ПО медиа-файлов, проверки зависимостей, а также сохранения нескольких последних версий приложения на серверах для быстрого возврата на них в случае возникновения неполадок.

Для связи серверной части с клиентской используется база данных *Firebase*, работающая в режиме мягкого реального времени. Изменение любой записи влечет за собой автоматическое обновление данных сразу для всех подключенных к базе в данный момент сторон.

Клиентская часть

Пользовательский интерфейс разработан с использованием фреймворка *AngularJS*, отличительной чертой которого является двухстороннее связывание данных. Данная функциональность особенно полезна в предлагаемом контексте использования, так как в связке с *Firebase* она позволяет реализовать интуитивно понятную связь между клиентской и серверной частью: правка статуса одной из версий через интерфейс сразу же отразится на работе сервиса, при этом с разработчика снимается задача по синхронизации изменений данных.

Результаты внедрения и перспективы развития

В результате внедрения разработанного сервиса, процесс создания проекта был сведен к следующим пунктам:

1. Изменения в коде сохраняются в системе управления версиями (*Git*);
2. Новая версия отправляется в репозиторий (*GitHub*);
3. *Travis-CI* получает оповещение об изменении кодовой базы и запускает тестовый набор на последней версии, получаемой из репозитория;
4. Результат прогона тестов отправляется на разработанный авторами сервис, который в случае отсутствия ошибок запускает процесс развертывания новой версии на серверах.

Текущее состояние процесса разработчики можно отслеживать через веб-интерфейс, позволяющий, при необходимости, вручную разворачивать версии даже при наличии ошибок, а также откатывать кодовую базу на предыдущее состояние в случае возникновения неполадок, не выявленных набором тестов.

Сокращение сроков применения изменений до нескольких минут позволило значительно ускорить получение обратной связи от пользователей, что, в конечном счете, положительно повлияло как на качество ПО, так и на скорость разработки новых модулей.

В дальнейшие планы по развитию ПО входит устранение зависимостей от конкретных сервисов репозиториев и серверов интеграции, а также добавление различных видов уведомлений о состояниях версий.

Литература

1. Jake Douglas. “Deploying at GitHub” [Электронный ресурс]. – URL: http://github.com/blog/1241 (дата обращения: 20.03.2014).

2. Scott Chacon. “GitHub Flow” [Электронный ресурс]. – URL: http://scottchacon.com/2011/08/31/github-flow.html (дата обращения: 20.03.2014).

3. Jason Rudolph. “API Design at GitHub” Yet another Conference 2013 [Электронный ресурс]. – URL: http://tech.yandex.ru/events/yac/2013/talks/1100/ (дата обращения: 20.03.2014).

4. “Travis CI Documentation” [Электронный ресурс]. – URL: http://about.travis-ci.org/ (дата обращения: 20.03.2014).

5. “Firebase Documentation” [Электронный ресурс]. – URL: https://www.firebase.com/docs/ (дата обращения: 20.03.2014).

УДК 004.4'23

РАЗРАБОТКА ПО С ИСПОЛЬЗОВАНИЕМ РАСПРЕДЕЛЕННЫХ СИСТЕМ УПРАВЛЕНИЯ ВЕРСИЯМИ

Г.П. Гаджиев, Т.Н. Ничушкина

Ключевые слова: *Совместная разработка, распределенные системы управления версиями.*

Key words: *Collaborative Development, Distributed Version Control System.*

Аннотация: *В статье выполнен сравнительный анализ централизованных и распределенных систем управления версиями (Distributed Version Control System, DVCS), рассмотрено внутреннее устройство DVCS Git, а также описаны основные принципы совместной разработки с использованием подобных систем.*

Перед разработчиками программного обеспечения рано или поздно встает проблема сохранения различных версий исходных кодов и обмена ими с другими членами

команды. Зачастую используемый классический подход состоит в создании множества директорий и сохранения копий исходных кодов в них. Такие директории обычно именуют по дате, версии, либо добавляемому функционалу.

Подобный подход, не смотря на свою простоту, имеет множество недостатков:

- повторение одних и тех же действий;
- ручное резервное копирование;
- большой объем хранилища;
- неудобная совместная работа;
- ручной поиск изменений.

Для автоматизации контроля истории изменений исходных кодов были разработаны системы управления версиями (*Version Control System, VCS*). Традиционные *VCS* (такие как *CVS* и *Subversion*) представляют собой клиент-серверную систему, в которой сервер является хранилищем и выполняет большую часть функций по управлению версиями. Централизованность таких систем является причиной ряда проблем:

- единая точка сбоя (сервер-хранилище);
- необходимость подключения к сети для сохранения и извлечения версий;
- отсутствие личных хранилищ у разработчиков (все изменения отправляются сразу в общее).

Распределенные системы управления версиями (*Distributed VCS, DVCS*) лишены подобных проблем: вся история изменения хранится в локальном хранилище на компьютере каждого пользователя и лишь избранные изменения отправляются на общий сервер (который, если вообще присутствует, является, по сути, хранилищем такого же типа, как и у пользователей).

Наиболее популярная распределенная *VCS*, *Git*, была создана Линусом Торвальдсом в 2005 году для управления разработкой ядра *Linux[2]*. Система, в соответствии с *UNIX*-идеологией, спроектирована как набор программ, подходящих для применения в скриптах, что позволяет использовать ее как часть другого программного продукта (например, системы документооборота) или создавать интерфейсы нужного типа.

Сильной стороной *Git* является мощная поддержка большого числа веток разработки и эффективная работа с ними. Git, в отличие от большинства других *VCS*, хранит историю не в виде набора файлов и их изменений (патчей), а, фактически, как снимки того, как выглядят все файлы проекта на текущий момент. Если файл не изменялся с прошлого коммита (фиксации изменений в хранилище), *Git* не сохраняет

файл снова, а просто ссылается на уже сохраненную версию[3]. Целостность файлов проверяется с помощью контрольных сумм *SHA*-1, вследствие чего даже малейшее изменение файла будет зафиксировано.

Файлы в *Git* могут находиться в одном из трех состояний (см. рисунок 1).

Рисунок 1. Рабочий процесс

Измененные файлы в рабочей директории добавляются в промежуточную область, из которой сохраняются в репозиторий. В свою очередь, из репозитория можно извлечь нужную версию проекта обратно в рабочую директорию.

Одним из главных преимуществ Git является удобная работа с ветвями разработки (и как следствие, удобная совместная разработка). Ветвь в *Git* – это легковесный указатель на коммит, который перемещается по ходу развития ветви дальше от родительского коммита.

Коммиты представляют собой объекты, ссылающиеся на другие объекты (деревья), которые, в свою очередь, хранят список всех объектов (файлов и других деревьев) этой версии.

Работа над исходными кодами в общем случае состоит из трех стадий:

- создание ветки (веток);
- работа над кодом;
- слияние.

Далее рассмотрен процесс разработки с использованием ветвления. Для каждого изменения (добавление функций, исправление ошибок и т.п.) создается отдельная ветка (см. рисунок 2).

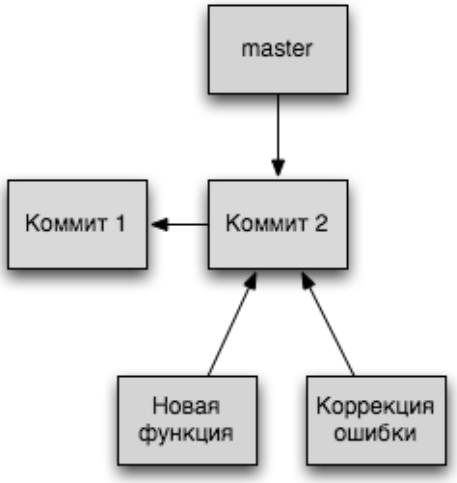


Рисунок 2. Создание веток

Затем, в каждой из веток ведется работа и последующее ее закрепление в хранилище (см. рисунок 3).

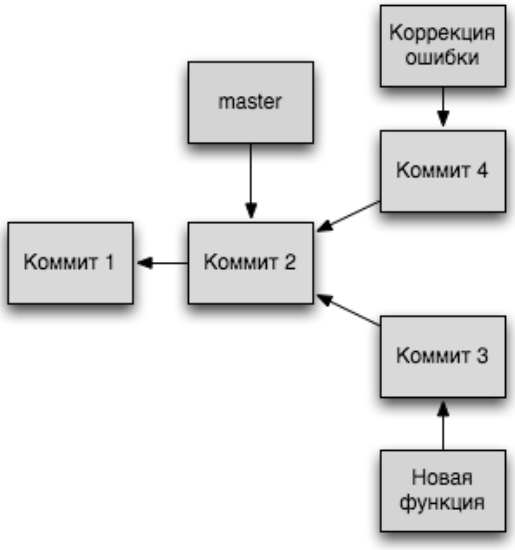


Рисунок 3. Работа в ветках

После выполнения необходимых изменений ветки поочередно сливаются с главной. В данном случае ветка *master* не изменялась с момента ответвления, конфликты не возникнут, поэтому ее объединение с веткой «Коррекция ошибки» происходит простым сдвигом указателя на позицию последнего коммита в этой ветке (см. рисунок 4).

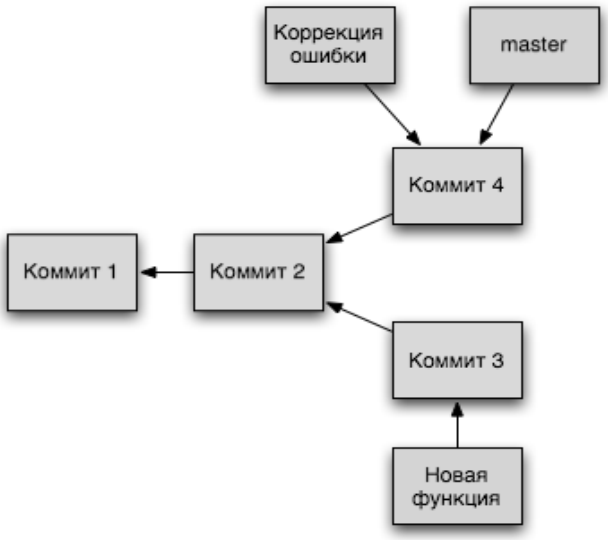


Рисунок 4. Слияние первой ветки

Ветку «Новая функция» таким способом объединить с основной уже нельзя: ветка *master* изменялась с момента ответвления, поэтому при слиянии возможны конфликты. Простые конфликты слияния (изменения разных файлов, либо изменение разных частей одного и того же файла) *Git* разрешает самостоятельно. Сложные (например, изменение одной и той же строки в файле в двух разных ветвях) разрешает самостоятельно разработчик, инициирующий процесс объединения. Слияние в данном случае оформляется как новый коммит (см. рисунок 5).

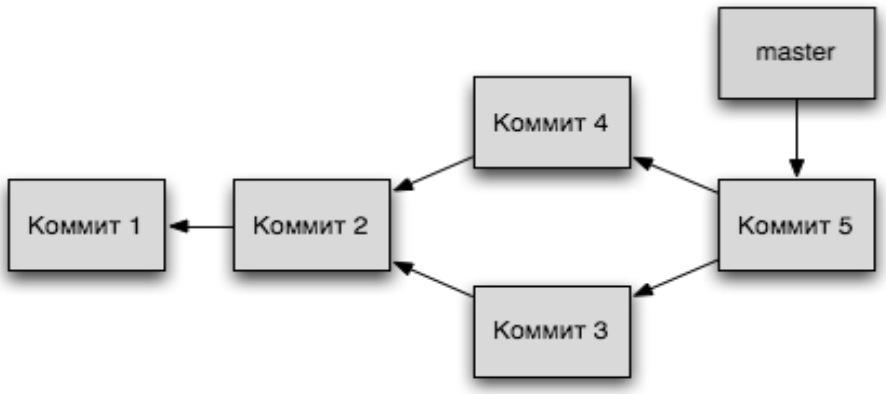


Рисунок 5. Слияние второй ветки

Особенностью такого подхода является то, что работа над различными ветвями может вестись независимо. Например, активное использование ветвления при разработке ядра ОС *Linux* позволяет различным командам (состоящим из десятков программистов, разбросанных по всему миру) работать над своей частью ядра без оглядки на остальных и затем объединять все изменения для выпуска новой версии продукта.

Кроме того, в такой процесс легко внедрить инспекцию кода: ответственный за проверку разработчик может просматривать все изменения перед слиянием с основной ветвью и, при наличии недочетов, отклонять слияние до тех пор, пока все они в новой ветви не будут исправлены. Такой подход активно применяется на сайте *GitHub.com* (крупнейший веб-сервис для совместной разработки): слияния оформляются как специальные запросы, в которых можно вести обсуждение и отклонять, либо принимать изменения с помощью веб-интерфейса. Помимо этого, сама команда *GitHub* при разработке использует дополнительный набор правил, позволяющий упорядочить и облегчить ведение множества веток[4].

Помимо этого, сервер с общим хранилищем может быть настроен так, чтобы специальным образом обрабатывать каждый принятый от разработчиков коммит. Ярким примером использования этой возможности являются системы непрерывной интеграции (такие как *Travis CI*), запускающие заранее определенный набор тестов при каждом коммите, что значительно повышает вероятность раннего выявления ошибок[5]. Кроме того, сервер можно настроить на автоматическую генерацию документации по добавляемому коду, а также на оповещение разработчиков о новой версии кода.

Выводы

Распределенные системы управления версиями значительно упрощают совместную разработку программного обеспечения за счет автоматизации выполнения рутинных действий, а также простого обмена версиями кода внутри команды. В свою очередь, универсальность подобных систем позволяет использовать их в качестве вспомогательного инструмента везде, где предполагается совместная работа над набором текстовых файлов, а затраты на внедрение в проект с длительным жизненным циклом многократно окупятся.

Литература

1. Scott Chacon. Pro Git. Apress, 2009. – 231 с.
2. Linus Torvalds. Kernel SCM saga [Электронный ресурс]. – URL: http://marc.info/?l=linux-kernel&m=111280216717070 (дата обращения: 14.12.2013).
3. Linux Kernel Development Community. Git Theory [Электронный ресурс]. – URL: https://git.wiki.kernel.org/index.php/GitTheory (дата обращения: 14.12.2013).
4. Scott Chacon. GitHub Flow [Электронный ресурс]. – URL: http://scottchacon.com/2011/08/31/github-flow.html (дата обращения: 14.12.2013).

5. Martin Fowler. Continuous Integration [Электронный ресурс]. – URL: http://martinfowler.com/articles/continuousIntegration.html (дата обращения: 14.12.2013).

УДК 81'322.2

МЕТОД СНИЖЕНИЯ РАЗМЕРНОСТИ МОДЕЛИ ТЕРМИН-ДОКУМЕНТ

К.М. Гречищев, Р.С. Самарев

Ключевые слова: модель термин-документ (vector space model), размерность (dimension), естественные языки (natural languages).

Аннотация: Рассматривается способ снижения размерности модели термин-документ. Идея метода заключается в удалении из модели всех термов, которые встречаются менее чем в k документах коллекции. Показано влияние снижения размерности на косинусную меру расстояния. Авторами получены аналитические выражения абсолютной погрешности расстояний при удалении одного терма. Способ тестировался на коллекции запросов в техническую поддержку. На практике достигнуто снижения размерности на 90%. Проведена оценка распределения изменения расстояний между документами при сниженной размерности. Проведено сравнение результатов работы двух алгоритмов кластеризации на исходной модели и модели со сниженной размерностью. Установлено, что разница в результатах не превышает 20%. Отмечено, что снижение размерности позволяет использовать освободившиеся ресурсы памяти и процессорного времени для того, чтобы сделать модель более гибкой.

Введение

Сегодня многие задачи, связанные с автоматической обработкой текстов на естественном языке, решаются с использованием модели термин-документ (англ. vector space model)[1]. В данной модели каждый документ представляется вектором в n-мерном пространстве термов. Недостаток модели, особенно заметный при обработке текстовых коллекций большого объема, состоит в том, что число уникальных термов быстро растет при увеличении размера коллекции. Это приводит к тому, что, зачастую, размерность векторов достигает десятков тысяч измерений. Обработка векторов такой длины требует большого расхода памяти и высоких затрат машинного времени.

В настоящее время существует ряд методов снижения размерности модели представления документов, например, методы, основанные на латентно-семантическом анализе и сингулярном разложении[2]. Недостатки данных методов состоят в потере части информации о документах, а также в потере четкой привязки размерности к конкретному уникальному терму, что затрудняет анализ результатов работы применяемого алгоритма, например, алгоритма кластеризации.

В данной работе предлагается метод снижения размерности, лишенный второго из приведенных выше недостатков. Метод заключается в исключении из модели уникальных терминов с низкой частотой появления в коллекции.

Модель термин-документ

Рассмотрим множество (коллекцию) документов D. Функцию $M : D \to V \subset R^m$, где m – число уникальных терминов коллекции D, называют моделью термин-документ. Тогда:

$\forall d_i \in D: \; M(d_i) = v_i = \begin{pmatrix} v_{i,1} \\ v_{i,2} \\ \dots \\ v_{i,m} \end{pmatrix}$, где величина $v_{i,j}$ может быть вычислена различными способами, например, по формуле $v_{i,j} = TF_{i,j} \cdot IDF_j$,

где $TF_{i,j}$ – частота термина в документе, IDF_j – инверсная частота термина в коллекции.

Величину v_i называют представлением документа d_i. Введем функцию расстояния (метрику) $\rho : V \times V \to R$. Расстоянием между документами будем называть значение этой функции для представлений этих документов.

В качестве функции расстояния обычно используют меру косинуса угла (англ. Cosine Distance) [1]:

$$\forall \vec{v_1}, \vec{v_2} \in V : \rho(\vec{v_1}, \vec{v_2}) = 1 - \cos\left(< \vec{v_1}, \vec{v_2}\right) = 1 - \frac{\vec{v_1} \cdot \vec{v_2}}{\left|\vec{v_1}\right| \cdot \left|\vec{v_2}\right|} = 1 - \frac{\sum_{i=1}^{m} v_{1,i} \cdot v_{1,i}}{\sqrt{\sum_{i=1}^{m} v_{i,1}^2} \cdot \sqrt{\sum_{i=1}^{m} v_{i,2}^2}}$$

Особенности текстовых коллекций

Интуитивно понятно, что не все слова в естественном языке употребляются с одинаковой частотой. Проведенные исследования[3] показывают, что распределение слов естественного языка по частоте подчиняется закону Ципфа (Zipf law), который является частным случаем распределения Парето:

$p(x,Q) = \begin{cases} \frac{\alpha}{Q} \cdot \left(\frac{Q}{x}\right)^{\alpha+1}, x \geq Q \\ 0, \quad x < Q \end{cases}$, где α и Q – параметры распределение Парето.

На практике отдельно взятая коллекция текстовых документов зачастую имеет специфические особенности, из-за которых распределение слов по частоте будет иметь другой вид. Кроме того, в модели термин-документ часто используются не только однословные термы, но и термы, состоящие из двух или трех слов.

В распоряжении авторов была коллекция из 12830 запросов в службу технической поддержки. Документы этой коллекции содержали большое количество серийных номеров, названий устройств и версий программного продукта, жаргонных выражений, а также опечаток. Кроме того, при анализе коллекции не использовался морфологический анализатор.

В таблице 1 приведено число уникальных термов, которые встречаются в текстовой коллекции. N – размер термина в словах. Видно, что от 72 до 90% термов встречаются лишь в одном документе из всей коллекции. Таким образом, если удалить эти редкие термы можно достичь значительного снижения размерности модели. Однако при этом необходимо оценить влияние удаляемых размерностей на меру расстояния между документами.

Распределение термов в конкретной коллекции Таблица 1

	N = 1		N = 2		N=3		N=1,2,3	
Всего термов	33 199	100.00%	140 464	100.00 %	162 689	100.00 %	336 352	100.00%
Число термов, встречающихся лишь в одном документе	23 908	72.01%	124 736	88.80 %	156 480	96.18 %	305 124	90.72%
Лишь в 2-х документах	3 017	9.09%	9 637	6.86%	4 478	2.75%	17 132	5.09%
Лишь в 3-х документах	1 397	4.21%	2 849	2.03%	989	0.61%	5 235	1.56%
Лишь в 4-х документах	775	2.33%	1 175	0.84%	270	0.17%	2 220	0.66%
В 5-и и более документах	*4 102*	*12.36%*	*2 067*	*1.47%*	*472*	*0.29%*	*6 641*	*1.97%*

Оценка метода

Итак, сущность предлагаемого подхода заключается в удалении из модели всех размерностей, соответствующих терминами, частота появления которых в коллекции

ниже заданной. С формальной точки зрения удаление из модели представления одного из терминов (l-го термина t_l) эквивалентно переходу от модели представления M к модели $M_1 : D \to \Lambda \subset R^{m-1}$. Тогда:

$M_1(d_i) = \lambda_i = \begin{pmatrix} \lambda_{i,1} \\ \dots \\ \dots \\ \dots \\ \lambda_{i,m-1} \end{pmatrix}$, где вектор λ_i может быть получен из вектора v_i вычеркиванием l-го элемента v_l.

Рассмотрим эффект, который оказывает удаление термина на величину расстояния между документами в общем случае:

$$\cos(<\vec{v_1}, \vec{v_2}) = \frac{\vec{v_1} \cdot \vec{v_2}}{\left|\vec{v_1}\right| \cdot \left|\vec{v_2}\right|};$$

$$\cos(<\vec{\lambda_1}, \vec{\lambda_2}) = \frac{\vec{v_1} \cdot \vec{v_2} - v_{1,l} v_{2,l}}{\sqrt{\left|\vec{v_1}\right|^2 - v_{1,l}^2} \cdot \sqrt{\left|\vec{v_2}\right|^2 - v_{2,1}^2}}.$$

Обозначим буквой Δ абсолютную погрешность расстояния. Тогда в общем случае:

$$\Delta = \rho(\vec{v_1}, \vec{v_2}) - \rho(\vec{\lambda_1}, \vec{\lambda_2}) = 1 - \cos(<\vec{v_1}, \vec{v_2}) - 1 + \cos(<\vec{\lambda_1}, \vec{\lambda_2}) =$$

$$= \frac{\vec{v_1} \cdot \vec{v_2} - v_{1,l} v_{2,l}}{\left|\vec{v_1}\right|\left|\vec{v_2}\right| \sqrt{1 - \frac{v_{1,l}^2}{\left|\vec{v_1}\right|^2}} \cdot \sqrt{1 - \frac{v_{2,1}^2}{\left|\vec{v_2}\right|^2}}} - \frac{\vec{v_1} \cdot \vec{v_2}}{\left|\vec{v_1}\right| \cdot \left|\vec{v_2}\right|} = \frac{\vec{v_1} \cdot \vec{v_2} \cdot \left(1 - \sqrt{1 - \frac{v_{1,l}^2}{\left|\vec{v_1}\right|^2}} \cdot \sqrt{1 - \frac{v_{2,1}^2}{\left|\vec{v_2}\right|^2}} \right) - v_{1,l} v_{2,l}}{\left|\vec{v_1}\right|\left|\vec{v_2}\right| \sqrt{1 - \frac{v_{1,l}^2}{\left|\vec{v_1}\right|^2}} \cdot \sqrt{1 - \frac{v_{2,1}^2}{\left|\vec{v_2}\right|^2}}} \quad (1)$$

Если ни один из документов не содержал терма t_l, то

$\forall\ d_1, d_2 \in D : t_l \notin d_1, t_l \notin d_2 \Rightarrow v_{1,l} = 0, v_{2,l} = 0 \Rightarrow \Delta = 0$

Следовательно, при удалении измерения l расстояние между документами, не содержащими терма t_l, не изменяется. Это значит, что данная метрика не выявляет сходство документов на основании того, что в них отсутствует один и тот же терм.

В промежуточном случае, когда термин t_l содержится лишь в одном документе, имеем:

$$\forall\ d_1,d_2 \in D : t_l \in d_1, t_l \notin d_2 \Rightarrow v_{1,l} \neq 0, v_{2,l} = 0 \Rightarrow$$

$$\Delta = \frac{\vec{v_1}\cdot\vec{v_2}\cdot\left(1-\sqrt{1-\frac{v_{1,l}^2}{\left|\vec{v_1}\right|^2}}\right)}{\left|\vec{v_1}\right|\left|\vec{v_2}\right|\sqrt{1-\frac{v_{1,l}^2}{\left|\vec{v_1}\right|^2}}} = \frac{\vec{v_1}\cdot\vec{v_2}}{\left|\vec{v_1}\right|\left|\vec{v_2}\right|}\cdot\left(\sqrt{\frac{\left|\vec{v_1}\right|^2}{\left|\vec{v_1}\right|^2 - v_{1,l}^2}} - 1\right) \qquad (2)$$

На практике при формировании векторов $\vec{v_1}$ и $\vec{v_2}$ применяют процедуру нормализации, следовательно $\left|\vec{v_1}\right| = \left|\vec{v_2}\right| = 1$, тогда формула (1) существенно упроститься:

$$\Delta = \frac{\vec{v_1}\cdot\vec{v_2}\cdot\left(1-\sqrt{1-v_{1,l}^2}\cdot\sqrt{1-v_{2,1}^2}\right)-v_{1,l}v_{2,l}}{\sqrt{1-v_{1,l}^2}\cdot\sqrt{1-v_{2,1}^2}}.$$

Аналогично для (2):

$$\Delta = \frac{\vec{v_1}\cdot\vec{v_2}\cdot\left(1-\sqrt{1-v_{1,l}^2}\right)}{\sqrt{1-v_{1,l}^2}}.$$

Эксперименты и результаты

Для проверки метода были поставлены два эксперимента. Первых из них заключался в последовательном удалении из модели документ всех терминов, которые встречаются менее чем в 2-х, 3-х, 4-х документах из всей коллекции. При этом оценивалось абсолютное и относительно изменение расстояний между всеми парами документов. Гистограммы абсолютного изменения расстояний приведены на рисунках 1-3. Отметим, что на всех приведенных гистограммах не показаны те расстояния, которые вообще не изменились. Количество таких расстояний составляло 77 624 778 при пороге удаления терма равном 2 документа, 77 268 294 при пороге в 3 документа и 77 105 375 при пороге в 4 документа. Общее число расстояний в коллекции составляло 82 298 035. Следовательно, даже в худшем из рассмотренных случаев 93,7% расстояний остались неизменными.

Рисунок 1. Изменение расстояний при удалении терминов, встречающихся менее чем в 2-х документах

Рисунок 2. Изменение расстояний при удалении терминов, встречающихся менее чем в 3-х документах

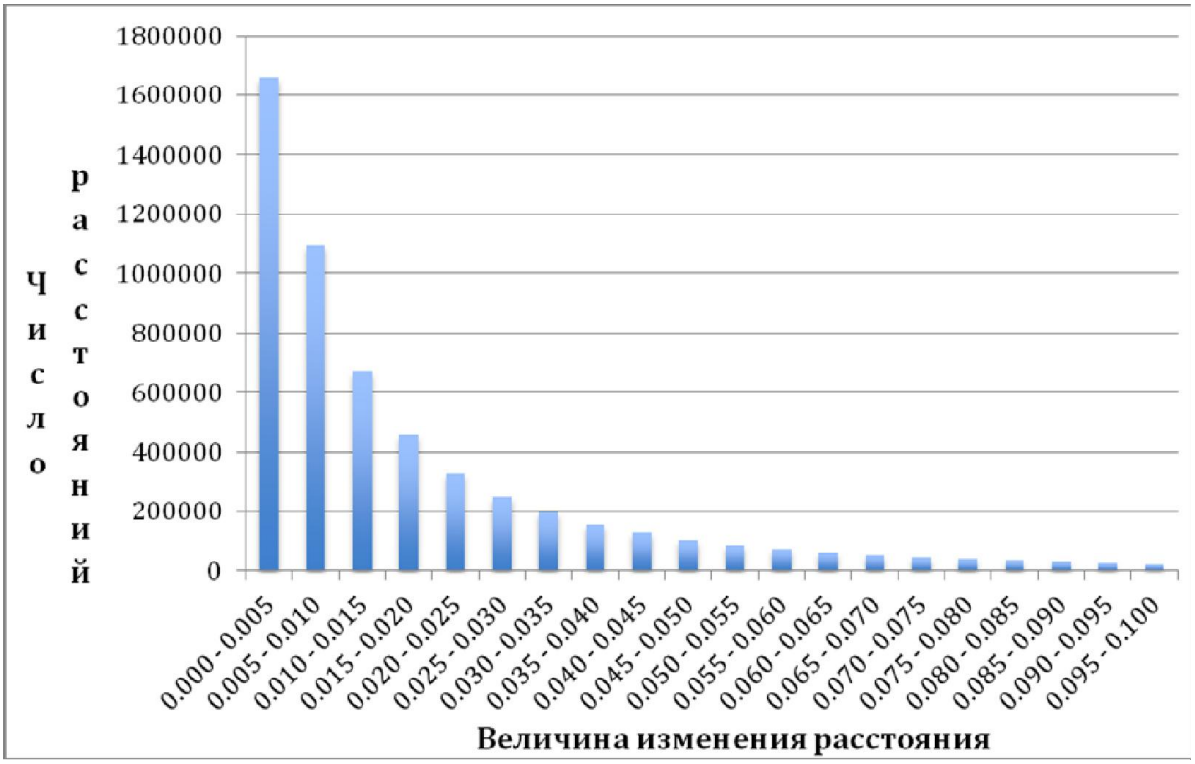


Рисунок 3. Изменение расстояний при удалении терминов, встречающихся менее чем в 4-х документах

Видно, что подавляющее большинство расстояний изменилось менее, чем на 0.01, что при диапазоне изменения меры от 0 до 1 является значительным. На рисунках 4 и 5 приведены гистограммы относительного изменения расстояний между документами.

Рисунок 4. Относительное изменение расстояний при удалении терминов, встречающихся менее чем в 2-х документах

Рисунок 5. Относительное изменение расстояний при удалении терминов, встречающихся менее чем в 4-х документах

Приведенные гистограммы показывают, что большая часть расстояний изменяется менее, чем на 10% в худшем из рассмотренных случаев.

Второй эксперимент заключался в оценке влияния удаленных компонентов на качество работы алгоритмов машинного обучения. Была исследована работа алгоритмов K-средних[4] и Meanshift[5] на исходной модели термин-документ и моделях, полученных путем удаления термов.

Результаты кластеризации, полученные на основе неизменной модели, считались эталонными. С ними сравнивались результаты работы алгоритма на модели с удаленными терминами. Для сравнения двух кластерных структур использовались метрики precision, recall и F1[6]:

$$precision(i,j) = \frac{n_{i,j}}{n_i} \qquad precision = \sum_j \frac{n_j}{N} \cdot \max_i precision(i,j)$$

$$recall(i,j) = \frac{n_{i,j}}{n_j} \qquad recall = \sum_j \frac{n_j}{N} \cdot \max_i recall(i,j)$$

$$F1(i,j) = \frac{2 \cdot precision(i,j) \cdot recall(i,j)}{precision(i,j) + recall(i,j)} \qquad F1 = \sum_j \frac{n_j}{N} \cdot \max_i F1(i,j)$$

в сравниваемой кластерной структуре, n_{ij} – число документов, принадлежащих i-му где n_i – размер i-го кластера в эталонной кластерной структуре, n_j – размер j-го кластера кластеру в эталонной кластерной структуре и одновременно j-му кластеру в

сравниваемой, N – число документов к коллекции. Результаты эксперимента приведены на рисунках 6 и 7.

Как показывают приведенные гистограммы, результаты, полученные на измененной модели термин документы, отличаются от исходных на 15-20 %. Можно утверждать, что данная величина зависит от особенностей и параметров конкретного алгоритма машинного обучения.

С сожалению, в распоряжении авторов не было построенного экспертами эталонного разбиения документов на кластеры, поэтому невозможно сказать, отличаются ли новые результаты от исходных в лучшую или худшую сторону. Однако можно утверждать, что сниженная размерность модели позволяет добавить в нее новую информацию, на пример включить многословные термы. Таким образом, использование приведенного метода снижения размерности делает модель термин-документ более гибкой, что позволяет инженеру более точно адаптировать ее под конкретную задачу и получить более качественные результаты.

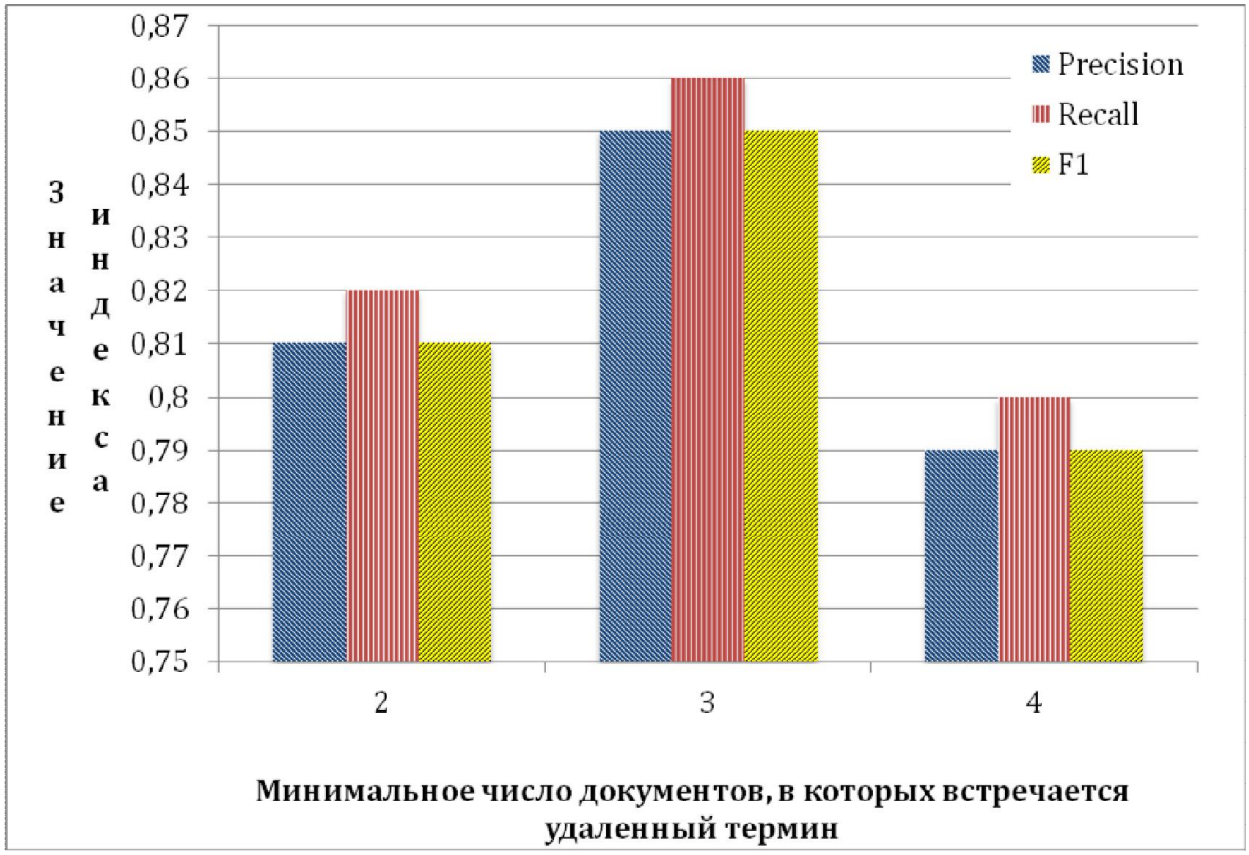


Рисунок 6. Оценка качества работы алгоритма К-средних при разбиении на 2 кластера

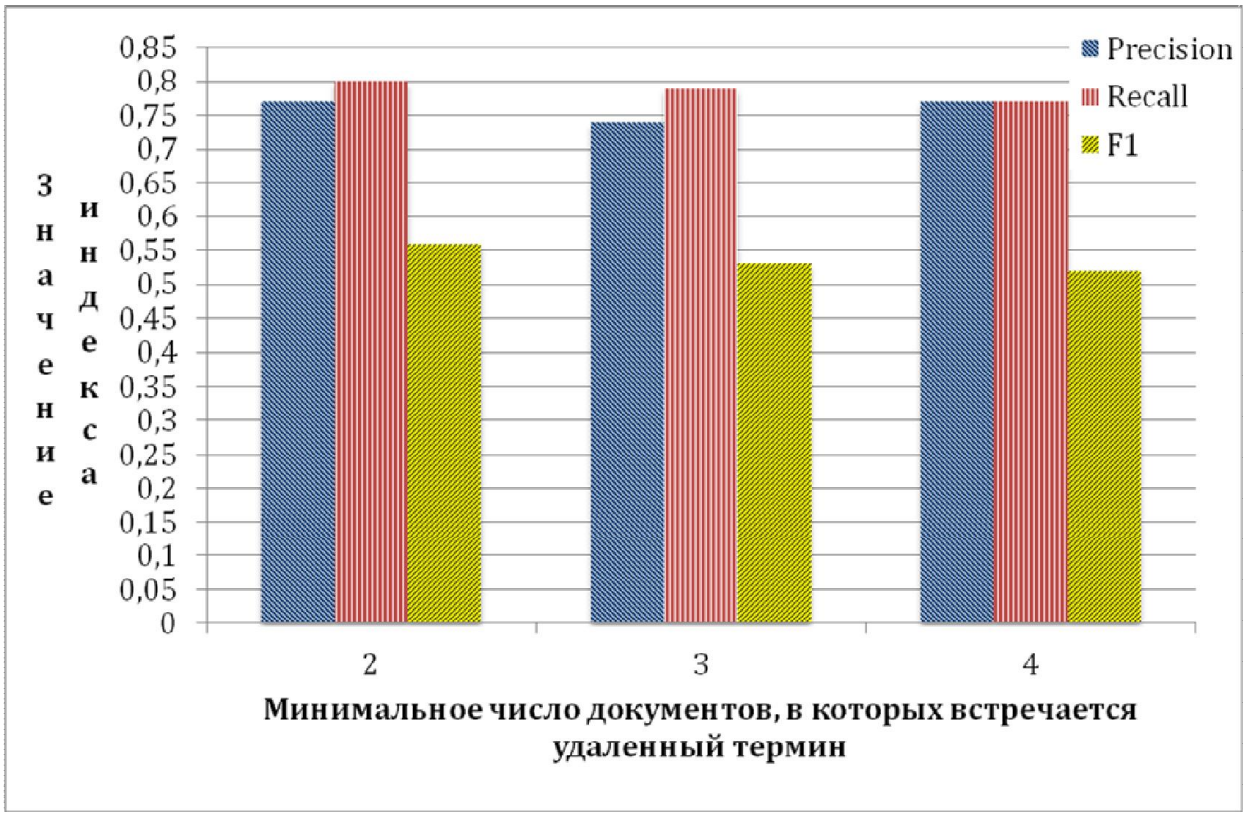


Рисунок 7. Оценка качества работы алгоритма MeanShift

Недостатком приведенного подхода является удаление части терминов из словаря. Данные термины, как было показано выше, не оказывают значительного влияния на меру расстояния, но могут быть важны для последующего анализа результатов работы алгоритма. Однако это недостаток может быть компенсирован использованием словаря термов, удаление которых запрещено, или повторной индексацией документов перед анализом результатов.

Выводы

Рассмотрен метод снижения размерности модели термин-документ. Получены теоретические оценки абсолютной погрешности расстояний при удалении из коллекции одного терма.

Подход проверен на тестовой коллекции из 12 тысяч запросов в техническую поддержку. Получена оценка распределения абсолютного и относительного изменения расстояний между документами коллекции при удалении из модели всех термов с частотой встречаемости ниже заданного порога. Было достигнуто значительное снижение размерности модели, при этом удалось сохранить неизменными 93,7 расстояний.

Проведено сравнение результатов работы двух алгоритмов кластеризации на исходной и измененной модели термин документ. Установлено, что различие между полученными результатами составило порядка 15-20%. Замечено, что использование

метода снижения размерности модели позволяет более точно адаптировать ее под решаемую задачу.

Предложенный подход может быть использован для снижения размерности модели термин-документ при использовании косинусной меры расстояния в задачах классификации, кластеризации текстовых документов.

Литература

1. Feldman R., Sanger J. The Text Mining Handbook, Cambridge University Press, New York, 2007, - 410 стр.
2. Aggarwal C., Zhai C. A survey of text clustering algorithms. – 52 стр. [Электронный ресурс]. – URL: http://charuaggarwal.net/text-cluster.pdf (дата обращения: 18.11.2013).
3. Zipf G.K., Human Behavior and the Principle of least Effort (Addison-Wesley, New York, 1949).
4. Ту Дж., Гонсалес Р. "Принципы распознавания образов", Издательство "Мир", Москва 1978, - стр. 109-112.
5. Comaniciu D., Meer. P. Mean shift: A robust approach toward feature space analysis. IEEE Trans. Pattern Anal. Machine Intell. 24, 2002, - стр. 603–619.
6. Сивоголовко Е. Оценка качества четкой кластеризации. Семинар Московской Секции ACM SIGMOD, 24 ноября 2011 г. [Электронный ресурс]. – URL: http://synthesis.ipi.ac.ru/sigmod/seminar/s20111124 (дата обращения: 5.12.2013).

УДК 004.021

ОРГАНИЗАЦИЯ ПАРАЛЛЕЛЬНОЙ ОБРАБОТКИ ЗАПРОСОВ В ЗАДАЧАХ ВЫЯВЛЕНИЯ ПЛАГИАТА

И.А. Евграфов, Р.С.Самарев

Ключевые слова: выявление плагиата, распределённые вычисления, веб-сервис.

Keywords: plagiarism detection, distributed computing, web-service.

Аннотация: алгоритмы выявления плагиата являются требовательными к объему ОЗУ и ресурсам ЦП. В статье рассматривается способ создания вычислительного кластера,

позволяющий существенно расширить объемы обрабатываемых данных. Автор уделяет основное внимание разработке эффективно масштабируемого веб-сервиса для выявления плагиата.

Точность анализа документов на наличие плагиата сильно зависит от используемого алгоритма. Наиболее точные алгоритмы требуют значительных объёмов оперативной памяти и ресурсов процессора. Поэтому для повышения эффективности большинства программных средств выявления плагиата необходимо использовать в процессе анализа наиболее точные алгоритмы, что влечёт за собой потребность в таком количестве ресурсов, которое часто недоступно одной вычислительной машине. Одним из возможных решений этого затруднения является создание вычислительного кластера, способ создания которого рассмотрен в данной статье.

Основой для создания кластера является разработанный ранее веб-сервис по выявлению плагиата. Именно его модификация для возможности горизонтального масштабирования рассматривается в данной работе. Сервис выявления плагиата использует достаточно простой алгоритм Рабина-Карпа с ускоряющими модификациями в виде шинглов, позволяющими его эффективно использовать в задаче выявления плагиата [1]. Модифицированный алгоритм имеет среднюю точность, однако он является высокопроизводительным и довольно простым для реализации, поэтому великолепно подходит для исследования алгоритмов выявления совпадений.

Разработанный ранее веб-сервис имеет клиент-серверную архитектуру, то есть поиск текстов и их сравнение осуществляется на стороне сервера, где находится так же и база данных. Каждый из анализируемых документов разбирается на шинглы, от которых затем вычисляются хеш-суммы. После этого идёт поиск в базе текстовых документов, в которых встречаются те же хеш-суммы. Фиксированное количество наиболее схожих документов (определяется по мощности пересечения множеств хеш-сумм) отдаётся клиенту в качестве результата поиска. В качестве базы данных используется зарекомендовавшее себя *key-value* хранилище *Oracle Berkeley DB,* которое обеспечивает достаточно высокий уровень производительности, пока размер данных не превосходит объём доступной оперативной памяти.

Реализация сервиса является однопоточной, то есть в один момент времени ведётся обработка не более чем одного запроса. Если следующий запрос приходит до окончания обработки текущего, то он помещается в очередь.

Клиент-серверная архитектура привносит с собой ряд проблем, некоторые из которых делают невозможным создание кластера на основе текущего решения:

• вертикальное масштабирование в этом варианте возможно для базы данных – увеличение оперативной памяти увеличит максимальный размер базы документов;

• вертикальное масштабирование логики сервиса затруднено: однопоточная реализация не позволяет целиком реализовать потенциал современных процессоров с несколькими вычислительными ядрами. Кроме того, обычно нехватка оперативной памяти является более существенной, чем недостаточная производительность процессоров;

• горизонтальное масштабирование невозможно из-за базы данных: несмотря на все свои достоинства (производительность, минимальные затраты процессорного времени и оперативной памяти) *Oracle Berkeley DB*, изначально используемая в сервисе, не имеет сетевой реализации.

По ряду вышеперечисленных причин необходимо привести сервис к масштабируемому варианту, примером которой может служить трехуровневая архитектура с отдельной сетевой базой данных (рисунок 1).

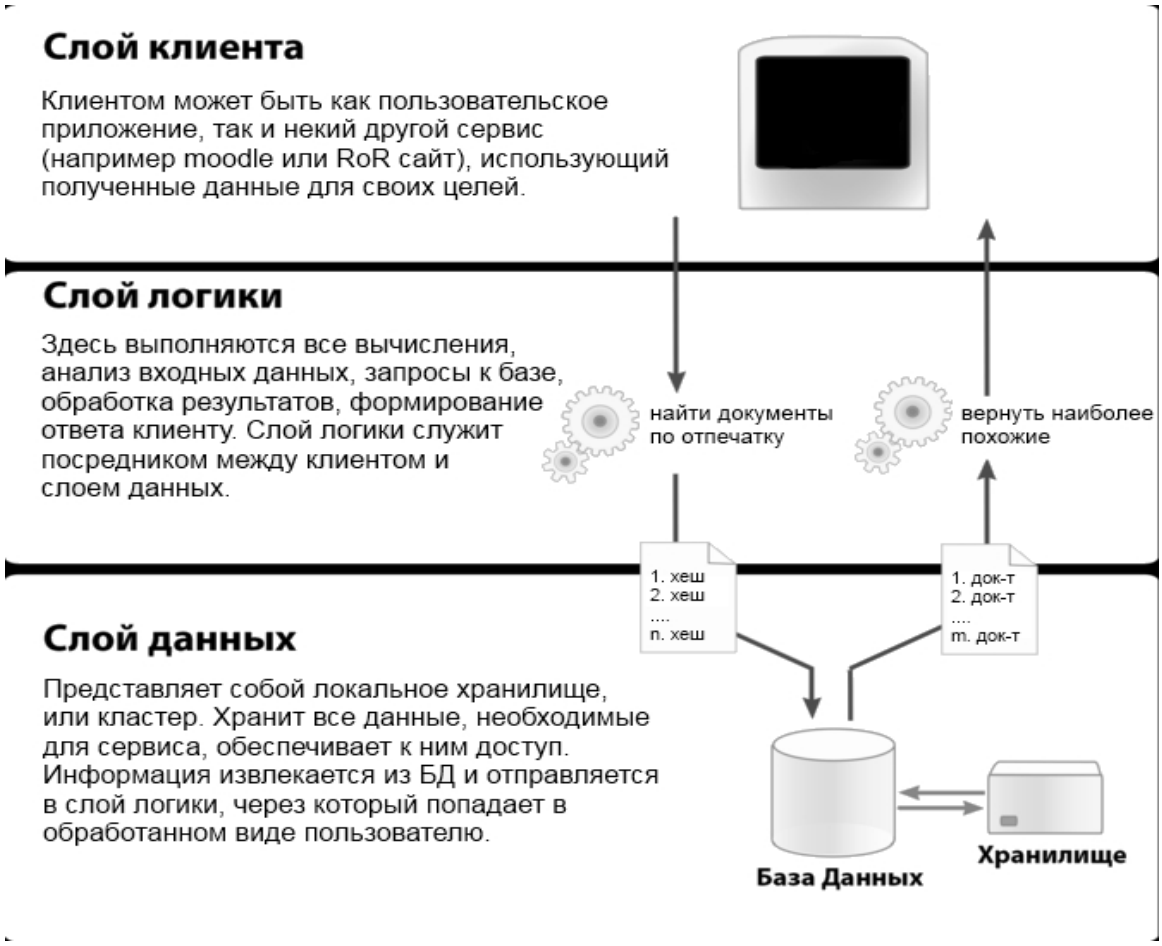


Рисунок 3. Пример трёхуровневой архитектуры

В первую очередь необходимы изменения в системе хранения, чтобы устранить

основную причину невозможности горизонтального масштабирования. Для алгоритма Рабина-Карпа в качестве варианта на замену могут быть рассмотрены распределённые *key-value* хранилища, такие как *Apache Cassandra [2], Redis Cluster [5], Infinispan [6], Hibari [7]*. В процессе разработки была выбрана база данных *Redis Cluster*. Это стало следствием нескольких причин: подходящий для реализации алгоритма набор структур данных, наличие транзакций, прямой доступ к каждой из ячеек кластера.

Внутри сервиса для взаимодействия с *Redis* отведён отдельный поток. На рисунке 2 он показан как внутренний клиент базы данных. Потоки-обработчики, когда им необходим доступ к базе, формируют задания и помещают их в очередь, откуда их получает клиент базы данных и выполняет.

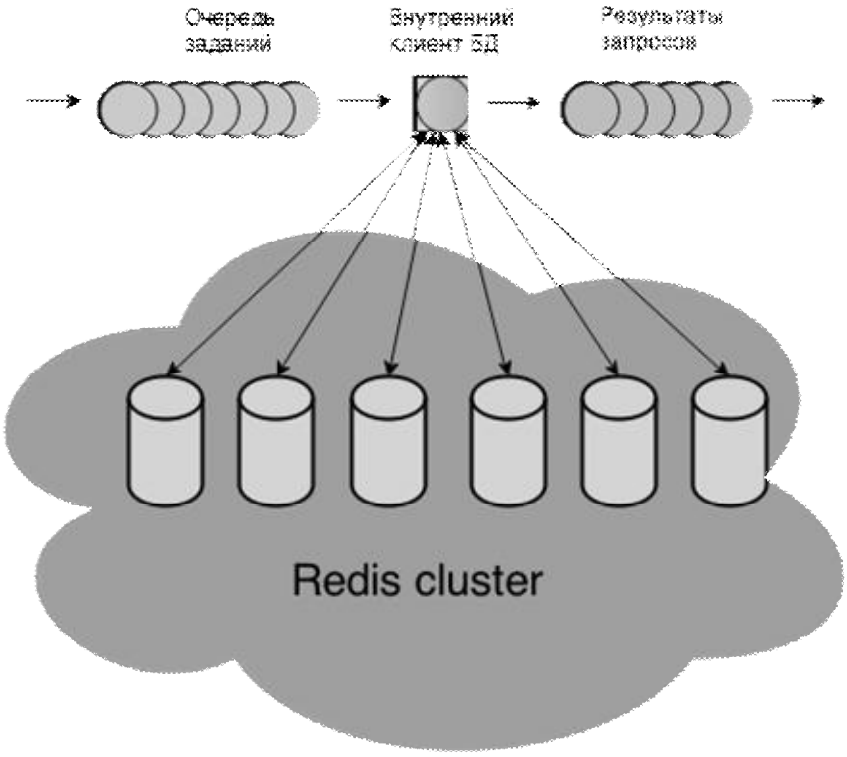


Рисунок 2. Схема осуществления запросов к базе данных

Реализация очереди требует безопасного её использования одновременно из нескольких потоков. Прежде чем передать задание на обработку, поток-отправитель добавляет в него необходимое количество элементарных запросов. Отведение отдельного потока исключительно для работы с базой данных имеет следующие преимущества по сравнению с использованием отдельного клиента внутри каждого обработчика:

- ресурсы на соединение с нодами (отдельные машины, составляющие кластер) расходуются однократно и используются для всех потоков, которым нужен доступ к базе;
- только один поток выполняет перестроение связей с кластером базы данных, если в нём проводится реструктуризация.

Так как *Redis* не перенаправляет запросы внутри кластера, то необходимо на стороне клиента определять адресата, чтобы запросы всегда попадали в нужную ноду, которая и должна содержать искомые данные. Поэтому в сервисе для использования *hiredis* (клиент *Redis* на языке *C*) был написан класс, обеспечивающий интерфейс для взаимодействия с базой данных, поддерживающий соединение со всем кластером и реагирующий на любые реструктуризации: добавление машин, сокращение их числа, замена ведущего ведомым, перераспределение данных внутри кластера. Это позволяет проводить любые работы с базой данных без остановки сервиса. Сам *Redis* также позволяет проводить реструктуризации во время работы кластера.

Реализованное решение позволяет распределять данные на множество машин практически без потерь в производительности каждой отдельной ноды кластера. Целостность данных обеспечивается транзакциями, в которые сервис оборачивает все цепочки запросов (фактически каждое задание выполняется в ходе отдельной транзакции). Отказоустойчивость достигается дублированием по схеме ведущий-ведомый средствами *Redis Cluster*. Таким образом, использованная база данных целиком удовлетворяет потребности разрабатываемого сервиса.

Кроме замены базы данных необходимо адаптировать сервис для использования всех доступных ресурсов вычислительной системы, в частности, эффективное использование всех процессоров и их вычислительных ядер. Текущее решение позволяет запустить на одной вычислительной системе несколько экземпляров приложения, что действительно использует все доступные ядра, но требует некоторых дополнительных условий:

- все экземпляры используют разные сетевые порты;
- запросы должны проходить через балансировщик нагрузки, который будет заниматься их распределением по экземплярам-обработчикам;
- каждый запущенный экземпляр должен иметь свои соединения с нодами *Redis Cluster*, что влечёт за собой дополнительные на них затраты.

Поэтому предпочтительным будет вариант приложения, один экземпляр которого целиком использует все доступные ресурсы процессора. Для этого обработка запросов

должна задействовать одновременно несколько ядер. Возможны несколько вариантов реализации этого:

- Параллельное выполнение анализа:

  Алгоритм:

  1. Процесс обработки разбивается на несколько частей;
  2. Для каждой из частей запускается отдельный поток;
  3. Из совокупности результатов, полученных каждой частью, формируется ответ на запрос.

  Преимущества:

  1. Уменьшение времени обработки каждого запроса пропорционально количеству потоков;
  2. Равномерная загрузка ядер.

  Недостатки:

  1. Область применения ограничена узким спектром алгоритмов;
  2. Значительные затраты на синхронизацию в случаях, когда части имеют разделяемые данные.

- Индивидуальный поток для каждого соединения:

  Алгоритм:

  1. Управляющий поток принимает соединение от клиента;
  2. Создаётся поток для обработки запроса;
  3. В поток передаётся дескриптор открытого соединения;
  4. После завершения обработки поток завершается;

  Преимущества:

  1. Увеличение числа обрабатываемых запросов в единицу времени;
  2. Нет необходимости в синхронизации.

  Недостатки:

  1. Непосредственно время обработки запроса остаётся не выше, чем у однопоточной реализации;
  2. Резкое падение производительности при росте числа соединений (ситуация, когда количество потоков многократно превосходит число ядер).

- Использование пула потоков:

  Алгоритм:

  1. Управляющий поток принимает соединения от клиента;
  2. Дескриптор соединения добавляется в очередь запросов;

3. Свободный поток из заранее созданного пула забирает запрос из очереди;
4. После выполнения запроса поток ждёт появления новых дескрипторов в очереди.

Преимущества:

1. Увеличение числа обрабатываемых запросов в единицу времени;
2. Малые затраты на синхронизацию (потокобезопасная очередь).

Недостатки:

1. Непосредственно время обработки запроса остаётся на уровне однопоточной реализации.

В первую очередь стоит отметить, что использование пула потоков оказывается более универсальным и приспособленным к нагрузкам решением, чем запуск потока под каждое соединение. Алгоритм Рабина-Карпа можно выполнять параллельно в нескольких потоках на стадиях: канонизация, вычисление хеш-функций, которые являются наиболее требовательными к процессорному времени. Поэтому в данной реализации сервиса вполне возможен первый вариант решения. Но, не смотря на это, для сохранения общности (далеко не все алгоритмы настолько же приспособлены к распараллеливанию) в сервисе использован третий подход, то есть используется пул потоков (рисунок 3). Для передачи дескрипторов соединений обработчикам используется обобщённая потокобезопасная очередь [3]. При необходимости получить данных из базы или сохранить их в неё, поток формирует запрос и запрашивает его выполнение. Таким образом, все потоки используют один комплект соединений с нодами кластера хранения, что позволяет уменьшить затраты.

Структура разработанного решения позволяет построить на своей основе вычислительный кластер, использующий для хранения распределённую базу данных. Производительность сервиса теперь зависит как от количества ячеек кластера, так и от ресурсов каждой отдельной машины, на которой запущен экземпляр. Для создания полноценного вычислительного кластера из n машин-обработчиков необходимо n+m+1 компьютеров, где m – количество нодов, отведённых под хранилище данных, а 1 используется как единая точка входа для всех запросов от конечных клиентов (рисунке 4).

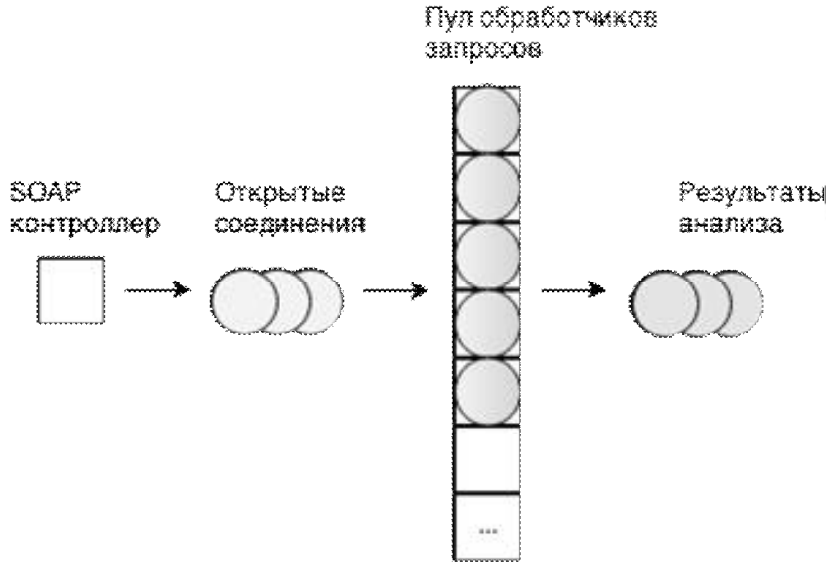


Рисунок 3. Схема приёма и обработки сервисом запросов конечных клиентов

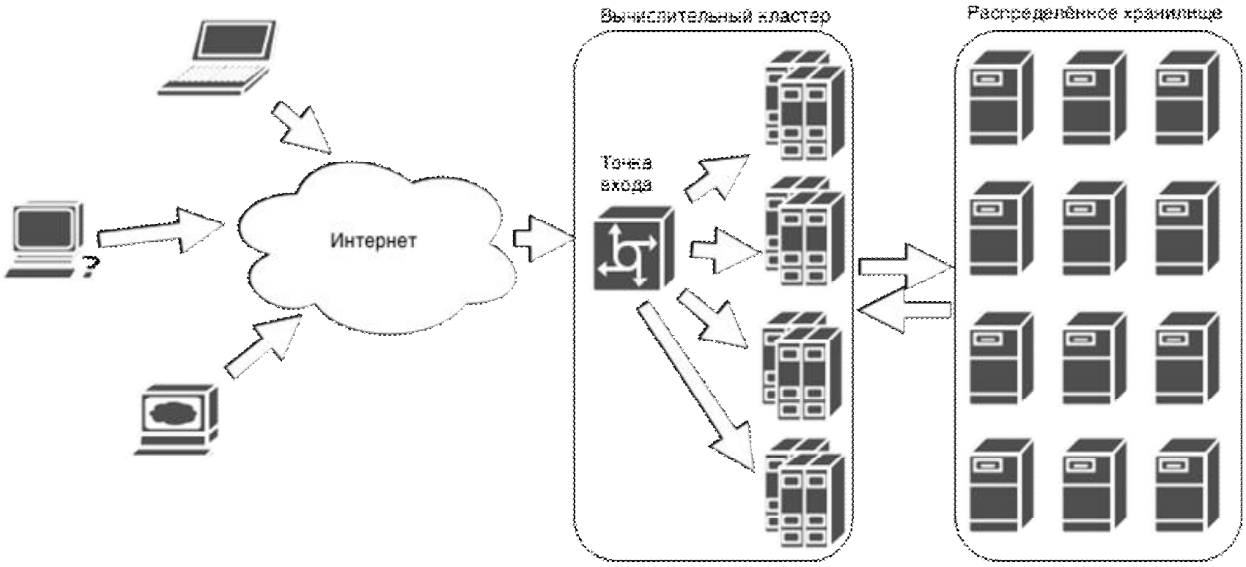


Рисунок 4. Пример структуры вычислительного кластера с распределенным хранилищем на основе разработанного решения

Производительность кластера линейно зависит от количества нод, но имеет предел, который определяется максимальным количеством перенаправляемых балансировщиком нагрузки запросов. Время обработки каждого отдельного запроса остаётся тем же, что у одного сервера, но количество одновременно обрабатываемых возрастает с числом машин.

Было проведено исследование производительности сервиса на виртуальной машине, которой выделялось от 1 до 8 ядер. База данных и программа, моделирующая клиент, были расположены на другом компьютере. В качестве тестового набора использовались случайные наборы заданного количества слов, которые комбинировались

в тексты, чтобы смоделировать заимствование. Количество ежесекундно обрабатываемых в ходе тестирования запросов приведено в таблице 1.

Производительность сервиса на многоядерных системах Таблица 1

Количество ядер	Размер текстов тестового набора (Кбайт)	
	16	1600
1	65	6
2	119	11
4	241	22
8	430	42

Верхний предел эффективности вертикального масштабирования решения можно оценить, воспользовавшись законом Амдала [4]. Доля последовательной части кода стремится к нулю, поэтому она не учитывается в формуле. Следует так же отметить, что сервис требует не менее трёх ядер для действительно параллельного выполнения различных потоков. Полученная оценка справедлива только для таких машин. Таким образом, увеличение производительности можно найти следующим образом:

$N = 1/\max(P_{ACC}, P_{BD}, \frac{P_{proc}}{n-2})$, где P_{BD} – доля кода, выполняющего запросы к базе данных, P_{ACC} – доля приёмника запросов, P_{proc} – доля обработки, а n – количество вычислительных ядер. Получить доли кода возможно в ходе профилирования (определение суммарного времени выполнения каждой функции в ходе работы сервиса). Экспериментально определённые в ходе профилирования доли кода приведены в таблице 2. По полученным значениям, определён верхний предел вертикального масштабирования реализованного сервиса: максимальное увеличение производительности в 25 раз достигается при использовании 27 вычислительных ядер.

Доли кода, выполняемого в различных потоках относительно всей программы Таблица 2

P_{BD}	P_{ACC}	P_{PROC}
0,0384	0,0023	0,9593

Таким образом, разработанная программа делает возможным эффективное использование для выявления плагиата значительного количества вычислительных ресурсов, что позволяет значительно увеличить как точность анализа, так и объёмы обрабатываемы

Литература

1. Подольский В.Э., Самарев Р.С. Использование алгоритма Рабина-Карпа для задачи выявления плагиата // Современные информационные технологии: Сб. трудов каф. ИУ-6. – 2010.
2. The Apache Cassandra Project [Электронный ресурс].– URL.http://cassandra.apache.org/ (дата обращения: 21.11.2013).
3. Williams A. C++ Concurrency in Action // Manning Publications Co. – 2012. – 530p.
4. Черняк Л. Закон Амдала и будущее многоядерных процессоров // Открытые системы. СУБД.2008.URL.http://www.osp.ru/os/2009/04/9288815/ (дата обращения: 21.11.2013).
5. Redis cluster Specification [Электронный ресурс].– URL: http://redis.io/topics/cluster-spec (дата обращения: 21.11.2013).
6. Infinispan Homepage [Электронный ресурс].– URL: http://infinispan.org/ (дата обращения: 21.11.2013).
7. Hibari Home Page [Электронный ресурс].– URL: http://hibari.github.io/hibari-doc/ (дата обращения: 21.11.2013).

УДК 004.04

СОЗДАНИЕ СЛОЖНЫХ БИЗНЕС-ПРОЦЕССОВ С ИСПОЛЬЗОВАНИЕМ ACTIVITI

П.И. Коняев, Р.С. Самарев

Ключевые слова: документооборот (document workflow), бизнес-процессы (business-process), Activiti, Eclipse.

Аннотация: Статья посвящена обзору технологии Activiti для создания бизнес-процессов. Рассмотрен процесс создания сложных бизнес процессов, включая параллельные процессы выполнения, создание визуальных форм для бизнес-процессов, а также организацию сложной логической модели. Рассмотрена интеграция сложных бизнес процессов в системе документооборота на примере Alfresco ECM.

Введение

Бизнес-процесс – это совокупность взаимосвязанных мероприятий или задач, направленных на создание определенного продукта или услуги для потребителей.

Alfresco ECM – тиражируемая интегрированная система управления контентом для организаций разработки одноимённой британской компании. Используется для управления документами, записями, веб-публикацией, групповой работой и бизнес-процессами в организации. Существует в двух редакциях. Alfresco Community является свободной, распространяется на условиях LGPL. Редакция Alfresco Enterprise является платной коммерческой версией продукта, распространяется под своей проприетарной лицензией, имеет открытый исходный код и соответствует открытым стандартам.

Activiti

Activiti – это средство управление бизнес-процессами (BPM), иначе называемое «движок». Платформа ориентирована на деловых людей, разработчиков и системных администраторов. Ее ядром является быстрый и надежный «движок» BPMN 2. Activiti распространяется под лицензией Apache с открытым исходным кодом. Activiti работает в любом приложение Java, на сервере, на кластере серверов или в облаке. Он может быть интегриррован со средствами Spring для разработки интерфейса пользователя, не требует больших вычислительных ресурсов. Для разработки бизнес-процессов с использованием Activiti имеется средство разработки с удобным интерфейсом. Данный движок используется в Alfresco ECM для работы с бизнес-процессами.

Некоторые ключевые моменты работы Activiti:

- используется нотация BPMN2, которая является более новой и популярной по сравнению с jPDL, используемой в других «движках», например jBPMN;
- поддержка работы с базой данных через стандартный интерфейс JPA (API для сохранения Java-объектов в базе данных);
- проект поддерживается большим количеством компаний-участников и постоянно развивается;
- простая интеграция со Spring-ом (универсальный фреймворк для Java-приложений);
- удобное описание процесса, позволяющее включать вызовы java-кода или использование spring-компонентов непосредственно в тексте бизнес-процесса;
- хорошо документированный набор API.

Создание бизнес-процессов происходит с помощью IDE Eclipse – свободной интегрированной среды разработки приложений. В IDE Eclipse добавляется модуль Activiti для работы с бизнес процессами.

Создание бизнес-процесса состоит из двух частей. Первая часть это графическое создание модели бизнес-процесса в визуальном редакторе. С помощью IDE Eclipse с использованием модуля Activiti. Этот модуль предлагает как готовые заготовки уже созданных бизнес-процессов для настройки и преобразования под нужды пользователя, так и большой визуальный инструментарий. Инструментарий состоит из набора коннекторов, стартовых событий, задач (отдельные этапы одного бизнес-процесса), контейнеров для процессов, логических развилок и сложных событий. Вторая часть это редактирование автоматически генерируемого .bpmn файла для разработки сложной логики бизнес-процесса.

После завершения разработки процесса на основании графического описания формируется .xml файл с моделью бизнес-процесса, описанного с помощью BPMN 2 и описанием логических переходов между задачами бизнес-процесса.

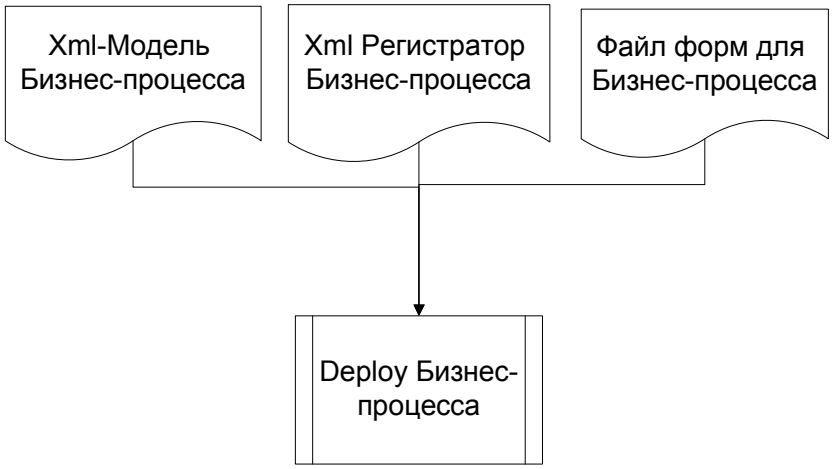


Рисунок 1. Схема размещения

На рисунке показана схема размещения созданного бизнес-процесса в Alfresco ECM. Кроме непосредственно xml-модели бизнес-процесса нужен xml-регистратор, в котором указывается тип размещения и вся информация о бизнес-процессе. Так же создается xml-файл с формами для интерфейсного отображения бизнес-процесса в Alfresco ECM. После этого все файлы помещаются в директории для бизнес-процессов внутри архива приложения Alfresco ECM, которое должно быть повторно размещено на сервере приложений, после чего производится инициализация бизнес-процесса с помощью командной строки Alfresco ECM, командой deploy.

Возможности по созданию сложного бизнес-процесса

Рассмотрим бизнес-процессы на движке Activiti вместе с интеграцией в систему документооборота Alfresco ECM. Первым инструментом для создания сложных бизнес процессов становится возможность вставки в каждую задачу JavaScript кода, который будет выполняться при создании задачи или завершении задачи. Пример:

```
<userTask id="alfrescoUsertask2" name="Составление документа"
activiti:assignee="${initiator.properties.userName}" activiti:formKey="wf:adhocTask">
  <extensionElements>
   <activiti:taskListener event="create"
        class="org.alfresco.repo.workflow.activiti.tasklistener.ScriptTaskListener">
     <activiti:field name="script">
      <activiti:string>
                var desc = bpm_workflowDescription + ":Склейка документа";
                execution.setVariable('bpm_workflowDescription', desc);
      </activiti:string>
     </activiti:field>
    </activiti:taskListener>
  </extensionElements>
</userTask>
```

JavaScript код помещается в отдельных тегах, с привязкой к событию (в данном случае create – моменту возникновения задачи). Данный инструмент позволяет использовать JavaScript API системы документооборота в процессе, а так же управлять параметрами задачи такими как: сообщение, срок выполнения, пакет документов бизнес-процесса, исполнитель и инициатор бизнес-процесса.

Вторым инструментом являются поля задачи assignee и formkey. Поле assignee предназначено для задания исполнителя задачи. В качестве аргументов могут быть заданы конкретные пользователи, группы пользователей или кандидаты – возможные исполнители, которые сами примут решение брать ли на себя задачу. Поле formkey позволяет определять и создавать для каждой задачи уникальную визуальную форму с наполнением и редактирование параметров бизнес-процесса. Модель Activiti для Eclipse позволяет использовать стандартные формы бизнес-процессов и задач системы документооборота Alfresco ECM, что упрощает работу и делает интеграцию бизнес-процессов легкой.

Рассмотрим детальные возможности поля assignee – в частности возможность задания в качестве исполнителей нескольких групп, должностных лиц. Рассмотрим на примере уже реализованной задачи:

```
<userTask id="alfrescoUsertask1" name="Исполнение"
activiti:formKey="wf:adhocTask">
    <extensionElements>
```

```
    <activiti:taskListener                                    event="create"
class="org.alfresco.repo.workflow.activiti.tasklistener.ScriptTaskListener">
          <activiti:field name="script">
                      <activiti:string>var desc = bpm_workflowDescription +
                      ":Обработка директивы";
                      execution.setVariable('bpm_workflowDescription'desc);
                  </activiti:string>
          </activiti:field>
       </activiti:taskListener>

    <activiti:taskListener                                   event="complete"
class="org.alfresco.repo.workflow.activiti.tasklistener.ScriptTaskListener">
      <activiti:field name="script">
          <activiti:string>
                  if(task.getVariableLocal('wf_reviewOutcome') == 'Approve')
                  {
                        var newApprovedCount = wf_approveCount + 1;
                        var   newApprovedPercentage   =   (newApprovedCount   /
                                                  wf_reviewerCount) * 100;

                        execution.setVariable('wf_approveCount',
                        newApprovedCount);
                        execution.setVariable('wf_actualPercent',
                        newApprovedPercentage);
                  }
                  else {
                           var newRejectCount = wf_rejectCount + 1;
                           var   newRejectPercentage   =   (newRejectCount   /
                        wf_reviewerCount) * 100;


                        execution.setVariable('wf_rejectCount',
                        newRejectCount);
                            execution.setVariable('wf_actualRejectPercent',
                        newRejectPercentage);
                         }
                     </activiti:string>
                  </activiti:field>
              </activiti:taskListener>
          </extensionElements>

          <humanPerformer>
               <resourceAssignmentExpression>
```

```
                <formalExpression>${reviewAssignee.properties.userName}
                </formalExpression>
            </resourceAssignmentExpression>
        </humanPerformer>

        <!-- For each assignee, task is created -->
        <multiInstanceLoopCharacteristics isSequential="false">
           <loopDataInputRef>bpm_assignees</loopDataInputRef>
           <inputDataItem              name="reviewAssignee"              />
           <completionCondition>${wf_actualPercent>=99}
                  </completionCondition>
        </multiInstanceLoopCharacteristics>
    </userTask>
```

В примере представлена интеграция JS-кода внутри задачи. Мы задаем расширенные параметры бизнес-процесса: процент выполнения всех копий задачи, количество копий, требуемый процент выполненных задач для логического перехода к следующей задаче. С помощью этих параметров мы организуем цикл, который создает копии задачи для каждого указанного инициатором исполнителя и осуществляет логический переход только по достижению 100% выполнения задач, то есть когда все исполнители выполнят свою копию задачи. Тем самым мы реализуем распараллеливание бизнес-процесса путем создания отдельных задач с идентичными пакетами документов для нескольких пользователей.

Далее рассмотрим процесс использования API при вставке JS-кода.

```
<activiti:taskListener                                    event="create"
class="org.alfresco.repo.workflow.activiti.tasklistener.ScriptTaskListener">
        <activiti:field name="script">
                        <activiti:string>var          desc          =
                        bpm_workflowDescription      +      ":Обработка
                        директивы";
                        execution.setVariable('bpm_workflowDescription',
                        desc);
                        var doc=bpm_package.children[0];
                        var par = doc.parent.parent;
                        par = par.createFolder("Обработка директивы");
                        var dir = par.createFolder("Части документа");
                        var    dir2    =    par.createFolder("Итоговый
                        документ");
                </activiti:string>
           </activiti:field>
      </activiti:taskListener>
```

В данном примере кода мы видим обращения как к API Activiti, так и API Alfresco ECM. Мы получаем ссылку на объект Alfresco, содержащий пакет документов и в частности на самый первый документ пакета – то есть документ, по которому был инициализированный бизнес-процесс. Далее мы получаем ссылку на библиотеку документов, в которой содержится этот документ и создаем там несколько папок для файлов, которые в будущем будут созданы в ходе бизнес-процесса.

Так же движок Activiti поддерживает такие возможности как условные переходы к следующей задаче (когда переход осуществляется только при выполнении определенного условия). На рисунке 2 показана модель бизнес-процесса с условием. Если условие выполняется, то бизнес-процесс переходит к следующей задаче, и возвращается на предыдущую задачу, если условие не выполнено.

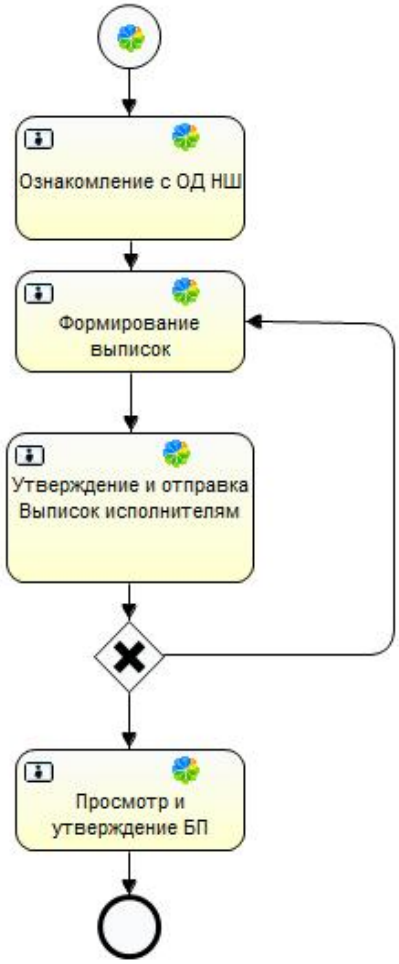


Рисунок 2. Пример модели бизнес-процесса

Еще одна возможность – это создание и запуск вложенных бизнес-процессов. Происходит это с помощью использования Alfresco ECM API.

```
<userTask id="task6" name=" " activiti:candidateGroups="GROUP_KVO"
activiti:formKey="wf:activitiReviewTask">
      <extensionElements>
        <activiti:taskListener event="create"
class="org.alfresco.repo.workflow.activiti.tasklistener.ScriptTaskListener">
          <activiti:field name="script">
            <activiti:string>
                        var doc = bpm_package.children[1];
                        var work = actions.create("start-workflow");
                        work.parameters.workflowName = "activiti$subprocess";
                        work.parameters["bpm:workflowDescription"] =
                        "Sub-process for User";
                       work.parameters["bpm:assignee"]=people.getPerson(User);
                                  work.execute(doc);

          </activiti:field>
        </activiti:taskListener>
      </extensionElements>
    </userTask>
```

Сначала с помощью API задается документ, по которому инициализируется бизнес-процесс. В данном примере это 2-ой документ пака документов бизнес-процесса, из которого производится запуск. Далее создается бизнес-процесс, устанавливается тип бизнес-процесса (subprocess), а так же устанавливается описание бизнес-процесса (Sub-process for user). В конце с помощью API Alfresco ECM находим нужного нам пользователя и назначаем его исполнителем. И в конце команда execute стартует бизнес-процесс.

Заключение

В итоге можно отметить, что средство Activiti позволяет создать бизнес-процесс, который будет описывать конкретную последовательность действий, задач на предприятии. С помощью описанных инструментов и возможностей созданы сложные бизнес-процессы, которые позволяют автоматизировать документооборот на предприятии и повысить производительность. Движок Activiti предоставляет широкие возможности для создания сложной логики бизнес-процессов и поддержки их в системах документа оборота, таких как Alfresco ECM.

Литература

1. Alfrescowiki [Электронный ресурс]. – Режим доступа: http://wiki.alfresco.com/wiki/Main_Page Загл. с экрана htm (дата обращения 17.12.2013).

2. Jeff Pots Alfresco Developer Guide Л.:Packt Publishing Ltd., Бирмингем, 2008. 556 с

3. Activiti.org [Электронный ресурс]. – Режим доступа: http://activiti.org Загл. с экрана htm (дата обращения 17.12.2013).

4. Tijs Rademakers Activiti in Action Л.:Manning Publications Co., Нью-Йорк, 2012. 456с

УДК 004.93'12

РАСПОЗНАВАНИЕ ОБРАЗОВ С ПОМОЩЬЮ БИБЛИОТЕКИ КОМПЬЮТЕРНОГО ЗРЕНИЯ OPENCV ДЛЯ РЕШЕНИЯ ЗАДАЧИ КОНТРОЛЯ СКОРОСТИ ДВИЖЕНИЯ ПО НОМЕРНЫМ ЗНАКАМ ТРАНСПОРТНЫХ СРЕДСТВ

О.С. Недильченко, Р.С. Самарев

Ключевые слова: распознавание образов, автомобильный номерной знак, OpenCV, шаблон, оптический поток, метод опорных векторов, гистограмма ориентированных градиентов.

Key words: pattern recognition, license plate, OpenCV, template, optical flow, support vector machines, histogram of oriented gradients.

Аннотация: Статья посвящена разработке и исследованию алгоритма распознавания автомобильных номерных знаков с целью контроля скорости транспортных средств и сбора статистики. В ней рассматриваются проблемы систем, одновременно контролирующих скорость транспортных средств и распознающих номерной знак. Проводится обзор и анализ возможных методов обработки изображений для поиска альтернативного способа распознавания номерных знаков транспортных средств. Предлагается алгоритм распознавания, основанный на поиске объектов по заданным шаблонам, и производится его практическое исследование с использованием библиотеки компьютерного зрения OpenCV. Делается вывод о недопустимости применения разработанного метода обособленно, но его возможном применении в качестве части другого алгоритма для повышения производительности.

Введение

Сегодня во многих больших городах остро стоит проблема регулирования дорожного движения. Увеличивающийся с каждым годом автомобильный поток

затрудняет проведение дорожно-ремонтных работ и расчет оптимальных путей перевоза грузов. А одной из наиболее существенных причин ДТП на дорогах является превышение скоростного режима, контроль которого становится всё сложнее.

Для решения данной проблемы в настоящее время применяются системы видеонаблюдения, состоящие из видеокамеры и радара, фиксирующего скорость автотранспорта [1]. При выявлении факта нарушения скоростного режима транспортным средством производится распознавание его государственного регистрационного знака.

Подобные системы видеонаблюдения легко позволяют собирать статистику о средней скорости движения автомобильного потока в определенных точках дороги, но имеют ряд недостатков при выявлении нарушителей скоростного режима.

Один из таких недостатков – некорректное присваивание транспортному средству скорости другого. Это может произойти, если автомобиль, превысивший скорость, совершил перестроение на другую полосу дорожно-транспортного полотна.

Другая проблема применения таких систем видеонаблюдения – высокая цена установки комплекса. Кроме того, некоторые водители заранее снижают скорость и снова набирают ее после проезда подобных систем, т.е. вместо организации безопасности дорожного движения происходит двукратное изменение режима движения, провоцирующего и возникновение ДТП, и возникновение дорожных заторов.

Существует и альтернативный способ решения этой задачи. Распознавание автомобильных государственных регистрационных знаков можно производить не в одной, а, как минимум, в двух точках некоторого участка дороги и на основании данных (самого номера и времени фиксации), полученных с двух или более камер, рассчитывать скорость транспортных средств.

Подобный подход позволит вычислять среднюю скорость транспортных средств на участке дороги, а значит, лучше отслеживать случаи нарушения водителями скоростного режима. Более того, цена комплекса, состоящего из нескольких камер и компьютеров, обрабатывающих данные, по предварительным оценкам является ниже стоимости сегодняшних систем видеофиксации, так как в стоимость оборудования не включается цена дорогостоящего радара для измерения скорости. Реальное применение подобного подхода ограничивается лишь необходимостью поиска оптимального способа, позволяющего получать номерные знаки транспортных средств без применения радара, т.е. только с помощью дорожной камеры и компьютера, обрабатывающего изображения.

Таким образом, целью данного исследования является разработка эффективного алгоритма распознавания автомобильных номерных знаков на изображениях, полученных с дорожной камеры.

Способы распознавания объектов на изображениях

Решая многие задачи распознавания образов, можно выбрать один из двух способов тестирования изображений на наличие требуемого объекта: сравнить тестируемое изображение с обучающими выборками или искать на изображении объект, параметры которого удовлетворяют заданным критериям.

Примером первого способа, заключающегося в составлении обучающих выборок, может служить использование двух методов: SVM и HOG [2].

SVM (support vector machines) – метод опорных векторов. Анализ изображений с помощью этого метода состоит в выделении объекта на изображении и определении, к какому классу из двух, как минимум, изначально известных, он относится. Метод SVM подразумевает также наличие заранее известных образцов каждого класса – обучающей выборки, поэтому метод SVM относится к категории методов «обучения с учителем».

В качестве объекта, который проверяется на принадлежность к классу, берется вектор в n-мерном вещественном пространстве, координаты которого описываются его отдельными атрибутами (можно рассматривать пример цвета color, который для модели RGB будет являться вектором в трехмерном пространстве: color = (red, green, blue)).

Решение о принадлежности объекта к какому-либо классу происходит с помощью разделения гиперплоскостью в многомерном пространстве векторов признаков на две группы (или более, если число известных заранее классов больше двух). Тогда классификация сводится к определению, по какую сторону гиперплоскости находится классифицируемый вектор признаков. Оптимальная разделяющая гиперплоскость - это плоскость, расстояние от которой до ближайшего вектора с любой стороны максимально. Ближайшие к плоскости вектора, «опираясь» на которые она построена и являются опорными векторами. Часто векторы признаков классов неразделимы в образованном ими пространстве. Тогда применяется нелинейное преобразование в пространство большей размерности, где эти векторы становятся уже линейно разделимыми.

Для текущей задачи в качестве двух классов можно рассматривать класс «фон» и класс «автомобиль» (или классы «фон» и «символы номера»). В качестве заранее известных и тестируемых объектов можно применять так называемые дескрипторы, полученные с помощью метода HOG (Histogram of Oriented Gradients). HOG – это метод, возвращающий гистограмму ориентированных градиентов. Он основан на разбиении изображения на определенное число зон и в подсчете в каждой зоне преобладающего направления градиентов яркости. Дескрипторы схожих изображений будут также схожи, на этом и основывается их сравнение с помощью метода SVM.

Таким образом, можно сформулировать последовательность действий при распознавании объектов (например, номерных знаков) на изображении с помощью обучающих выборок на примере методов HOG и SVM (рисунок 1).

Рисунок 1. Последовательность действий при поиске объектов на изображении с помощью обучающих выборок (использование HOG и SVM)

Недостатком данного подхода является требование большой обучающей выборки (40 000 - 60 000 изображений с возможными перспективными искажениями объектов и примерами ложного распознавания). Кроме того, сочетание методов SVM+HOG считается достаточно медленным для работы в режиме реального времени, поэтому для увеличения скорости обработки изображений часто бывает необходимо применять такие способы, как распараллеливание, введение каскадных деревьев и др.

Для применения второго способа распознавания образов, заключающегося в поиске объекта по заданным критериям, необходимо проводить ряд преобразований исходного изображения (например, уменьшение или увеличение резкости, изменение числа каналов и т.п.).

Уменьшение резкости (сглаживание) применяют с целью исключения шумов или выделения более крупных деталей на изображении. Обычно для сглаживания применяются разнообразные фильтры, позволяющие для каждого пикселя изображения вычислить новое значение на основании значений соседних пикселей. Наиболее распространенные фильтры – медианный фильтр и фильтр Гаусса.

При использовании медианного фильтра на изображение последовательно накладывается квадратная матрица с размерностью, зависящей от требуемой степени сглаживания. Каждому центральному пикселю в матрице присваивается значение среднего пикселя в отсортированном массиве, состоящем из остальных пикселей, попавших в матрицу. Данный фильтр позволяет устранять мелкие детали на изображении и выделять большие цветовые пятна. Применение гауссова фильтра основано на использовании «матрицы свертки» – матрице коэффициентов, заполненной по нормальному (гауссовому) закону. Данная матрица последовательно накладывается на изображение и перемножается со значениями покрытых ею пикселей. На основании результата умножения вычисляется значение центрального пикселя, покрытого матрицей свертки. Гауссов фильтр размывает изображение равномерно.

Изменение числа каналов изображения может заключаться в переводе цветного изображения в формат градаций серого или в бинарный формат и обратные преобразования и чаще всего используется для выделения границ объектов. Если изображение переводится в градации серого, то значения цветов всех его пикселей должны находиться в интервале от 0 до 255. Бинаризация означает перевод изображения в черно-белый формат, при котором значения пикселей становятся равными 0 (черный цвет) или 255 (белый цвет). Часто бинаризация осуществляется следующим образом: в зависимости от контраста изображения вычисляется цветовой порог, который разделяет все пиксели изображения на две части. Те пиксели, чьи цвета больше по значению, чем порог, становятся белыми, остальные – черными.

Для нахождения контуров объектов можно анализировать контрастность соседних пикселей изображения. Находя резкое изменение контраста между соседними пикселями, можно считать, что в этом месте на изображении проходит граница между двумя объектами или объектом и фоном.

При обнаружении некоторых контуров на изображении бывает полезно сравнить их с некоторым шаблоном. Для этого часто используют сравнение моментов контуров. В наиболее простом случае, момент контура вычисляется по формуле:

$$m(p,q) = \sum_{i=1}^{n} i(x,y) x^p y^q \quad , \qquad (1)$$

где p и q – порядки возведения в степень соответствующего параметра при суммировании, n – число пикселей в контуре, x, y – координаты пикселя в изображении.

У контуров одних и тех же символов моменты будут схожи. Таким образом, сравнение контуров можно свести к сравнению их моментов. Важно, что при сравнении

моментов контуров необходимо учитывать возможные углы поворота объекта на изображении, а также различия в масштабе. Поэтому, формула (1) отражает лишь общий принцип вычисления моментов контуров, тогда как на практике используются более сложные алгоритмы.

Нередко для упрощения анализа изображений (кадров), полученных с видеокамеры, пользуются оптическим потоком. Оптический поток является векторным полем видимого движения объектов, поверхностей и ребер в визуальной сцене, вызванным относительным движением между наблюдателем и сценой [3]. Для каждого кадра оптический поток любого пикселя с координатами *(x, y)* представляет собой вектор, характеризующий сдвиг пикселя относительно предыдущего кадра (рисунок 2).

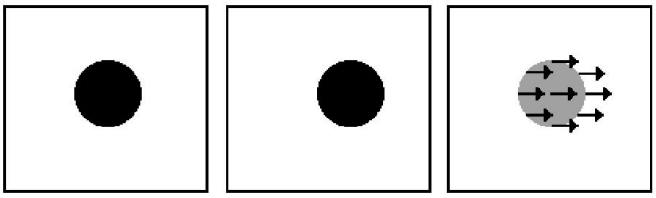


Рисунок 2. Пример оптического потока между двумя кадрами

Если камера, производящая видеосъемку, закреплена статично, то возникновение оптического потока будет обусловлено лишь движением объектов в кадре. Таким образом, для некоторых задач распознавания образов можно заранее узнать, есть ли искомый объект на изображении. Это позволит сократить вычислительные затраты на анализ изображения, т.к. необходимо будет обрабатывать лишь ту часть кадра, которая содержит оптический поток.

Выбор метода распознавания номерных знаков транспортных средств

Метод HOG + SVM требует создания больших обучающих выборок. Если использовать этот метод сначала для поиска границ автомобильного номера, а затем для распознавания символов, требуемая выборка еще больше увеличивается. Недостаточная скорость обработки изображений при работе в режиме реального времени также может стать серьезным препятствием для применения данного метода для контроля скорости движения автотранспорта.

Сложность другого метода заключается в выборе критериев поиска автомобильного номера или его знаков, а также в подборе фильтров для выделения их контуров. С другой стороны, данный метод может позволить получить высокую скорость обработки изображений, поэтому выберем его для дальнейшего исследования.

Для данного метода первым этапом распознавания номерных знаков будет являться предварительное сглаживание изображения с целью исключения шумов, бинаризация,

выделение контуров и поиск областей изображения, которые могут включать в себя автомобильный номер (рисунок 3). Учитывая особенности алгоритмов фильтрации, для выделения границ номера лучше применять медианный фильтр.

На втором этапе (рисунок 4) будет производиться распознавание символов внутри найденных на предыдущем этапе областей. К исходному изображению будет применяться гауссов фильтр, т.к. при его использовании символы номерных знаков не будут размываться больше, чем фон. Далее также будет производиться бинаризация и поиск контуров. Для идентификации символов будет применен методу сравнения моментов контуров, рассмотренный выше, с заранее подготовленными шаблонами букв и цифр.

Рисунок 3. Первый этап обработки изображения (выделение областей изображения, предположительно содержащих номерные знаки)

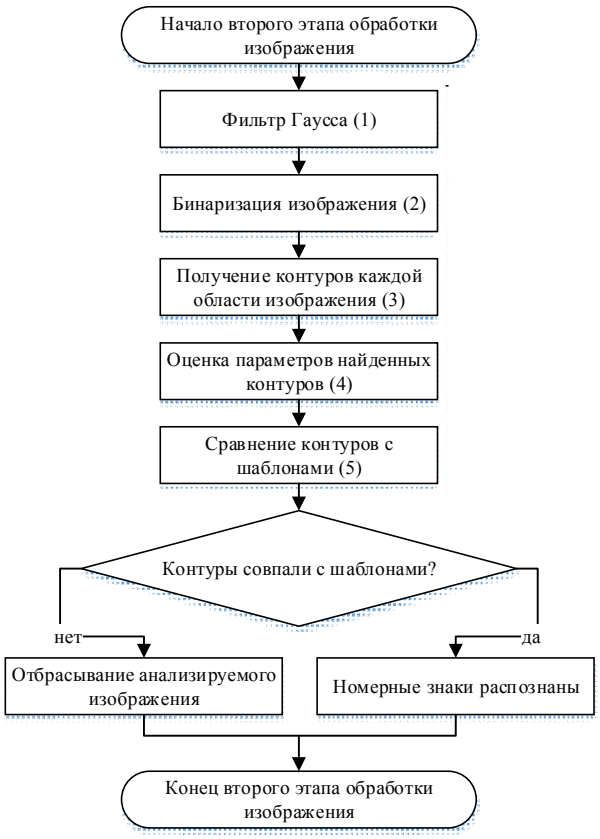


Рисунок 4. Второй этап обработки изображения (распознавание символов номерных знаков)

Реализация алгоритма с помощью библиотеки OpenCV

Для разработки алгоритма распознавания по выбранному методу поиска объектов с определенными параметрами можно воспользоваться популярной библиотекой компьютерного зрения OpenCV. Данная библиотека содержит множество функций, упрощающих решения базовых задач распознавания объектов и, кроме того, является хорошим выбором при разработке приложений для работы в режиме реального времени.

Рассмотрим функции библиотеки OpenCV, с помощью которых реализован данный метод и примеры обработки изображений.

Первый этап обработки изображения

1. Медианный фильтр (рисунок 3, блок 1)

Реализацией медианной фильтрации в OpenCV является функция medianBlur() [4]. Результат ее работы показан на рисунке 5 (анализируемое изображение) и рисунке 6 (изображение после медианного фильтра). Как видно из этих рисунков, применение медианного фильтра позволяет лучше выделить белый прямоугольник номерного знака.

Рисунок 5. Исходное изображение

Рисунок 6. Применение медианного фильтра (матрица 11 х 11)

2. Бинаризация (рисунок 3, блок 2)

Бинаризация с помощью OpenCV может быть осуществлена переводом изображения в градации серого (если входные данные являются цветными изображениями), а затем применением функции adaptiveThreshold(), пример работы которой показан на рисунке 7. Данная функция работает по принципу, описанному выше, при котором вычисляется цветовой порог, определяющий будущие значения пикселей.

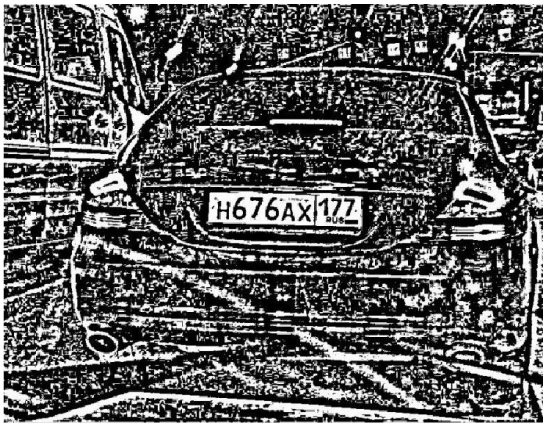


Рисунок 7. Бинаризация изображения

3. Получение всех контуров изображения (рисунок 3, блок 3)

После фильтрации и бинаризации изображения можно приступать непосредственно к поиску контуров, похожих на автомобильный регистрационный знак. Для этого в OpenCV используется функция findContours(), возвращающая все найденные контуры изображения.

4. Оценка параметров найденных контуров (рисунок 3, блок 4)

Для оценки схожести с автомобильным номером производится очерчивание каждого контура минимальным прямоугольником, что позволяет оценить пропорции

контура, а также отношение белых и черных пикселей внутри него и дать заключение, о том, может ли этот контур являться границами автомобильного номера или нет. Если контуры похожи на границы автомобильных номеров, выделяются соответствующие им области исходного изображения для дальнейшего анализа (например, рисунок 8). Если таких контуров не найдено – изображение отбрасывается. На этом завершается первый этап обработки изображения.

Рисунок 8. Найденная область изображения, предположительно содержащая автомобильный номер

Второй этап обработки изображения

1. Фильтр Гаусса (рисунок 4, блок 1)

Реализацией гауссовой фильтрации в OpenCV является функция GaussianBlur(). Результат ее работы показан на рисунке 9 (анализируемое изображение) и рисунке 10 (изображение после фильтра Гаусса).

Рисунок 9. Исходное изображение

Рисунок 10. Применение фильтра Гаусса (матрица 11 x 11)

2. Бинаризация изображения (рисунок 4, блок 2)

Производится так же, как и в п.2 первого этапа обработки изображения.

3. Получение контуров каждой области изображения (рисунок 4, блок 3). Аналогично поиску контуров границ в п.3 первого этапа обработки изображения производится поиск контуров символов внутри найденных на предыдущем этапе областей.

4. Оценка параметров найденных контуров (рисунок 4, блок 4)

Оценка производится по принципу, описанному в п.4 первого этапа обработки изображения. Пример результата поиска контуров символов показан на рисунке 11.

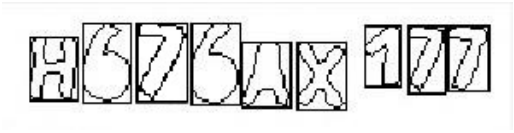


Рисунок 11. Найденные контуры символов автомобильного регистрационного знака

5. Сравнение контуров с шаблонами (рисунок 4, блок 5)

Если на предыдущей стадии для области было найдено требуемое количество контуров, похожих на символы автомобильного регистрационного знака, начинается их сравнение с контурами заранее подготовленных шаблонов, иначе автомобильный номер считается не обнаруженным на данном изображении. Сравнение найденных символов с шаблонами осуществляется с помощью вычисления моментов контуров. Функция OpenCV, реализовывающая сравнение контуров по их моментам называется matchShapes(). Если сравнение прошло успешно, номер считается распознанным. Если нет, производится анализ других областей.

После завершения этой стадии или выдаются номерные знаки, находящиеся на этом изображении, или изображение отбрасывается, если номерных знаков не обнаружено. На этом завершается второй этап обработки изображения.

Результаты работы разработанного алгоритма

Условия проведения эксперимента:

- освещенность во время съемки: дневной свет (пасмурный и солнечный день, 500-1000 и 10000 лк соответственно),

- чувствительность камеры (ISO): от 100 до 400, авто-настройка,

- форматы полученных кадров: 800 x 600, 1280 x 760, 3800 x 2600,

- количество анализируемых кадров: 100 кадров, содержащих изображения автомобильного номера, 50 кадров, не содержащих изображения автомобильного номера, для каждого формата.

Результаты эксперимента (таблицы 1, 2, 3).

Вероятности обнаружения границ номерного знака Таблица 1

Формат кадра	800 x 600	1280 x 760	3800 x 2600
Вероятность обнаружения границ номерного знака	90%	85%	60%

Вероятности правильного распознавания символа номерного знака Таблица 2

Формат кадра	800 x 600	1280 x 760	3800 x 2600
Вероятность распознавания символа автомобильного номера	40%	50-60%	60-70%

Время обработки одного кадра (Intel Core i7, ОЗУ 4Гб, x64) Таблица 3

Формат кадра	800 x 600	1280 x 760	3800 x 2600
Время обработки кадра, мс	5-10	10-15	35-45

Как видно по полученным результатам, при оптимальном разрешении камеры (1280 x 760) данный метод позволяет с достаточно большой вероятностью обнаружить границы автомобильного номера (85%), но не позволяет с высокой точностью распознавать символы номерных знаков (50-60%).

Заключение

Предложенный алгоритм распознавания номерных знаков транспортных средств, основанный на анализе признаков контуров изображений и сравнении с шаблонами, не дает требуемой точности распознавания государственных регистрационных знаков. Он позволяет с вероятностью около 85% распознавать границы номерных знаков на изображениях и с вероятностью 50-60% – символы номерных знаков и неприменим в качестве самостоятельного алгоритма распознавания.

Тем не менее, предложенный алгоритм может быть полезным для быстрого поиска границ автомобильного номера в целях сокращения времени обработки другим методом распознавания номерных знаков или как экспресс-метод, позволяющий обеспечить быструю обработку при ограниченных ресурсах вычислительных средств.

В дальнейшем возможно усовершенствование разработанного алгоритма путем анализа оптического потока видеозаписи, способного улучшить процесс выделения зон, содержащих государственные регистрационные знаки.

Также возможно проведение новых исследований с целью создания высокопроизводительного метода распознавания номерных знаков, сочетающего в себе реализованный алгоритм для выделения границ и описанный выше метод HOG+SVM.

Литература

1. Пирогов А. Автомобиль – безопасное средство передвижения. Детектирование нарушений на российских дорогах. – URL: http://www.secuteck.ru/articles2/dvr/avtomobil--bezopasnoe-sredstvo-peredvizheniya.-detektirovanie-narusheniy-na-rossiyskih-dorogah/ (дата обращения: 02.12.2013).

2. Christopher M. Bishop. Pattern Recognition and Machine Learning (Information Science and Statistics). Springer, 1st ed. 2006. corr. 2nd printing edition, 2007. – 708 p.

3. R. Laganire. OpenCV 2 Computer Vision Application Programming Cookbook. – Packt Publishing, 2011. - 304 p.

4. G. Bradski, A. Kaehler. Learning OpenCV. Computer Vision with the OpenCV Library. – O'Reilly Media, 2008. - 571 p.

УДК 81'367

НЕКОТОРЫЕ ПРОБЛЕМЫ СИНТАКСИЧЕСКОГО АНАЛИЗА ПРЕДЛОЖЕНИЙ НА РУССКОМ ЯЗЫКЕ

М.Ф. Пантелеев, Р.С. Самарев

Ключевые слова: *Синтаксическое дерево, синтаксический анализатор, разбор предложения, синтаксические неоднозначности.*

Key words: *Syntax tree, syntax analyzer, sentence parsing, syntax ambiguities.*

Аннотация: *В статье рассмотрены основные причины возникновения неоднозачностей при распознавании предложений на русском языке и дана их классификация. Показаны возможные направления борьбы с этими неоднозначностями. Приведены примеры разбора «проблемных предложений» следующими системами, включающими в себя синтаксический анализатор: AOT, Treevial.*

Введение

В настоящее время существует очень важная проблема извлечения информации из текста естественном языке или его классификации. Автоматизация этого процесса представляется очень важной задачей в основном потому что каждый час в СМИ появляется огромное количество информации, которое человеку обработать просто не под силу. Один из возможных и наиболее действенных способов извлечения информации из текста базируется на использовании синтаксических деревьев предложений.

Синтаксический анализ

В лингвистике и информатике, синтаксический анализ – это процесс сопоставления линейной последовательности лексем (слов, токенов) естественного или формального языков с его формальной грамматикой. Результатом обычно является дерево разбора

(синтаксическое дерево). Обычно применяется совместно с лексическим анализом.

Для русского языка затруднительно провести синтаксический разбор предложения без установления морфологических характеристик слов. Поэтому необходимым условием правильности работы синтаксического анализатора является правильный результат работы морфологического анализатора. Простейшая схема получения дерева разбора предложения показана на рисунке 1.

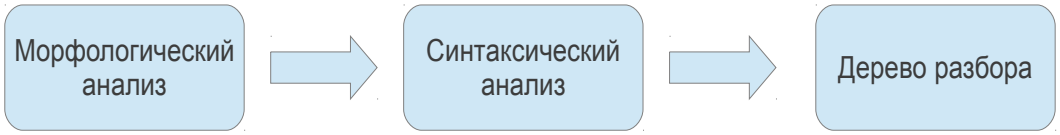


Рисунок 1. Простейшая схема получения дерева разбора предложения

На сегодняшний день морфологические анализаторы, использующие словари, обеспечивают очень высокую точность – около 100%. Существенно хуже обстоит дело со словами, отсутствующими в словарях, то есть новыми или несуществующими словами (например профессиональный жаргон) – для таких слов порождаются гипотезы и точность разбора может варьироваться достаточно сильно. Из существующих на сегодняшний день словарных морфологических анализаторов стоит выделить следующие:

1) mystem [1];

2) solarix [2];

3) treeton [3];

4) морф. анализатор системы АОТ [4].

Они используют словари (один из лучших – грамматический словарь А.А. Зализняка), базы данных, сформированные силами открытых или закрытых корпусов, которые занимаются морфологической и синтаксической разметкой (opencorpora, СинТагРус). Морфологические анализаторы на базе правил (стеммеры)[6] существенно проще, но точность анализа ниже.

Проблемы синтаксических анализаторов

Актуальными на сегодняшний день являются проблемы синтаксических анализаторов. Самая главная из них – проблема синтаксической неоднозначности. Исследования показали, что явления, вызывающие неоднозначность, зависят, как правило, от значения входящих в предложение слов и от их способности сочетаться. Можно выделить три подгруппы причин, способствующих появлению синтаксической неоднозначности:

1. Неоднозначность определения входящих в предложение единиц;

2. Неоднозначность определения связей между единицами, упомянутыми в п.1;

3. Особенности самой синтаксической структуры.

Рассмотрим каждую из этих подгрупп по отдельности. При этом примеры разбора проблемных предложений будут показаны по результатам работы средствам синтаксических анализаторов. В качестве примера будет приводиться дерево предложения, построенное по синтаксическому разбору. Использованы анализатор системы АОТ, как один из наиболее известных свободного распространяемых анализаторов, а также анализатор Treevial, разрабатываемый на факультете ВМиК МГУ им. Ломоносова.

Неоднозначность определения входящих в предложение единиц

Сюда можно отнести следующие явления:

1. Омонимия и полисемия;

2. Грамматическая конверсия;

3. Неоднозначность интерпретации проформы.

Омонимия и полисемия. Омонимы – разные по значению, но одинаковые по звучанию и написанию слова, морфемы и другие единицы языка. Пример: «Звериная клетка»/«клетка в тетради», «Косил косой косой косой». Лингвисты не приводят однозначного различия между полисемией и омонимией, но можно считать, что омонимия – случайное совпадение слов, в то время, как полисемия – наличие у слова разных исторических связанных значений. Например, «бор» в значении «сосновый лес» и «бор» в значении «химический элемент» являются омонимами, в то время, как «эфир» в смысле «радиовещание и телевидение» и «эфир» в смысле органического вещества являются полисемией, так как оба происходят от древнегреческого слова αἰθήρ , которое в переводе означает «горный воздух».

В качестве примера возьмём предложение «Косил косой косой косой». Видно, что существует несколько вариантов интерпретации предложения в зависимости от положения в нём главного действующего лица – косого. Результаты показаны на рисунках 2,3.

Можно сказать, что синтаксический анализатор АОТ плохо справился с разбором данного предложения, так ему не удалось правильно определить главное действующее лицо. Также он посчитал, что два последних слова являются прилагательными, которые характеризуют один предмет – «коса». Анализатор Treevial предоставил несколько вариантов разбора, в числе которых был верный.

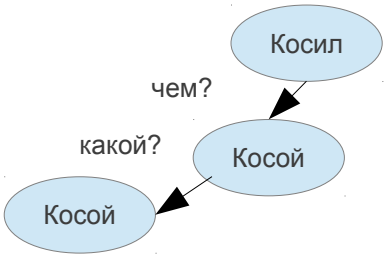


Рисунок 2. Пример разбора анализатором АОТ

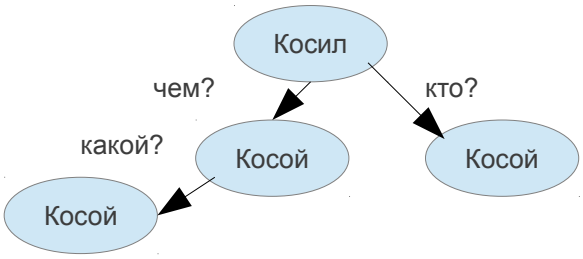


Рисунок 3. Пример разбора анализатором Treevial

Грамматическая конверсия. Грамматическая конверсия – это выражение одного и того же действия в разных направлениях.

В качестве примера возьмём, разные варианты интерпретации фразы «Раскапывайте погребённых в земле слепых исполинов». Ожидается один из следующих вариантов интерпретации предложения анализатором: «[Раскапывайте погребённых] [в земле слепых исполинов]» или «[Раскапывайте погребённых в земле] [слепых исполинов]». Результаты разбора показаны на рисунках. 4,5.

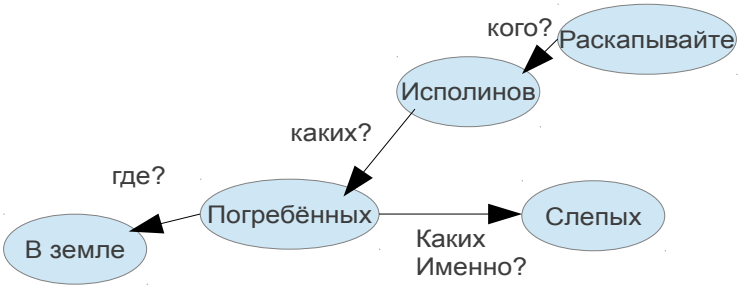


Рисунок. 4. Пример разбора анализатором АОТ

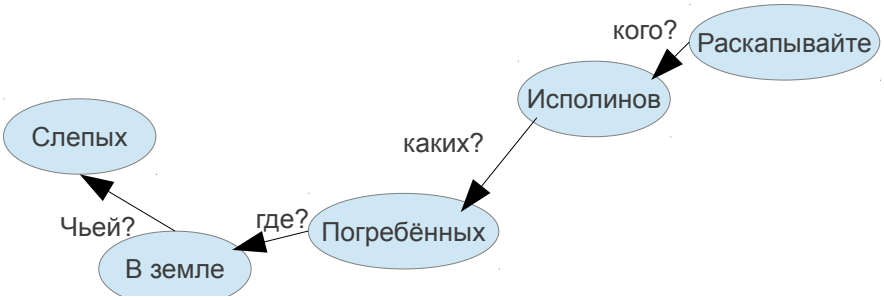


Рисунок. 5. Пример разбора анализатором Treevial

Синтаксический анализатор АОТ правильно разобрал предложение, отдав

предпочтение второму варианту разбора.

Синтаксический анализатор системы Treevial предложил несколько вариантов разбора предложения, в том числе предложенный анализатором АОТ, как минимум два были верны.

Неоднозначность интерпретации проформы

В качестве примера можно взять предложение “Доклад ученого, о котором я вам говорил”. Здесь неопределенность заключается в том, что непонятно, о ком говорит действующее лицо - о докладе или об учёном. Результат разбора приведён на рисунке 6.

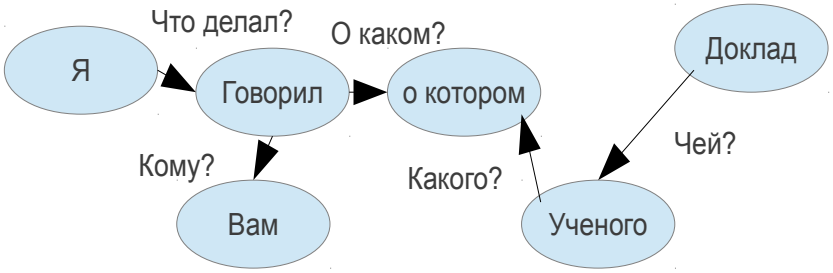


Рисунок 6. Пример разбора анализатором АОТ

Синтаксический анализатор АОТ правильно разобрал предложение, посчитав, что ранее говорилось об учёном.

Синтаксический анализатор системы Treevial предложил один вариант разбора, отличающийся только тем, что проформой был определён доклад.

Неоднозначность определения синтаксических связей между этими единицами

Сюда можно отнести следующие явления:

1. Факультативность сирконстантов.
2. Вариативность актантов.
3. Факультативность актантов.

Факультативность сирконстантов. Сирконстант – слово, отражающее, при каких обстоятельствах происходит действие. Например: «вчерашнее пятно», «завтрашний день». Сирконстанты не являются обязательными элементами и имеют способность легко находить в предложении потенциальных «хозяев».

В качестве примера можно рассмотреть предложение «Новость о преступлении в полицейском участке». Неоднозначность заключается в том, что непонятно, произошло ли преступление непосредственно в полицейском участке или о нем там узнали.

Результаты разборов приведены на рисунках 7, 8, 9:

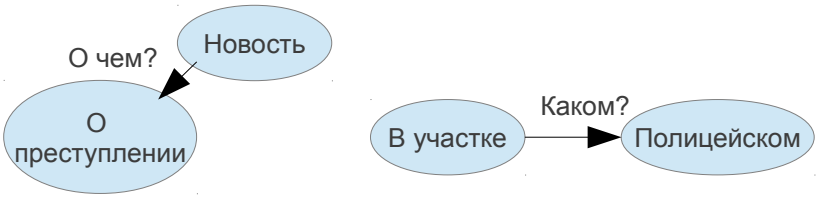


Рисунок 7. Пример разбора анализатором АОТ

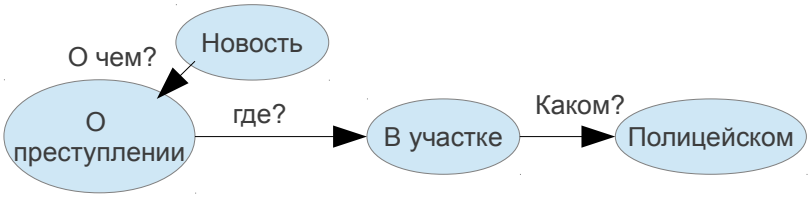


Рисунок 8. Пример разбора анализатором Treevial

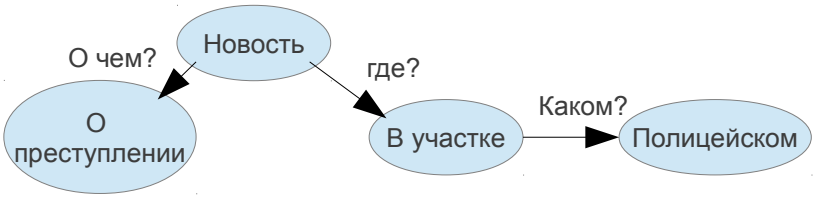


Рисунок 9. Пример разбора анализатором Treevial

Синтаксический анализатор АОТ не смог полностью правильно построить дерево, так как в первом варианте интерпретации не хватает отношения «преступление (где?) в участке», а во втором – отношения «новость (где?) в участке».

Синтаксический анализатор системы Treevial предложил несколько вариантов разбора, в том числе два ожидаемых и полностью верных.

Вариативность актантов. Актант – активный, действующий участник ситуации. Вариативность актантов означает способность слова заполнять валентности актантами разных типов, а также подчинять себе разные формы.

В качестве примера рассмотрим предложение «Имеется возможность просьбы начальника избежать». Неоднозначность заключается в том, что неизвестно, имеется ли возможность избежать просьбы начальника или имеется возможность не просить начальника о чём-то. На самом деле, существует больше интерпретаций, ограничимся этими двумя. Результаты разборов показаны на рисунках 10,11.

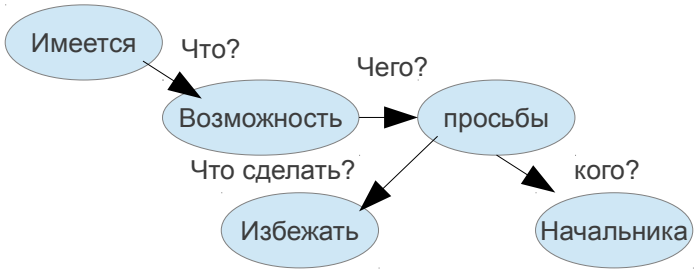


Рисунок 10. Пример разбора анализатором АОТ

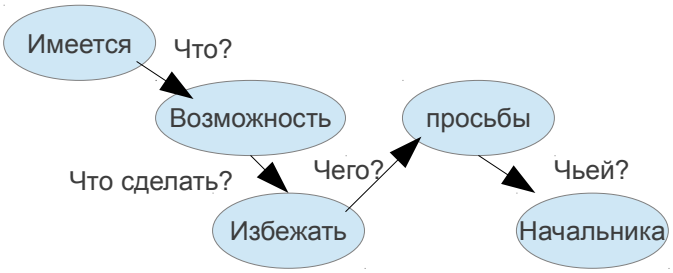


Рисунок 11. Пример разбора анализатором Treevial

Синтаксический анализатор АОТ продемонстрировал построение графу, близкое к ожидаемому при использовании второй интерпретации. Для полноты не хватает отношения «возможность (что сделать?) избежать».

Синтаксический анализатор системы Treevial продемонстрировал верное построение дерева, совпадающее с перым вариантом.

Факультативность актантов, то есть возможность относить зависимый элемент как к одному, так и к другому хозяину.

Рассмотрим следующий пример: «Учитель пения не слышит». Неоднозначность заключается в следующем: Учитель по пению не слышит что-либо или учитель не слышит пения. Результат раз приведён на рисунок 12.

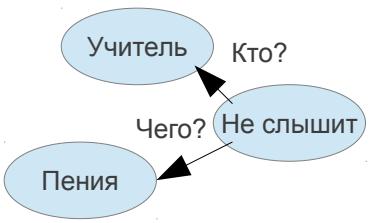


Рисунок 12. Пример разбора анализатором АОТ

Видно, что синтаксический анализатор АОТ правильно произвёл разбор предложения, который соответствует второму варианту интерпретации.

Синтаксический анализатор системы Treevial не предоставил ни одного варианта построения дерева предложения.

Неоднозначности, вызванные особенностью синтаксической структуры

Сюда можно отнести следующие явления:

1. Наличие зависимого элемента, который может быть отнесён к одному или нескольким членам предложения.

2. Наличие элемента, который может состоять в отношении с несколькими однородными членами предложения или только с одним из них.

3. Случаи перераспределния однородности.

Здесь неоднозначность связана со свойствами самой синтаксической структуры.

Наличие зависимого элемента, который может быть отнесён к одному или нескольким членам предложения

Рассмотрим в качестве примера предложение «Век атомной энергетики, автоматизации». Здесь неоднозначность заключается в следующей неопределенности: характеризует ли прилагательное «атомной» только существительное «энергетики», или еще и существительное «автоматизации»?.

Результат разбора показан на рисунке 13.

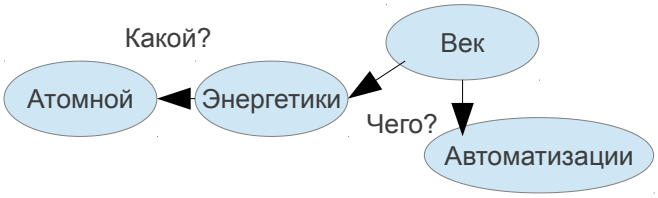


Рисунок 13. Пример разбора анализаторами АОТ, Treevial

Видно, что синтаксический анализатор АОТ правильно произвёл разбор предложения, который соответствует первой интерпретации.

Синтаксический анализатор системы Treevial произвел разбор идентично.

Наличие элемента, который может состоять в отношении с несколькими однородными членами предложения или только с одним из них

Рассмотрим предложение «Доверять дочери и отцу запрещали». Здесь союз «и» может выступать как в роли соединительной – доверять запрещали (дочери и отцу), так и в роли усилительной конструкции – доверять дочери запрещали даже отцу. Результаты приведены на рисунках. 14,15.

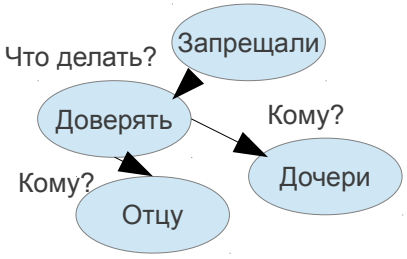


Рисунок. 14. Пример разбора анализатором АОТ

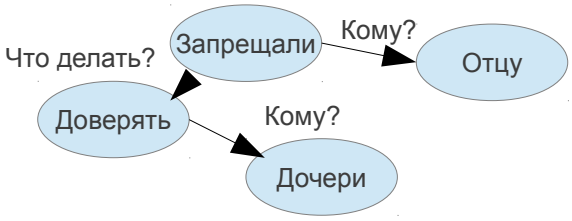


Рисунок 15. Пример разбора анализатором Treevial

Синтаксический анализатор АОТ правильно произвёл разбор предложения, посчитав, что союз «и» соединяет однородные члены.

Синтаксический анализатор Treevial правильно произвёл разбор предложение, отдав предпочтение второму варианту, однако этот вариант разбора имел далеко не самый высокий приоритет среди предложенных.

Случаи перераспределния однородности

Можно рассмотреть следующий пример: «Мой брат или Петя и Миша останутся». Союзы «и» и «или» могут по-разному объединять однородные члены предложения. Так, в одном случае останутся брат или Петя с Мишей, а в другом – Миша – либо с братом, либо с Петей. Результат разбора приведен на рисунке 16:

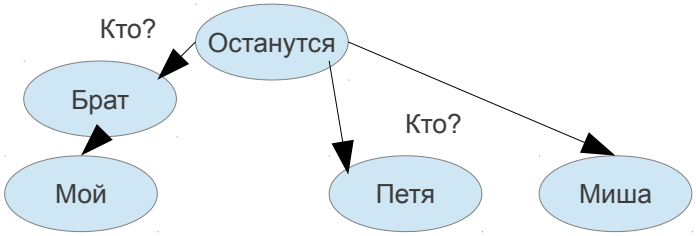


Рисунок 16. Пример разбора анализатором АОТ

Можно заметить, что средствам АОТ не удалось удовлетворительно разобрать приведённый пример, потому он не подходит ни под один из возможных вариантов разбора.

Синтаксический анализатор Treevial также не смог предложить верный вариант разбора.

Заключение

Проведенные исследования показали, что АОТ смог предоставить лишь один не всегда правильный вариант разбора, а Treevial практически всегда предлагал несколько вариантов построения синтаксических деревьев, но эти правильные варианты, как правило, приходилось отыскивать среди множества неправильных. Так, например, для

предложения «Косил косой косой косой» было порождено более 100 вариантов разбора. Можно отметить, что использованные примеры продемонстрировали следующее: при проектировании синтаксического анализатора необходимо учитывать, что для правильного построения дерева разбора предложения недостаточно просто использовать правила, описывающие отношения между словами, нужно брать также в расчёт результаты семантического анализа – это сильно повысит вероятность верно произвести разбор. Если некоторые причины, ведущие к синтаксической неоднозначности в большинстве случаев можно снять (напр. омонимию), то при разборе некоторых предложений снять неоднозначность без контекста очень трудно или вообще невозможно. Разумно будет сказать, что чем сложнее использовать контекст, тем лучше должен быть спроектирован семантический анализатор, чтобы он мог справиться с этой задачей. Сегодня решена только задача поверхностного семантического анализа: например, средства АОТ производят поверхностный семантический анализ только для одного разбираемого предложения. Однако может возникнуть такая ситуация, когда для снятия неоднозначности в каком-либо предложении может потребоваться информация, содержащаяся в другом. Нетрудно привести такие примеры, в которых использование контекста невозможно и из нескольких вариантов разбора надо выбрать наиболее правдоподобный, то есть подходящий по смыслу. Научить машину “понимать” образы является ключевой проблемой при создании уже искусственного интеллекта.

В завершение обзора целесообразно привести таблицу с результатами разбора.

Таблица 1

Явление / Пример предложения	**АОТ**	**Treevial**
Омонимия/ Косил косой косой косой	Неправильно	Правильно
Грамматическая конверсия/ Раскапывайте погребённых в земле слепых исполинов	Один правильный вариант разбора	Два правильных варианта разбора
Неоднозначность интерпретации проформы/ Доклад ученого, о котором я вам говорил.	Правильно	Правильно
Факультативность сирконстантов/ Новость о преступлении в полицейском участке	Один не полностью правильный вариант разбора	Два правильных варианта разбора
Вариативность актантов/ Имеется возможность просьбы начальника избежать	Один не полностью правильный вариант разбора	Один правильный
Факультативность актантов/ Учитель пения не слышит	Правильно	Не было предложено вариантов разбора
Наличие зависимого элемента, который может	Правильно	Правильно

Явление / Пример предложения	**АОТ**	**Treevial**
быть отнесён к одному или нескольким членам предложения/ Век атомной энергетики, автоматизации		
Наличие элемента, который может состоять в отношении с несколькими однородными членами предложения или только с одним из них/ Доверять дочери и отцу запрещали	Правильно	Правильно
Случаи перераспределения однородности/ Мой брат или Петя и Миша останутся	Неправильно	Неправильно

В настоящее время ведутся активные разработки, способствующие снятию неоднозначности в том числе при помощи использования семантического анализа. Обе приведённые системы использовали в ходе построения дерева семантические анализаторы, но не всегда результаты были верными. Это можно объяснить тем, что сама по себе задача написания семантического анализатора является еще более сложной, чем задача написания синтаксического. На сегодняшний день не был выявлен алгоритм, позволяющий однозначно интерпретировать предложение и вряд ли его можно будет найти вообще ввиду особенностей грамматики языка.

Литература

1. Программа mystem [Электронный ресурс]. - URL: http://company.yandex.ru/technologies/mystem/

2. Программа solarix [Электронный ресурс]. - URL: http://www.solarix.ru/

3. Программа Treeton [Электронный ресурс]. - URL: http://starling.rinet.ru/treeton/

4. Программа АОТ [Электронный ресурс]. - URL: http://www.aot.ru/

5. Митренина О.В., Проблемы неоднозачности синтаксического анализа: Дис. канд.фил.наук. Санкт-Петербург 2005. 133с.

6. Стеммер Snowball [Электронный ресурс]. - URL: http://snowball.tartarus.org/

УДК 004.72

ТЕНДЕНЦИИ РАЗВИТИЯ ОБЛАЧНЫХ ТЕХНОЛОГИЙ НА РОССИЙСКОМ РЫНКЕ

А.С. Паус, О.А. Целовальникова, О.Ю. Ерёмин

Ключевые слова: облачные вычисления, GRID вычисления, виртуализация, облачные вычисления в российском бизнесе

Keywords: cloud computing, GRID computing, virtualization, cloud computing in Russian business

Аннотация: В данной работе рассматриваются основные тенденции развития рынка облачных технологий в России, включая решения для домашнего и корпоративного пользования. Цель работы: выявить наиболее распространенные проблемы, с которыми сталкиваются пользователи при переводе IT-инфраструктуры на взаимодействие с облачными сервисами и ресурсами. Одним из методов, примененных в исследовании, является сравнение российских моделей облачных вычислений с зарубежными, а также анализ данных, полученных в результате сторонних социологических исследований.

Введение

В последние годы на рынке ИТ-индустрии начало стремительно развиваться новое направление – так называемые облачные технологии. Это стало следствием всеобщей тенденции к размыванию границ корпоративной среды, что означает:

1. Удаленный мобильный доступ к корпоративным данным и приложениям;
2. Разнообразие устройств доступа (смартфон, ноутбук;
3. Перемещение данных на удаленные сервера;
4. Разнообразие технологий подключения (Wi-Fi, GSM/GPRS/EDGE/LTE) (см. рисунок 1).

Облачные технологии – это модель доступа к общему пулу конфигурируемых вычислительных ресурсов, которые могут быть предоставлены оперативно и освобождены с минимальными эксплуатационными затратами. Подразумевается, что этот пул доступен повсеместно, а, следовательно, удобен в использовании.

Сам по себе удаленный доступ – не такая уж и новость, элементарный пример – использование почты в окне браузера. Однако историю облачных технологий как решения для бизнеса можно начать с 2006 года, когда компания Амазон представила свою инфраструктуру веб-сервисов, не только обеспечивающую хостинг, но и предоставляющую клиенту удаленные вычислительные мощности.

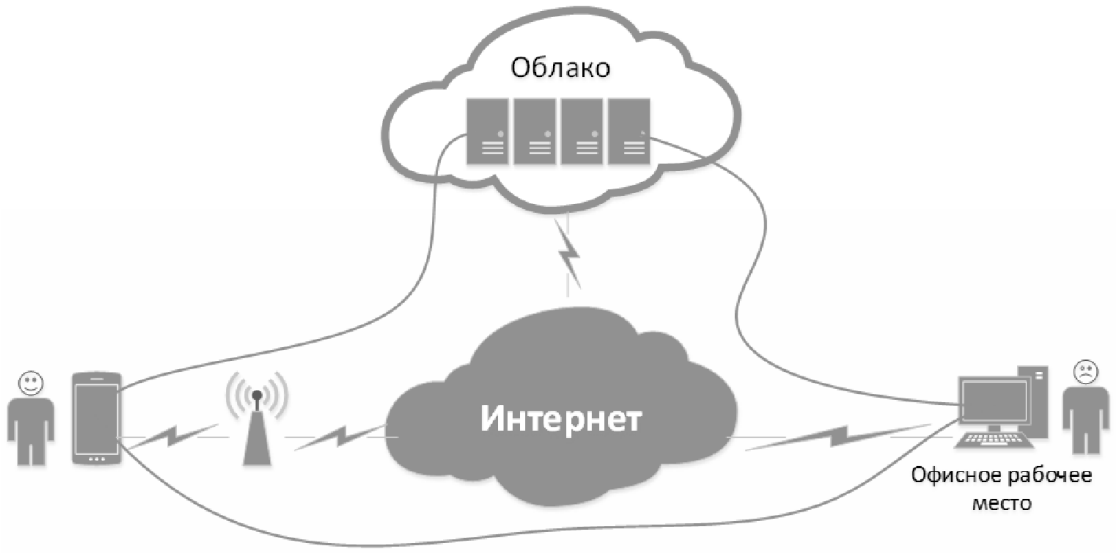


Рисунок 1. Структура системы, построенной с использованием облачных технологий

Отличие облачных вычислений от виртуализации и grid-вычислений. С точки зрения пользователя нет никакой разницы между работой в облаке и работой с приложением, установленным на локальном сервере. Принципиально же отличает «облако» от классических серверов тот факт, что свои ресурсы оно использует как глобальный виртуальный компьютер, где приложения работают независимо от каждого конкретного компьютера и его конфигурации. При этом стоит заметить, что в отличие от технологии GRID, где вычислительные мощности объединяются для обеспечения выполнения одной задачи, в облаке на одном и том же оборудовании могут параллельно работать несколько приложений, совместно используя динамически выделяемые ресурсы (см. рисунок 2). Такой способ организации вычислительного процесса несколько напоминает виртуализацию, однако является более гибким и мощным решением. Функционирование приложения не зависит от конфигурации каждого конкретного устройства. В зависимости от нужд пользователя, в работу могут быть вовлечены дополнительные ресурсы, и наоборот, неиспользуемые вычислительные мощности всегда могут быть освобождены.

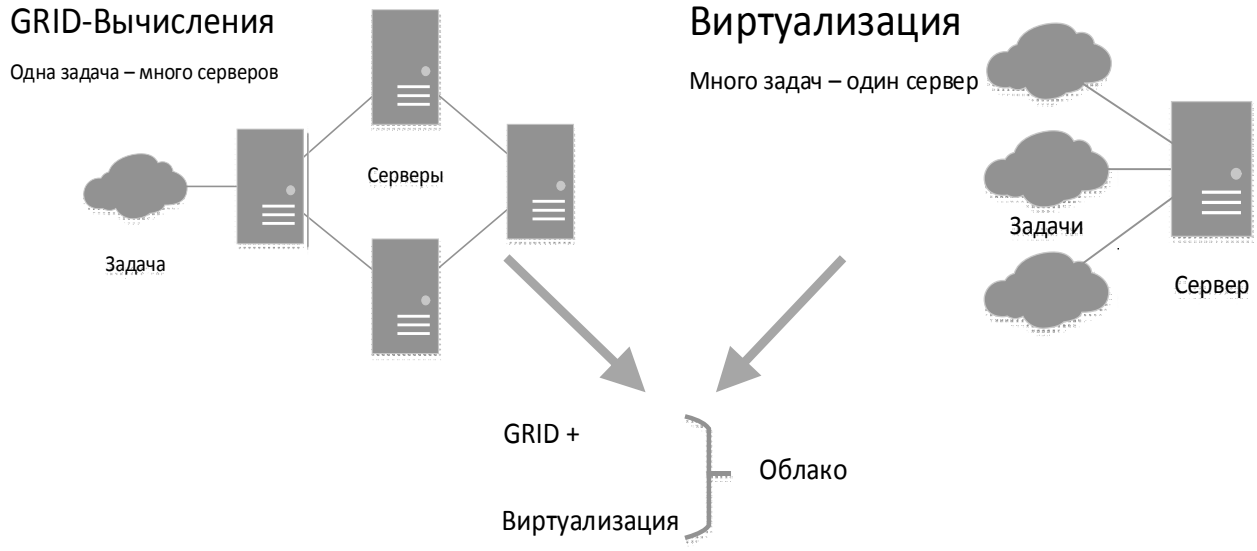


Рисунок 2. Сравнение «облака» с GRID-вычислениями и виртуализацией

Модели реализации облачных технологий. С ростом интереса к переносу части задач предприятия на внешние вычислительные мощности, перед компаниями-провайдерами встала задача, в каком виде можно продавать решения, базирующиеся на использовании облачных технологий. Со временем сформировалось три основных модели обслуживания: Saas (ПО как услуга), Paas (платформа как услуга) и Iaas (инфраструктура как услуга), которые дополняют друг друга и занимают разные ниши рынка (см. таблица 1).

- **Инфраструктура как услуга (IaaS, Infrastructure as a Service)**. Потребителю предоставляются средства обработки данных, хранения, сетей и других базовых вычислительных ресурсов, на которых потребитель может развертывать и выполнять произвольное программное обеспечение, включая операционные системы и приложении. Например, распространенные заграницей сервера Amazon.
- **Платформа как услуга (PaaS, Platform as a Service).** Потребителю предоставляются средства для развертывания на облачной инфраструктуре создаваемых потребителем или приобретаемых приложений, разрабатываемых с использованием поддерживаемых провайдером инструментов и языков программирования. Типичным примером может служить хостинг сайтов.
- **Программное обеспечение как услуга (SaaS, Software as a Servise).** Потребителю предоставляются программные средства – приложения провайдера, выполняемые на облачной инфраструктуре. Одним из наиболее распространенных примеров – электронная почта google.

Модели облачных услуг Таблица 1

Модель облачной услуги	Краткое описание модели	Предназначение модели, существующие реализации
IaaS	Эластичная среда разнородных ресурсов: серверных, сетевых, ресурсов хранения.	Модель позволяет гибко и на ходу переконфигурировать платформы. Реализованный пример – облачный сервис компании Amazon
PaaS	Интерфейс управления IaaS из приложений	Модель позволяет управлять облако из прикладных систем. Реализованный пример – сервис Google drive
SaaS	Модель продажи ПО как услуги из внешнего IaaS-облака	Модель позволяет сократить расходы на внедрение и сопровождение ПО. Реализованный пример – сервис Google docs

Также облака можно классифицировать на частные и публичные, а также внешние и внутренние (см.таблицу 2).

Классификация облаков Таблица 2

	Частные	Публичные
Внутренние	Собственный датацентр	Свое облако, излишки которого можно продать
Внешние	Облако для себя на внешней платформе	Облако на продажу

Три модели облачных услуг занимают разные ниши рынка, и в общем случае их можно представить иерархично (см. рисунок 3). На серверное «железо», т.е. системы хранения данных, обработки информации, связи, устанавливается среда, управляющая этими ресурсами (IaaS). На эту среду устанавливается некий интерфейс управления ресурсами сервера (PaaS), с помощью которого на облачной инфраструктуре можно развертывать определенные приложения для пользования ими через сеть (SaaS).

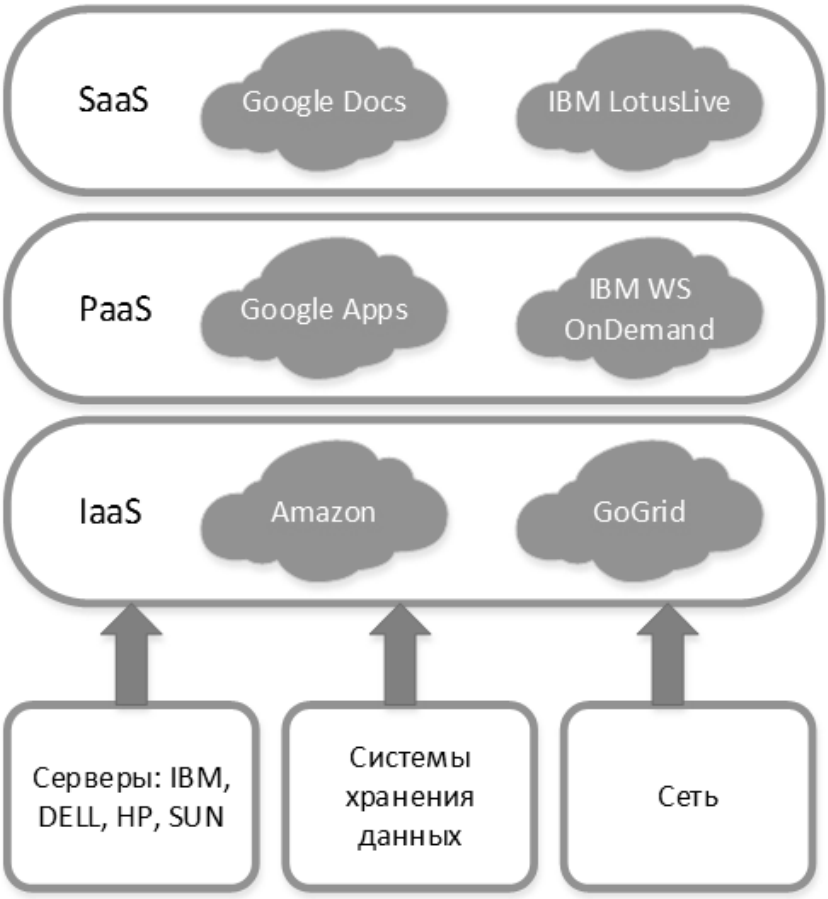


Рисунок 3. Иерархия «облачных» продуктов

Преимущества облачных сервисов

Облачные сервисы имеют ряд преимуществ по сравнению с обычными серверами, установленными в компаниях:

Доступность. Информация в облаке имеет высокую доступность, около 99,999, которую могут поддерживать только крупные компании-провайдеры. Минимальное время простоя сложно гарантировать собственными силами. Доступ к информации, хранящейся на облаке, может получить каждый, кто имеет компьютер, планшет, любое мобильное устройство, подключенное к сети интернет. Из этого вытекает следующее преимущество.

Мобильность. У пользователя нет постоянной привязанности к одному рабочему месту. Из любой точки мира менеджеры могут получать отчетность, а руководители – следить за производством.

Экономичность. Одним из важных преимуществ называют уменьшенную затратность. Пользователю не надо покупать дорогостоящие, большие по вычислительной мощности компьютеры и ПО, а также он освобождается от необходимости нанимать специалиста по обслуживанию локальных IT-технологий.

Возможность брать вычислительные мощности в аренду. Пользователь получает необходимый пакет услуг только в тот момент, когда он ему нужен, и платит, собственно, только за количество приобретенных функций.

Гибкость. Все необходимые ресурсы предоставляются провайдером автоматически.

Высокая технологичность. Большие вычислительные мощности, которые предоставляются в распоряжение пользователя, которые можно использовать для хранения, анализа и обработки данных.

Надежность. Некоторые эксперты [3] утверждают, что надежность, которую обеспечивают современные облачные вычисления, гораздо выше, чем надежность локальных ресурсов, аргументируя это тем, что мало предприятий могут себе позволить приобрести и содержать полноценный ЦОД.

Реальная ситуация на Российском рынке

По данным исследований, проведенных компанией Parallels, в рейтинге стран по внедрению облачных сервисов в бизнес Россия занимала на 2009 год скромное 34 место с долей рынка 0,03% (см. рисунок 4). Прогнозы были неутешительные.

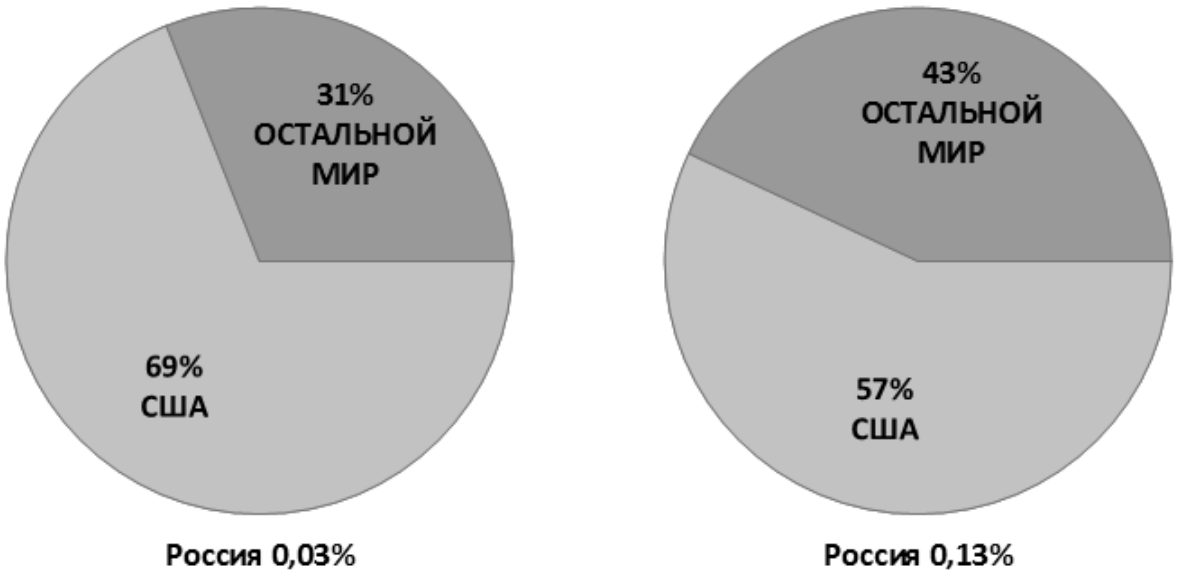


Рисунок 4. Внедрение облачных сервисов в бизнес

Причины неразвитости облачных технологий в России:

- **Отсутствие безопасности**. Не во всех точках нашей страны хорошо обстоят дела с поддержкой мобильных каналов связи. Более того, даже в столице отдавать вопрос доступности собственных данных в руки интернет провайдера решается далеко не каждый. Вариантом гарантии активности интернет-подключения является создание

резервного канала связи, что зачастую оказывается непозволительно дорого для малого бизнеса.

- **Ненадежность**. Безопасность, а точнее её отсутствие является одним из главных препятствий развития «облаков» в России. Руководство многих компаний считает, что надежнее иметь свой физически ощутимый сервер в офисе, чем хранить данные на удаленных серверах. Это беспокоит малый и средний бизнес, который не хочет отдавать свои данные в чужие руки, а уж тем более крупные компании, отвечающие за сохранность конфиденциальных данных своих клиентов.

По словам генерального директора компании PR-group, в Европе 50% лабораторного тестирования информационных систем связано с информационной безопасностью (ИБ). В России данный сегмент почти полностью отсутствует. Законодательство РФ только начинает развиваться в направлении регулирования взаимоотношений провайдеров облачных вычислений и их клиентов. На данный момент все договоры в этой сфере носят смешанный характер и базируются на нормах гражданского права, включая право интеллектуальной собственности, международное и частное право, информационное право, законодательство о защите персональных данных. В связи с этим некоторые заказчики предпочитают зарубежные площадки. «Там более надежный сервис, законодательство, меньшие риски, удобство, прозрачность, гибкость политики».

- **Параноидальность**. Однако помимо перечисленных выше проблем в России существует еще одна, решить которую оказывается гораздо сложнее. Медленнее всего меняется менталитет, и в нашей бизнес-среде высок уровень «параноидальности» в отношении вопросов ИБ. Появляющиеся на рынке новые технологии и продукты оцениваются с точки зрения рисков специалистами по безопасности, и часто даже явные преимущества в экономической эффективности не могут перевесить предвзятое отношение и неготовность служб ИБ следовать технологическим веяниям. Снижение уровня «параноидальности» — вопрос времени. При этом позиция компаний в вопросах ИБ будет определяться требованиями защиты информации. Чтобы преодолеть фактор инерционность, компаниям, предлагающим облачные сервисы, необходимо решить огромный спектр технических, юридических и организационных вопросов, включая сертификацию продуктов и сервисов и т.п. Компаниям придется не только поднять уровень безопасности своих облачных решений, но и доказать безопасность своих сервисов клиентам.

Также крупный бизнес менее склонен к использованию публичных облаков, преимущества которых во многом обусловлены экономией за счет масштаба. В крупных

организациях размер инфраструктуры и так велик, поэтому выигрыш будет не столь значительным. Кроме того, у таких компаний обычно большой парк собственного оборудования, и им приходится решать задачи по интеграции систем. Такие компании чаще оказываются заинтересованы в организации частного облака для своих филиалов на собственной аппаратной базе.

Выводы

Переход к облачным вычислениям пока что не является панацеей для российского бизнеса, но ситуация меняется к лучшему. Технологии связи развиваются, Минкомсвязи создает государственный «облачный» сервис, который будет обслуживать все ведомства России. Рабочая группа, созданная в 2012 году, вынесла предложения по изменению и улучшению законодательства в сфере облачных вычислений. Основная причина непопулярности данных решений в неготовности многих ИТ-департаментов организаций к существенным изменениям инфраструктуры. Внедрение облачных решений на государственном уровне должно постепенно снизить скептическое отношение бизнеса к переходу в облака.

«Облака» способны обеспечивать непрерывность бизнеса лучше, чем это делают локальные решения. А это значит, что рано или поздно Россия в сфере облачных вычислений выйдет на мировой уровень.

Литература

1. Gillam, Lee. Cloud Computing: Principles, Systems and Applications / Nick Antonopoulos, Lee Gillam. — L.: Springer, 2010. — 379 p. — (Computer Communications and Networks). — ISBN 9781849962407.

2. SoCC '10: Proceedings of the 1st ACM symposium on Cloud computing / Hellerstein, Joseph M. — N. Y.: ACM, 2010. — ISBN 978-1-4503-0036-0.

3. Бизнес в облаках. Чем полезны облачные технологии для предпринимателя.– Интернет: <http://kontur.ru/articles/225>, 2012. – 1 с.

4. Облачные сервисы (рынок России).– Интернет: <http://www.tadviser.ru/index.php/Статья:Облачные_сервисы_(рынок_России)>, 2012. – 1 с.

УДК 004.93’12

РАСПОЗНАВАНИЕ ДОРОЖНЫХ ЗНАКОВ СРЕДСТВАМИ БИБЛИОТЕКИ OPENCV

П.В. Романов, Р.С. Самарев

Ключевые слова: распознавание, видео, opencv, сглаживание контура, контура моменты, компактность фигуры, объекта форма, контур, детектор границ, контрастность.

Key words: recognition, video, opencv, contour smothing, contour moments, figure compactness, object shape, contour, boards detector, contrast.

Аннотация. В статье рассматривается решение проблемы распознавания знаков дорожного движения, представляющих наибольшую опасность для водителя, средствами библиотеки openCV. Описываются алгоритмы их применения, а также по результатам тестирования в реализованной распознающей программе, формируются рекомендации по применению.

Введение

В настоящее время широко распространились различные системы распознавания объектов. В автоматизированных системах регулировки дорожно-транспортного движения также используются возможности этих систем. Однако в данный момент все еще нет в свободном доступе предметно ориентированных средств распознавания, рассчитанных именно на дорожную обстановку.

Цель проведенной работы – решение задачи распознавания дорожных знаков, а также проверка эффективности и удобства использования алгоритмов, реализованных в библиотеке openCV. Для получения видеопотока применялась камера, закрепленная в автомобиле, а именно видеорегистратор с довольно высокими характеристиками AdvoCam-FD7 Profi-GPS. Основными задачами были: фиксирование знака на видео с камеры, определение его типа по упрощенной классификации.

Распространенные методы распознавания

В большинстве программ подобного назначения используются обученные нейронные сети. Также применяются статистические методы анализа изображения, в частности, когда выбранные существенные признаки объекта характеризуются числовыми значениями, которые для объектов одного типа могут быть и различными. Среди них[1]:

- байесовское правило принятия решения, минимизирующее функцию риска;
- мини-максное правило принятия решения;

- правило принятия решения, основанное непосредственно на вероятности;
- критерий Неймана – Пирсона принятия решения;
- проверка многих гипотез;
- проверка сложных гипотез;
- метод максимума правдоподобия Фишера.

Все это методы, подражающие способам распознавания, применяемым живыми организмами. Также существует много методов чисто алгоритмических, наиболее популярные из которых[1]:

- метод построения эталонов;
- метод дробящихся эталонов;
- линейные решающие правила;
- структурный (лингвистический) метод;
- кластерный анализ;
- отбор информативных признаков.

С помощью выше перечисленных методов решают задачи на классическом ряде этапов распознавания[2]:

- сглаживание, фильтрация помех, увеличение контраста – морфологические преобразования;
- выделение контуров объектов рассмотренными и некоторыми другими методами
- первичная фильтрация контуров (по периметру, площади и т.д.);
- эквализация контуров (приведение к единой длине, сглаживание) – позволяет добиться инвариантности масштаба;
- перебор всех найденных контуров и поиск шаблона, максимально похожего на данный контур (или же сортировка контуров по какому-либо признаку).

Методы решения и ход разработки

В процессе анализа возможных подходов к решению задачи было решено не использовать «нейронные сети» и статистические подходы. Надо сказать, что в библиотеке компьютерного зрения есть много функций для статистического анализа и обучения по заранее приготовленным шаблонам. Однако в нашем случае проверялись более точные методы, реализованные в ней. Для решения задачи была написана программа-распознаватель. Процесс распознавания осуществлялся в три этапа:

- фиксация в кадре видеопотока области поиска и самого объекта (знака, в случае правильного срабатывания);
- анализ параметров найденного объекта, отнесение его к одному из типов (шаблонов из базы данных);

- адаптация программы для работы в более неблагоприятных условиях: в темноте, присутствии помех.

На каждом из указанных этапов вызываются функции из библиотеки. Обобщенная схема алгоритма анализа приведена на рисунке 1. В схеме показаны этапы и используемые функции OpenCV.

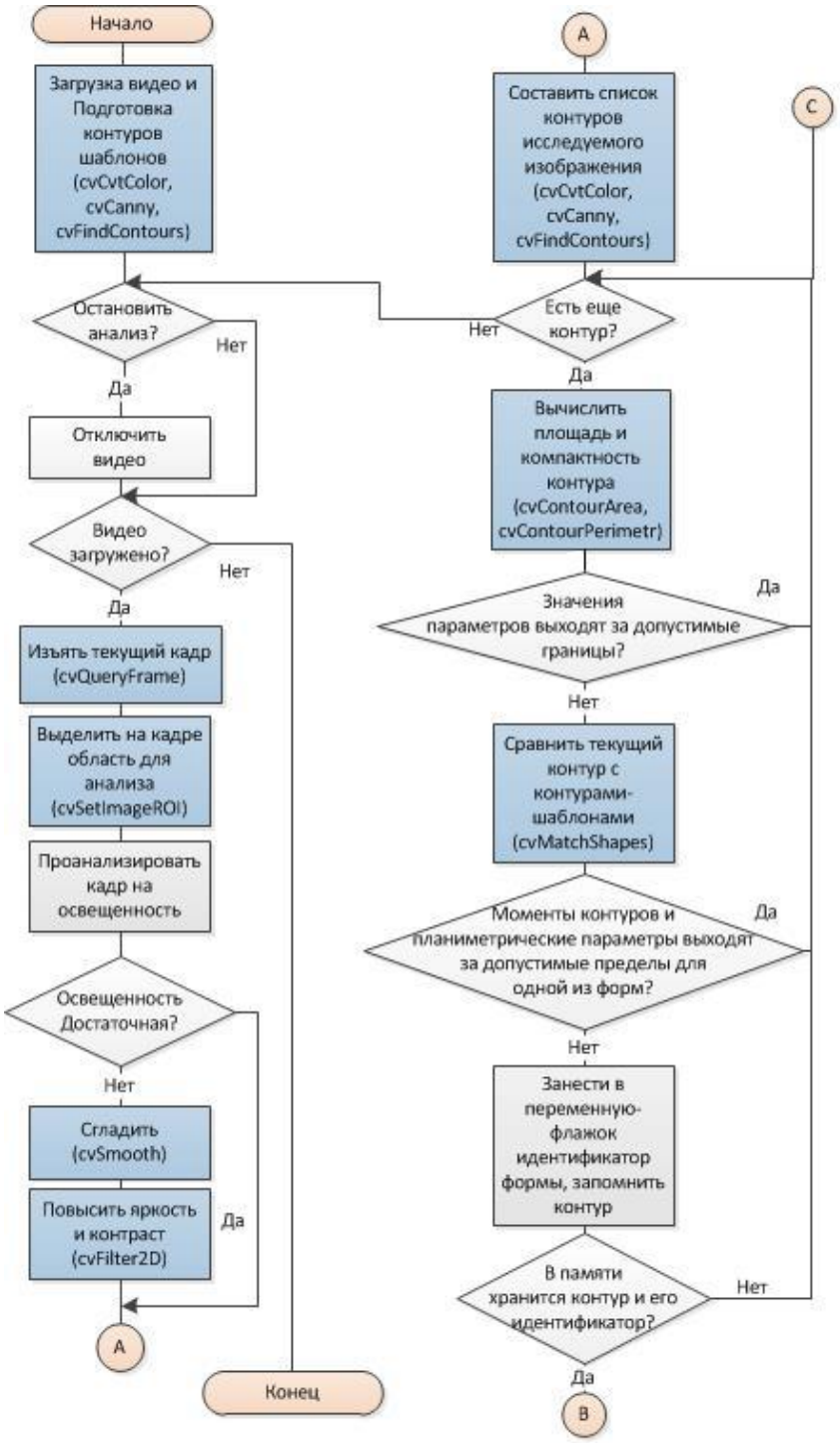


Рисунок 1.Обобщенная схема алгоритма (начало)

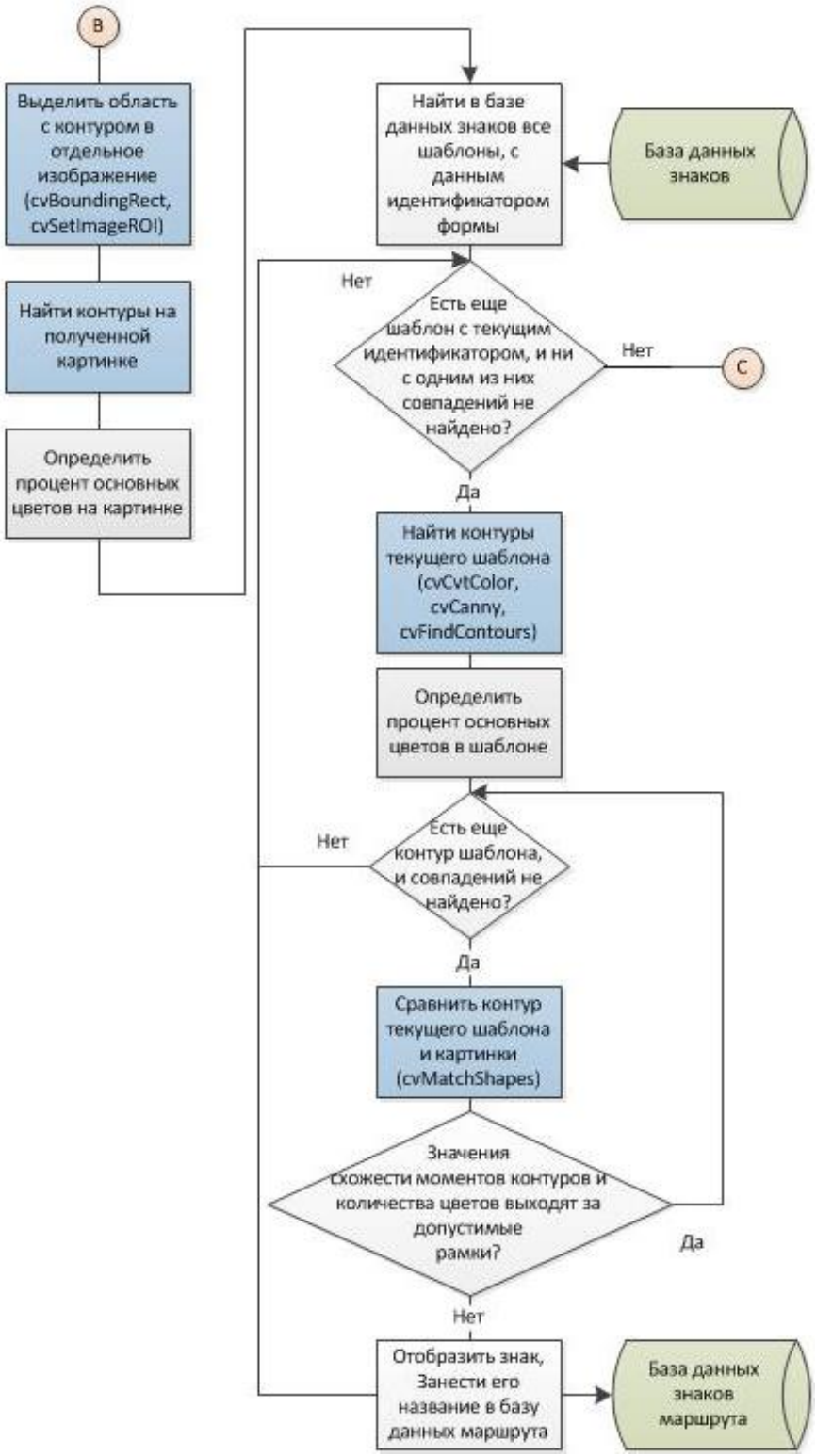


Рисунок 1. Обобщенная схема алгоритма (продолжение)

На первом этапе использованы функции cvSetImageROI, cvBoundingRect и cvCopy. Первая позволяет «сфокусироваться» на интересующем участке изображения (ROI – region of interest)[2]. На рисунке 3 можно увидеть, какую часть исходного изображения (рисунок 2) программа реально анализирует и представляет пользователю. Одни из ее параметров, как и у cvBoundingRect – относительные координаты левого верхнего угла

участка, который необходимо выделить – легко вычислить, поскольку в openCV также имеется специальный тип-структура IplImage, хранящий множество необходимой информации о картинке. Рассматривался вариант хранения изображения в переменных типа cvMat (матрица), но это потребовало бы использования объектов и функций нового стандарта, взаимодействие которых со старыми стало бы немного затруднительнее.

Рисунок 2. Исходное изображение

Рисунок 3. Анализируемая часть изображения (кадра)

Итак, после того, как выделена область, в которой наиболее вероятно появление знаков, полоса на уровне горизонта высотой в треть высоты кадра, нужно приготовить ее для поиска похожих на знаки форм. Для этого использовались следующие инструменты из openCV. Функция cvCvtColor, одной из опций которой является перевод картинки в градации серого. Это необходимо для применения порогового преобразования (в функция cvThreshold) или преобразования Кенни (функция cvCanny)[3]. Оба метода предназначены для бинаризации изображения(только черное и белое) и выделения на нем границ объектов, но первый оставляет на изображении только участки с яркостью не ниже заданного значения, тогда как алгоритм Кенни также убирает шумы, лишние детали, утончает и стягивает разорванные края. По результатам экспериментов было

подтверждено преимущество функции cvCanny, которая в итоге и была использована. Ниже (рисунок 4) показан результат последовательного применения указанных функций к участку кадра.

Рисунок 4. Результат последовательного применения cvCvtColor и cvCanny к изображению

Далее на изображении нужно найти все возможные замкнутые контуры. Для этого подошла функция cvFindContours, одним из параметров которой является бинаризированное изображение, полученное ранее[4]. Были попытки применить оператор Собеля для вычисления градиента яркости каждой точки картинки (функция cvSobel) и путем преобразования Хафа найти геометрические фигуры, в данном случае круги (функция cvHoughCircles). Ни одна из этих функций не показала себя в экспериментах достаточно универсальной. Они требуют точного подбора входных параметров, что сложно обеспечить при работе с видео, где картина постоянно меняется.

Следующий шаг – выбор контуров, удовлетворяющих заданным условиям. Первое, что анализировалось – площадь контура, поскольку она колеблется в заранее известных границах[2]. Ее можно получить, вызвав cvContourArea. cvContourPerimeter возвращает длину контура, что необходимо для вычисления второго параметра – компактности. Это отношение площади ко квадрату периметра. Проще говоря, характеризует схожесть объекта с кругом, поскольку круг – самая компактная фигура. У него этот коэффициент равен примерно 0,79, у прочих фигур это значение меньше. И наконец, проводилась проверка на совпадение моментов контуров (их отличительных форм) с помощью функции openCV cvMatchShapes, сравнивающей переведенные в цепной код Фримана контуры и выдающей коэффициент их отличия. Опыты показали, что почти полное сходство интерпретируется как значение меньшее чем 0.06 – 0.08. На рисунке 5 – пример

всех зафиксированных контуров (выделено ярко зеленым и фиолетовым), а на рисунке 6 – правильно узнанных знаков.

Рисунок 5. Все найденные в области контуры

Рисунок 6. Контуры, подошедшие по форме

По результату работы этой функции бывает нельзя точно сказать, подходит контур или нет, поэтому последний из приемов установления различий – это вычисление гаммы внутри контура. Иными словами, подсчет процента точек, цвет и яркость которых удовлетворяют некоторому условию. Для разбиения изображения на цветовые каналы применялись функции cvSplit и cvCvtPixToPlane. Однако функция cvCvtPixToPlane предназначена для разъединения изображения, представленного не цветовыми, а тональными каналами: оттенком, яркостью и глубиной (модель HSV – hue, saturation, value). В конечной реализации программы функция cvCvtPixToPlane не используется, поскольку по результатам тестов избыточна и в некоторых случаях не позволяет достоверно оценить цвет точки.

Второй этап отличается от первого тем, что анализируется изображение гораздо меньшего размера – предполагаемый знак. Но сравнение моментов найденных на нем контуров с помощью функции cvMatchShapes производится более тщательно. Происходит сопоставление со всеми знаками базы данных, удовлетворяющими условиям, проверенным на предыдущем этапе. Также производилось вручную контрастирование изображения для обеспечения более четкой работы функций детекции границ. Все описанное упрощалось за счет удобной организации в openCV структуры, хранящей изображения(IplImage), и типа данных cvRect для работы с произвольными прямоугольниками. Пользователю, в случае правильного срабатывания, будет представлена следующая информация (рисунок 7).

Рисунок 7. Пример работы в ясный день

Третий этап – адаптация программы для работы в сложных условиях – требовал действий с изображением, предшествующих первому этапу. То есть прежде чем начать искать контуры, нужно было подготовить их. Серьезными проблемами, которые потребовалось решать, были слабое освещение, множество предметов на изображении, удовлетворяющих одним и тем же условиям поиска, различного вида помехи. Для осветления и удаления помех предпринимались попытки воспользоваться функцию CvFilter2D, однако это не принесло желаемого результата, так как матрица свертки – основной параметр фильтра – должна подбираться под конкретные условия освещения, иначе возникает риск потерять нужные контуры[2][5]. Это недостаточно универсально. Вместо этого проводилось ручное контрастирования и сглаживание с использованием функции cvSmooth. Знание даже только параметров изображения уже даёт немалую свободу действий с ним. Далее (рис. 8) приведен пример работы программы в ночных условиях с применением названных методов.

Рисунок 8. Пример работы ночью

Как уже было отмечено, все указанные методы и типы реализованы в библиотеке openCV и имеют открытые коды.

Результаты

Выборка сформирована с учетом включения различных форм знаков и изображений на них. Тестировалось распознавание 7 знаков на 20 видео с разрешением 1920x1080.

Результаты распознавания

Пример знака	По классификации			Пример формы знака	По форме	
	Ясный день	Затенение, Ночь			Ясный день	Затенение, ночь
Обгон запрещен	85%	35%		Круглый	75%	50%
Въезд запрещен	65%	30%				
Ограничение скорости 40 км/ч	65%	45%				
Ограничение скорости 70 км/ч	70%	35%				
Поворот запрещен	40%	5%				
Уступите дорогу	70%	30%		Треугольный	75%	70%
Примыкание второстепенной дороги	80%	45%				

Заключение

По результатам проведенной работы можно выделить следующие преимущества и недостатки openCV. Применение функций позволило существенно сократить время разработки, упростить работу с изображением и представить его в удобной форме, получить множество его параметров. Что же касается задания сложных условий поиска, ее стандартных функций бывает недостаточно для решения задачи. Однако это не мешает продолжать использовать ее для быстрого выполнения операций, может быть, более простых, помогающих быстрее достичь основной цели. Сжатость сроков не позволила широко применить математический аппарат в комбинации с уже реализованными методами или тонкости того или иного подхода.

Удалось обеспечить распознавание в среднем 60% знаков, что свидетельствует о необходимости дальнейшего поиска решения в части повышения качества распознавания.

Литература

1) Лекции Ворошин Г. Я «Методы распознавания образов» [Электронный ресурс] // Учебно-методические материалы ВГУЭС [Офиц. сайт]. URL: http://abc.vvsu.ru/Books/Metody_r/default.asp (дата обращения 15.10.2013).

2) Статья Владимир Н. «OpenCV шаг за шагом» [Электронный ресурс] // ROBOCRAFT.RU: компьютерное зрение [Интернет-портал]. URL: http://robocraft.ru/page/opencv/ (дата обращения 25.11.2013).

3) Статья Распознавание образов с OpenCV: контуры против haartraining [Электронный ресурс] // PVSM.RU: Новости ИТ мира [Интернет-портал]. URL: http://www.pvsm.ru/algoritmy/30704 (дата обращения: 15.10.2013).

4) Gary Bradski, Adrian Kaehler «Learning OpenCV. Computer vision with the OpenCV library.» [Electronic resource] // LOCV.RU: learning OpenCV [Электронная книга]. URL: http://locv.ru/wiki/ (дата обращения: 22.11.2013).

5) RECOG.RU: распознавание образов для программистов [Интернет-портал]. URL: http://recog.ru/page/library/opencv (дата обращения: 02.11.2013).

УДК 004.77

НЕКОТОРЫЕ АСПЕКТЫ ОБЕСПЕЧЕНИЯ АНОНИМНОСТИ ИНФОРМАЦИИ НА САЙТЕ

Б.С. Романчиков, О. Ю. Еремин

Ключевые слова: сети, анонимность, информация.

Keywords: network, anonymity, information.

Аннотация: В данной статье рассмотрены некоторые аспекты обеспечения анонимности информации на сайте. Так же даны ответы на вопросы, возникающие при обеспечении анонимности, проведен сравнительный анализ наиболее популярных анонимных сетей и предоставлен способ разделение доступа к информации на сайте для пользователей использующих анонимную сеть и для пользователей использующую сеть I2P.

Введение

В современном мире практически каждый человек пользуется Интернетом для работы или развлечения. Люди обмениваются информацией, выкладывают файлы и просто работают в ней. Но Интернет не дает возможности остаться незаметным и,

теоретически, любое действие, которое выполнил пользователь можно отследить. Это стало причиной появления анонимных сетей.

Анонимные компьютерные сети — компьютерные сети, созданные для достижения анонимности в Интернете и работающие поверх глобальной сети. Специфика таких сетей заключается в том, что разработчики вынуждены идти на компромисс между степенью защиты и лёгкостью использования системы, её «прозрачностью» для конечного пользователя.[2]

При этом источник или получатель информации остается анонимным, соответственно никто не сможет понять, кто получил или кто выложил информацию. Сейчас это актуально тем, что в интернете возможно узнать расположение источника передаваемой информации, а значит к нему можно применить меры, ограничивающие информацию, которую он может передавать и получать.

Для обеспечения анонимности необходимо ответить на следующие вопросы:

1. какую анонимную сеть использовать;
2. как разделить информацию на сайте, которая доступна через анонимную сеть и которая доступна через сеть Интернет.

Сравнение I2P и Tor.

На данный момент существует множество анонимных сетей, но наиболее популярными являются сети Tor и I2P. Каждая из этих сетей предназначена для разных целей. И имеет свои особенности, достоинства и недостатки.

I2P предназначена для создания изолированной, закрытой сети, без выхода во внешний интернет. И хотя в Германии есть прокси-сервер "на выход", это не снимает концептуального предназначения системы. Фактически, изолированностью I2P обеспечивается приемлемый уровень анонимности и безопасности пользователей. I2P идеальна для анонимного и безопасного файлообмена, анонимного хостинга сайтов, анонимного общения. Это доступно только внутри сети I2P.[3]

Tor изначально предназначался для работы с открытым интернетом. С его помощью можно посещать заблокированные сайты, анонимно посещать сайты обычные, в общем – это динамически распределённый прокси-сервер, с несколькими промежуточными этапами. Кроме того, достаточно неплохо проанализирована устойчивость Tor-а к атакам на анонимность.[5]

Преимущества I2P по сравнению с Tor-ом:

1. выше скорость;
2. сложнее взломать и, как следствие, сеть анонимнее;
3. наличие псевдо-доменного пространства .i2p.

Недостатки:

1. сложнее настроить;
2. меньше пользователей;

В результате была выбрана сеть I2P, потому что она обладает большим количеством преимуществ по сравнению с Tor-ом. Главным достоинством является то, что I2P гораздо сложнее взломать, а это является ключевым фактором при выборе сети. Хоть у I2P на данный момент меньше пользователей, чем у Tor-а, сеть быстро развивается и в скором времени вполне может догнать или даже перегнать Tor по количеству пользователей в сети. На данный момент у Tor популярнее, потому что сеть гораздо старше, она стала доступной в 2002 году, а I2P лишь в 2009.

Перспективы развития I2P. За последний год сеть I2P становиться все более и более популярной. Причем самым активным пользователем сети I2P по количеству узлов является Российская Федерация это видно из рисунка. Специалисты связывают это с усилением цензуры в интернете. Кроме введения цензуры в России, руководитель проекта I2P Ларс Шиммер (Lars Schimmer) ещё одной причиной роста называет выход официального I2P-приложения под Android.

Согласно последним данным аудитория сети I2P содержит до 35 тысяч пользователей и их рост увеличивается. А значит, вполне возможно, что в скором будущем будут существовать множество сайтов использующих псевдо-домен .i2p.[6]

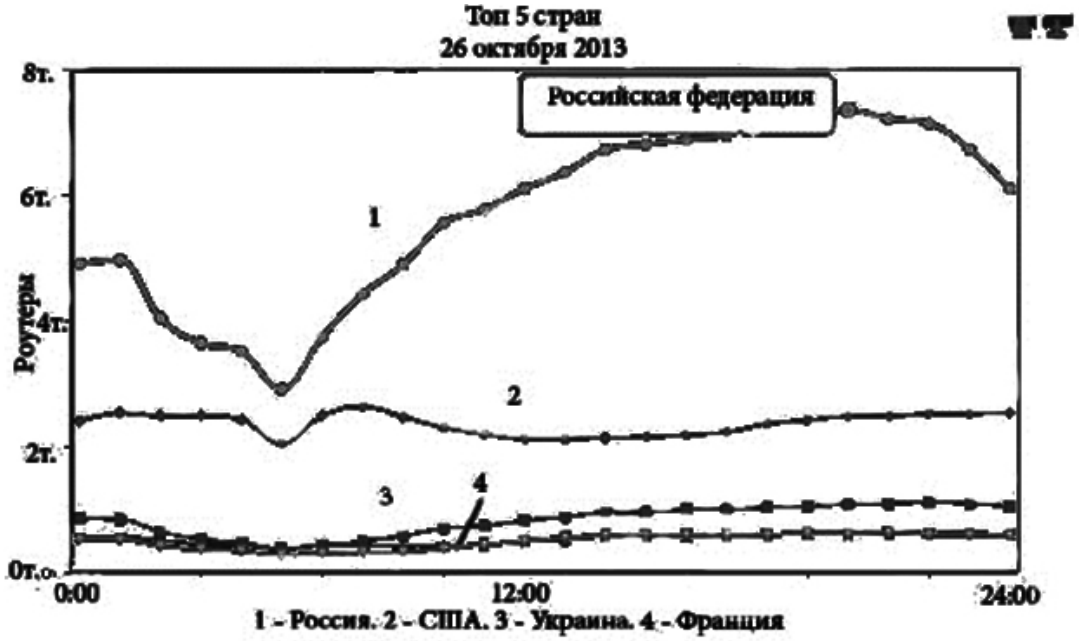


Рисунок. 5 стран, жители которых используют сеть I2P

План развития:

1. Версия 0.9
   1.1 Карты достижимости/сделать пиров частично доступнее/улучшенное ограничение для роутера
   1.2 Улучшение страниц помощи и веб-сайта
   1.3 Больше трансляций
   1.4 SSU сообщения о разрыве
   1.5 Интерактивный floodfill поиск.
2. Версия 1.0
   2.1 Полный обзор уязвимостей, связанных с анонимностью и другим
   2.2 Уменьшить использование памяти, убрать отладочные неизбежные накладные расходы. Сделать работу сети лучше на медленных машинах.
   2.3 Документация
3. Версия 2.0
   Полное ограничение по маршрутам
4. Версия 3.0
   4.1 Туннельные смешивания и заполнения.
   4.2 Использование различной задержки сообщений. [6]

Поддержка сайтом сети I2P и разделение доступа между Интернетом и I2P

Для того, что бы пользователь сети I2P мог получить доступ к сайту, необходимо установить официальный клиент сети с сайта. Затем необходимо создать туннель для сайта и разрешить веб-серверу локально взаимодействовать. При этом в настройках клиента можно указать количество узлов, которые используются в туннеле. Количество узлов влияет на безопасность и скорость. Чем больше узлов в туннеле тем сложнее получить адрес конечного узла, но при этом и скорость меньше. Таким образом пользователь сможет заходить на сайт через сеть I2P.

Для разделения доступа между Интернетом и I2P можно написать программу на любом серверном языке, которая предоставляет доступ к определенному ресурсу только для определенного IP адреса, если клиент для I2P установлен локально то адрес 127.0.0.1. Это связанно с тем, что сеть I2P можно представить как одну большую локальную сеть. Если клиент установлен на другом узле, то необходимо указать его IP адрес. Таким образом, определенный контент будет доступен только для сети I2P.

Преимущества:

1. Нельзя модерировать или запретить данные, которые выложены для сети I2P;
2. Контент на сайте будет доступен через сеть I2P.

Недостатки:

1. Медленная работа сети I2P;
2. Наличие туннеля, что может потреблять ресурсы сервера.

Заключение

Для обеспечения анонимности необходимо было выбрать анонимную сеть. Выбор состоял между сетью I2P и Tor. В результате была выбрана сеть I2P как более анонимная и при этом имеющая не очень существенные недостатки. Так же были рассмотрены перспективы развития этой сети и предложен способ разграничения информации доступной в сети Интернет и доступной в сети I2P.

Литература

1. Таненбаунм Э. Компьютерные сети. М.: Питер, 2003. 993 с.
2. Анонимные сети – Википедия [электронный ресурс]//метод доступа: ru.wikipedia.org: Википедия – свободная энциклопедия. 2000. URL. http://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%BE%D0%BD%D0%B8%D0%BC%D0%BD%D1%8B%D0%B5_%D1%81%D0%B5%D1%82%D0%B8#Invisible_Internet_Project_.28I2P.29.
3. I2P – Википедия [электронный ресурс]//метод доступа: ru.wikipedia.org: Википедия – свободная энциклопедия. 2000. URL. http://ru.wikipedia.org/wiki/I2P.
4. I2P:Принципы функционирования основных сервисов сети [электронный ресурс]//метод доступа: Habrahabr.ru. 2006. URL. http://habrahabr.ru/post/152771/.
5. I2P или Tor – что лучше? [электронный ресурс]//метод доступа: www.shpargalko.com.2009. URL. http://www.shpargalko.com/2011/03/10/i2p-ili-tor-chto-luchshe/.
6. Анонимная сеть I2P//i2p2.de.URL.http://www.i2p2.de/roadmap.html.

УДК 004.056

СРЕДСТВО ДЛЯ ПОСТРОЕНИЯ ПРОГРАММ ТЕСТИРОВАНИЯ ВИДЕОСТЕГАНОГРАФИИ

Р. Ю. Сорокин, Р.С. Самарев

Ключевые слова: видеостеганография, виртуальная сеть, средство тестирования, видеотелефония, обход цензуры, программный продукт.

Key words: *video steganography, virtual network, testing tool, video telephony, bypassing censorship, software.*

Аннотация: *В данной работе рассматривается архитектура программного средства для тестирования алгоритмов видеостеганографии в контексте создания виртуальной скрытой сети поверх цифровой видеотелефонии. Особое внимание уделяется необходимости предварительного тестирования алгоритмов стеганографии, наилучшим образом подходящих для использования в такой сети. Предлагается новый программный продукт для тестирования алгоритмов видео стеганографии и анализа, аналогов которого не существует. При реализации программы был использован объектно-ориентированный подход к программированию и библиотека компьютерного зрения openCV. В качестве основных требований к программе выступали простота, производительность, гибкость, расширяемость и кроссплатформенность. Тестирование программы доказало работоспособность разработанной архитектуры.*

Введение

В Конституциях большинства стран мира гарантируется право на тайну коммуникаций, однако в настоящее время возросло количество фактов слежки за пользователями сети Интернет со стороны спецслужб. Введение цензурных барьеров и фильтрация трафика ограничивают возможность свободно обмениваться информацией в сети. Для решения двух вышеописанных проблем требуются программные средства, способные выполнять следующие функции:

- скрывать от провайдера трафик;
- зашифровывать трафик от отправителя до получателя [1].

Сегодня существует ряд программных средств, которые тем или иным образом способны обеспечить двустороннюю секретную передачу данных между пользователями в сети, как например Tor, I2P и некоторые другие. Недостаток подобного рода систем заключается в том, что:

- факт наличия трафика от этих программ может быть выявлен с помощью специальных анализаторов трафика у провайдера (DPI) [2];
- все программные системы имеют разные цели и архитектуру, таким образом, плохо совместимы;
- некоторые программы (например, I2P) пересылают много транзитного трафика, предназначенного другим пользователям, что может засорять канал передачи данных.

Для организации обмена секретными данными, когда участники знают друг друга, требуются системы другого класса, наиболее успешной из которых является SkypeHide

[3]. SkypeHide использует Skype по принципу LACK, однако стеганограмма передается с помощью специальных 70-битных пакетов, отправляемых программой Skype при наличии тишины (условная архитектура существующей системы приведена на рисунке 1) [4]. Система имеет два серьезных недостатка:

- зависимость от программы Skype (если в будущем архитектура системы Skype будет изменена – потребуется изменять и программу SkypeHide);
- несовместимость с другими средствами видеотелефонии.

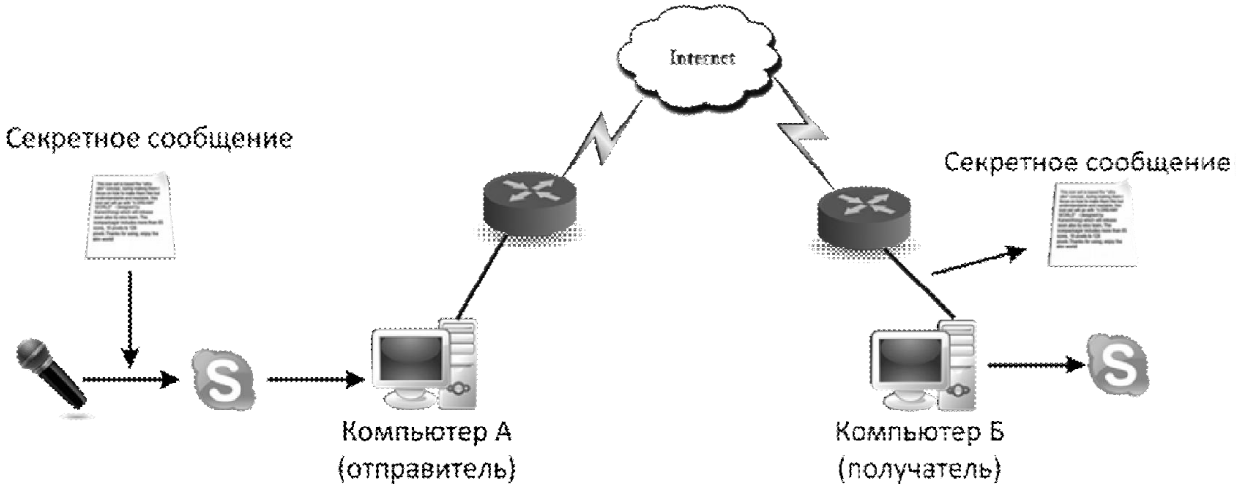


Рисунок 1. Архитектура функционирования существующей системы SkypeHide

Модель усовершенствованной системы двустороннего обмена данными

Принципиально новая система может быть построена на основе существующих систем видеотелефонии, но независимо от особенностей функционирования конкретных IP-телефонов. В такой системе секретные данные будут добавляться к видео, захватываемому с камеры пользователя. Принцип функционирования показан на рисунке 2. Видеотелефон на «Компьютере А» захватывает данные с виртуальной вебкамеры, которая в свою очередь осуществляет добавление секретных данных к захватываемому видео с настоящей вебкамеры. Таким образом, добавление секретных данных к видеопотоку происходит до того, как они попадают непосредственно в видеотелефон. Далее видеотелефон отправляет данные по сети. Видеотелефон на «Компьютере Б»выводит видеоданные на экран, после чего программа для считывания стеганограммы осуществляет извлечение секретных сообщений.

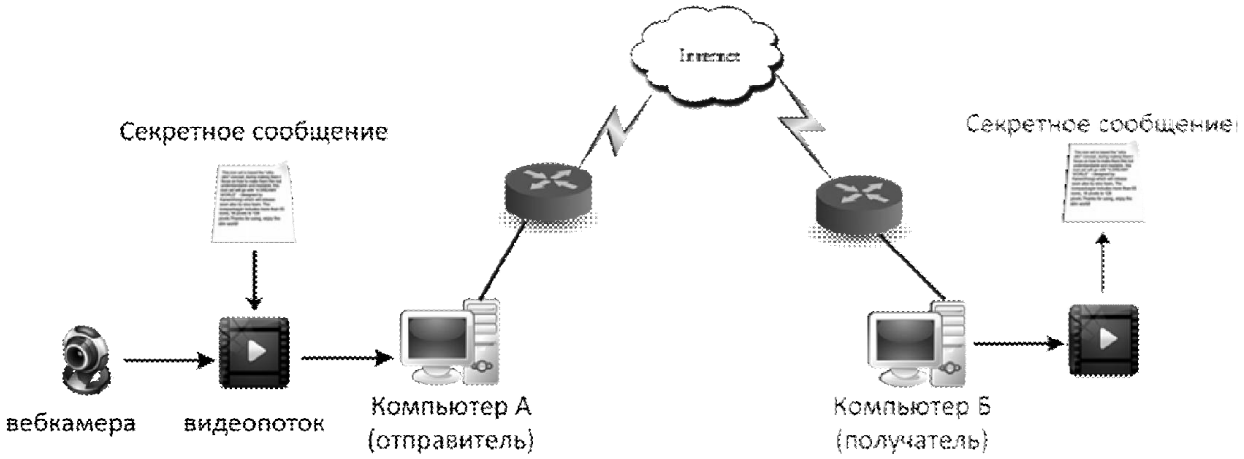


Рисунок 2. Архитектура новой системы секретного обмена данными

Расширив функционал системы, можно организовать виртуальный сетевой интерфейс над видеотелефонией, используя видеопоток в качестве физической среды передачи пакетов данных. В этом случае возможно построение виртуальной скрытой сеть между двумя компьютерами, а полный стек сетевых протоколов автоматически обеспечит операционная система. Через вышеописанную систему можно будет передавать не только данные из определенного файла (статически заданные до момента развертки системы), а также и данные, поступающие в реальном времени, например, трафик от интернет пейджера, веб-сервера и т.д. Структура и пример использования стека протоколов показан на рисунке 3. Веб-сервер, развернутый на «Компьютере А» отправляет данные через виртуальную сеть веб-браузеру, запущенному на «Компьютере Б». При этом браузер может одновременно передавать данные веб-серверу. Это достигается за счет того, что на обоих хостах А и Б функционирует одинаковое ПО.

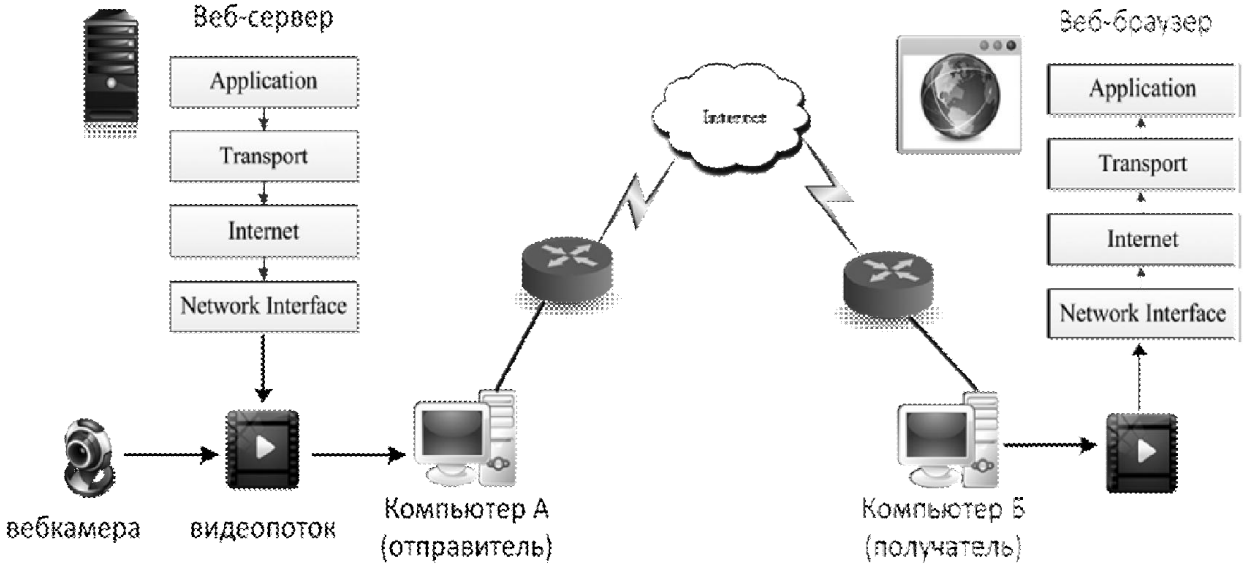


Рисунок 3. Организация стека протоколов на примере отправки данных с веб-сервера браузеру

Задача построение такой системы сводится к написанию виртуального драйвера сетевого интерфейса, ответственного за передачу данных в модуль виртуальной камеры и виртуального драйвера видеокамеры, ответственного за добавление данных к видеопотоку, полученных от сетевого интерфейса.

Преимущества использования стеганографии в потоковом видео перед использованием стеганографии в других данных (изображениях\тексте):

- видео трафик сложнее хранить для последующей обработки;
- большое количество пользователей видео телефонии;
- независимость от оператора видео телефонии и конкретного программного видеотелефона;
- достаточная пропускная способность канала для секретных сообщений;
- возможность двунаправленно обмениваться данными в режиме реального времени, в том числе и обмениваться открытыми криптографическими ключами;
- возможность организации двунаправленного стека протоколов поверх видео телефонии;
- возможность обходить цензурные барьеры, в связи с тем, что секретный трафик спрятан внутри «разрешенного» видеотрафика.

Преимущества использования новой архитектуры перед существующей:

- система не зависит от провайдера видео телефонии (возможно использование как с проприетарным ПО, так и с видеотелефонами с открытыми исходными кодами);
- устойчивость к искажениям на пути следования видео трафика (в том числе умышленным) может задаваться конкретными алгоритмами стеганографии;
- возможность регулирования пропускной способности канала применяемым методом стеганографии;
- степень скрытости данных также может регулироваться с помощью применяемого алгоритма стеганографии.

Как было отмечено выше, для функционирования подобной системы важно подобрать набор методов видео стеганографии, которые бы удовлетворяли различным условиям секретности и пропускной способности. В настоящее время существуют несколько программных продуктов, тем или иным образом связанных с тестированием стеганографии, среди которых наиболее примечательны:

- *Steganosaurus* – система тестирования аудио- и видеостеганографии. Поддерживается только один формат видео MP4, нет возможности захвата видео с камеры, отсутствует гибкая возможность добавления новых методов. Отсутствует

возможность добавления собственных фильтров и имитаторов искажений. Нет удобного вывода на экран в режиме реального времени для визуальной оценки искажений [5].

- *RT Steganography* – приложение, предназначенное для поточной стеганографии. Находится в стадии разработки. Бесплатная. Отсутствует возможность добавлять собственные алгоритмы стеганографии, анализа, искажений [6].

Ни одна из ныне существующих программ не дает исследователю стеганографических алгоритмов максимально удобного и эффективного функционала. Возникает потребность в программном продукте, способном протестировать различные методы поточной видеостеганографии и методы стенографического анализа. Рассмотрим архитектуру разработанного программного средства, которое может быть использовано в качестве основы для создания полнофункционального средства тестирования видеостеганографии.

Разработка программы «Видеостеганографический Фреймворк»

По результатам анализа типовых задач при оценке алгоритмов видеостеганографии была сформулирована задача разработать набор программных средств «Видеостеганографический Фреймворк», предназначенный для тестирования видеостеганографических методов и методов анализа, который имел бы следующие особенности:

1. Простота использования. Программа должна быть интуитивно понятна разработчику методов стеганографии, иметь дружественный пользователю интерфейс, обладать необходимой отказоустойчивостью.
2. Производительность. Программа должна справляться с обработкой видео в режиме реального времени, выводя на экран несколько окон с видеоданными.
3. Гибкость и кроссплатформенность. Разрабатываемое ПО должно иметь возможность дальнейшей модернизации и перекомпиляции под различные платформы.
4. Расширяемость за счет удобного добавления новых методов стеганографии, имитации искажений видео и анализа, в том числе статистического.

При создании набора программных средств тестирования видеостеганографических методов были использованы следующие средства программирования и разработки ПО:

1. Базовая платформа: операционная система с ядром Linux, и библиотека Qt для языка программирования C++. При необходимости программа может быть легко перенесена на другие операционные системы.
2. Библиотека компьютерного зрения openCV. ПО использует стандартные функции захвата видео с камеры, искажения и вывода на экран.

3. Для удобства подключения новых функция использован объектно-ориентированный подход к программированию, что позволяет добавлять новые классы обработки и анализа видеокадров в виде классов потомков предоставленных интерфейсов (архитектура разрабатывалась под условия расширяемости).

Для реализации программного продукта была разработана следующая архитектура, кратко представленная на рисунке 4. Видео, захватываемое с вебкамеры пользователя, последовательно проходит несколько блоков. При этом на экран выводятся 6 окон, отображающих видеоданные в режиме реального времени:

- окно захваченного видео (1);
- окно с видеоданными, подвергшимися обработки блоком модификации (2);
- 2 окна видеоданных, полученных в результате работы анализаторов (3,4);
- окна вывода сравнительного и неинформированного анализатора (5,6).

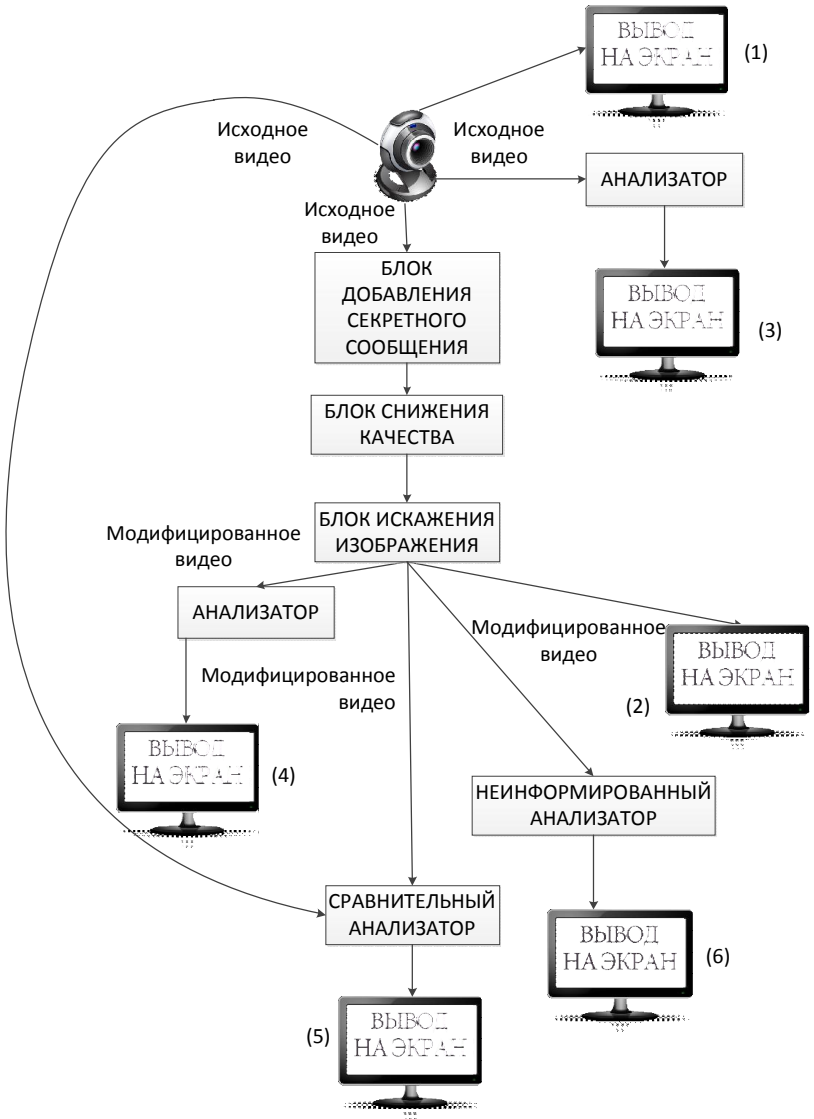


Рисунок 4. Архитектура программы «Видеостеганографический Фреймворк»

Такая архитектура наиболее удобна в использовании – пользователь в реальном времени видит результат работы примененного алгоритма стеганографии, а также результаты анализаторов.

Краткая диаграмма классов программы «Видеостеганографический Фреймворк» приведена на рисунке 5. Для реализации собственного алгоритма стеганографии, искажений или анализа достаточно унаследовать класс от классов *FrameProcessor*, *Losses* или *FrameAnalyzer* соответственно и передать адрес объекта функции-обработчику. Программа осуществит покадровый захват видео и обработку каждым таким пользовательским объектом.

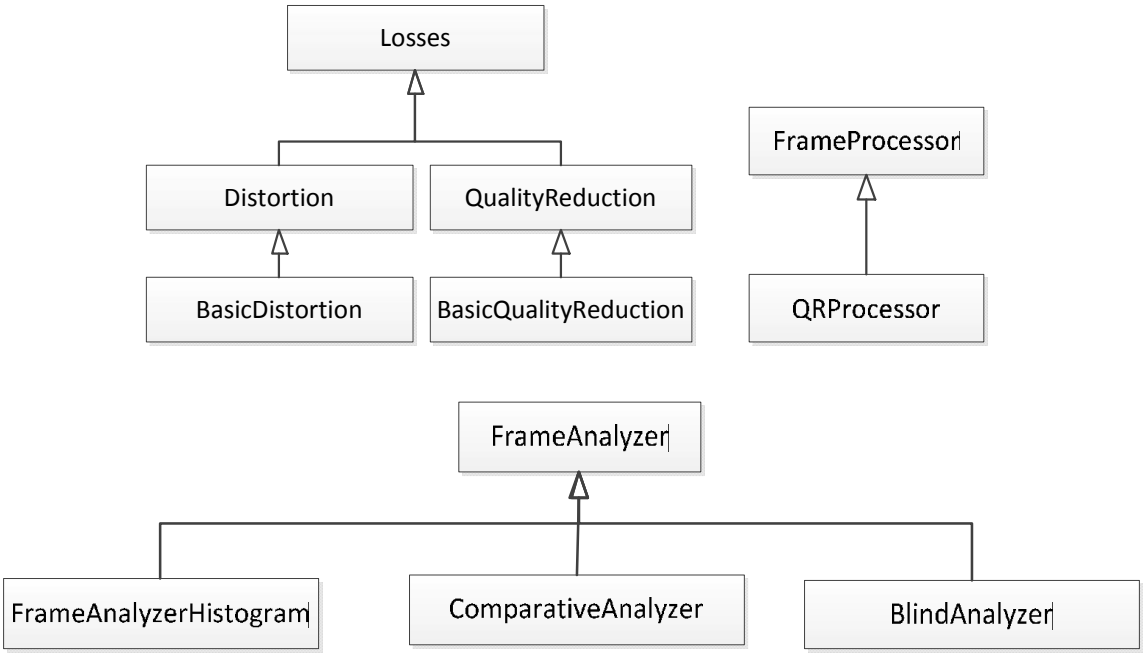


Рисунок 5. Краткая диаграмма классов программы

Базовый класс *Losses* является абстрактным классом. Все классы, реализующие какие-либо типы искажений, также являются абстрактными, и должны быть унаследованы от *Losses* (классы *Distortion* и *QualityReduction*). Классы, реализующие непосредственно сами операции искажений, должны наследоваться от конкретного класса-типа искажения с переопределением виртуального метода *process*, в который передаются адреса объектов неизмененного кадра и модифицированного. Такая организация классов удобна при добавлении как новых классов, непосредственно реализующих обработку кадров, так и новых классов-типов искажений.

Базовый класс *FrameProcessor* является родителем для всех классов, выполняющих добавление стеганограмм. Предполагается, что разработчик имеет возможность создания в наследуемом классе вектора объектов-кадров, чтобы осуществлять добавление сложных стеганограмм, данные в которых зависят от приходивших ранее кадров.

Базовый класс *FrameAnalyzer* также является абстрактным классом. Все классы, реализующие анализ какого-либо вида должны быть унаследованы от *FrameAnalyzer* с переопределением метода *analyze*. Класс *FrameAnalyzer* снабжен вектором кадров, хранящим все кадры, попадавшие в анализатор. Класс позволяет вернуть кадр с заданным номером, что удобно использовать при статистическом анализе.

Для решения задачи обработки кадров видеопотока были использованы следующие компоненты библиотеки openCV:

- *Mat* – класс кадра;
- *open(<имя файла или устройства>)* – открытие файла или устройства видеозахвата;
- *cv::namedWindow(<имя окна>)* – вывод именованного окна с видеоданными;
- *cv::medianBlur(<список параметров>)* – медианное размытие кадра;
- *cv::boxFilter(<список параметров>)* – фильтр попиксельной аппроксимации;
- *cv::GaussianBlur(<список параметров>)* – фильтр Гаусса;
- *cv::resize(<список параметров>)* – изменение геометрических размеров кадра с сохранением остальной информации о кадре.

Пример использования программы «Видеостеганографический Фреймворк»

В качестве примера использования и тестирования архитектуры были разработаны специальные наглядные алгоритмы:

- в качестве метода стеганографии – добавление QR-кода к выбранному цветовому каналу RGB видеоизображения (класс QRProcessor на рисунке 5);
- в качестве метода искажения - медианное размытие кадров (класс BasicQualityReduction на рисунке 5);
- в качестве метода сравнительного анализа – разностный сравнительный анализ исходного видео, захваченного с камеры и результирующего видео, полученного в результате применения метода добавления QR кода и медианного размытия (класс ComparativeAnalyzer на рисунке 5);
- в качестве метода неинформированного анализа – неинформированный визуальный анализ по заданному цветовому каналу (класс BlindAnalyzer на рисунке 5);
- в качестве оценочного метода анализа – методы вычисления гистограмм исходного и результирующего кадра по трем цветовым каналам RGB (класс FrameAnalyzerHistogram на рисунке 5).

Тестирование проводилось на компьютере с Intel Core i5 2.70 GHz , 2Gb RAM, OS Ubuntu 12.04.

На рисунке 6 показаны исходное видео и видео со встроенным QR-кодом, подвергшееся медианному размытию. Подмеска проводилась в красный канал RGB путем увеличения значения интенсивности пикселя, попадающего в область QR-кода. Как видно из правого рисунка – QR код практически не заметен.

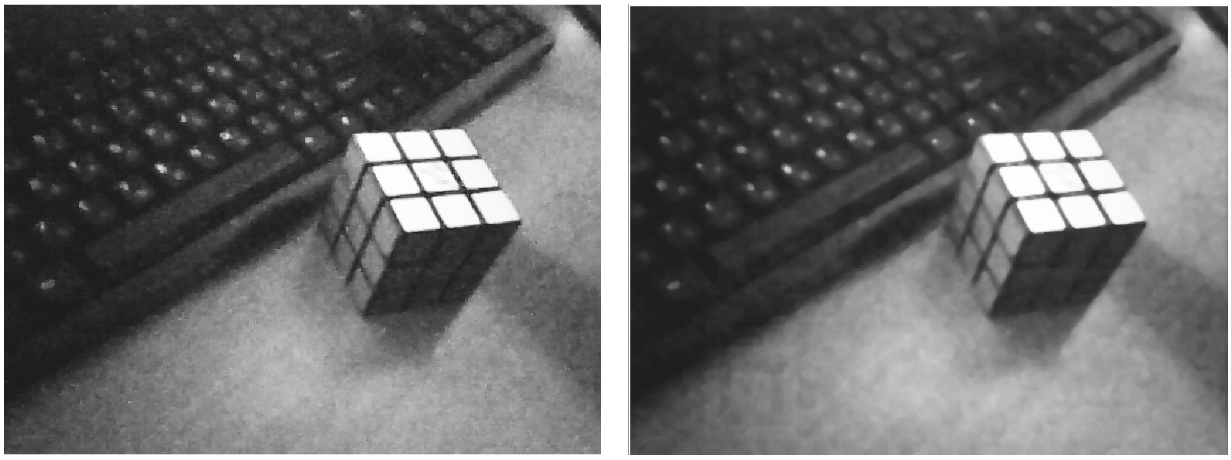


Рисунок 6. Исходное видео (слева) и видео, содержащее QR-код в красном канале, подвергшееся медианном размытию (справа)

На рисунке 7 показан результат работы сравнительного и неинформированного анализаторов. QR-код слева получен путем вычитания из интенсивности каждого канала каждого пикселя результирующего искаженного видео интенсивности каждого канала каждого пикселя исходного видео. При определенной яркости фона и качестве печати возможно распознать QR-код с помощью стандартных распознавателей для Android, iOS и т.д. Секретное сообщение представляет собой фразу: *“enter your secret message here”*.

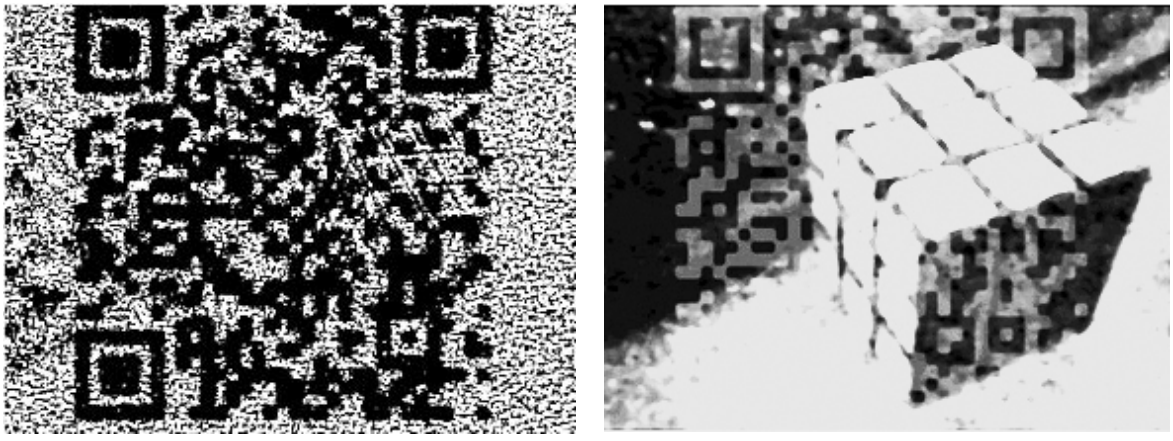


Рисунок 7. Результат работы сравнительного анализатора (слева) и результат работы неинформированного визуального анализатора (справа)

На рисунке 8 показаны результаты работы визуальных анализаторов, выводящих на экран гистограммы исходного видео, которое захвачено с вебкамеры, и результирующего видео с добавленным QR-кодом. По оси абсцисс – интенсивность цвета пикселей (от 0 до 255), по оси ординат – количество пикселей с данной интенсивностью. На гистограмме справа явно представлено увеличение интенсивности красного цвета. Такой анализ может быть усовершенствован добавлением статистического покадрового анализа уровней каждой из трех гистограмм.

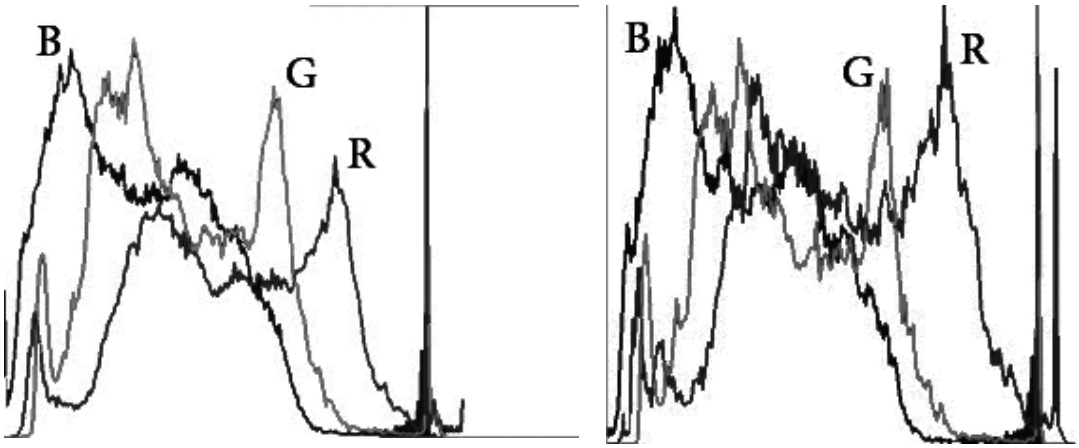


Рисунок 8. Гистограммы исходного (слева) и результирующего видео (справа)

Заключение

В результате работы была предложена модель новой системы секретной передачи данных, работающей поверх видеотелефонии. Для создания эффективных алгоритмов стеганографии для такой системы было разработано программное обеспечение, предназначенное для тестирования видеостеганографических методов и методов анализа. Специально созданный пример использования показал работоспособность разработанной архитектуры.

В качестве дальнейшей работы в этой области автор статьи предполагает усовершенствовать функционал программы «Видеостеганографический Фреймворк», наполнить его наиболее актуальными на сегодняшний день методами анализа, в том числе добавить статистический анализ. После проведенных сравнительных тестов будет выбран наиболее подходящий алгоритм для использования в архитектуре новой виртуальной сети.

Проект доступен в сети Интернет по адресу:

github.com/quattromann/Realtime-videosteganography-test-framework

Литература

1. Ding Feng, Yang Zhiqing, Chen Xuelong, Guo Jianfeng. Effective Methods to Avoid the Internet Censorship. Institute of Electrical and Electronics Engineers. Jan 11, 2012. pp. 67-71.
2. I2P Introduction http://www.i2p2.de/techintro.html (дата обращения - 20.11.2013).
3. Wojciech Mazurczyk, Maciej Karaś. Krzysztof SzczypiorskiSkyDe: a Skype-based Steganographic Method // INT J COMPUT COMMUN. Warsaw., -2013. C. 432-443.
4. Wojciech Mazurczyk, Józef Lubacz. LACK—a VoIP steganographic method. Telecommunication Systems, Volume 45 (2) – Oct 1, 2010.
5. Steganosaurus main page http://www.steganosaur.us (дата обращения - 16.11.2013).
6. RT Steganography wiki page. http://sourceforge.net/p/rtstegvideo/wiki/Home (дата обращения 14.11.2013).
7. Gandharba Swain, Saroj Kumar Lanka. A Quick review of Network Security and Steganography. // International Journal of Electronics and Computer Science Engineering. -2012. C. 426-435.
8. Kaushal M. Solanki. Multimedia Data Hiding: From Fundamental Issues to Practical Techniques. Santa Barbara. University of California. 2005. 250 с.
9. Steganography And Digital Watermarking / Jonathan Cummins [и др.]. Birmingham. School of Computer Science, The University of Birmingham. 2004. 24 с.
10. Wen-Chuan Wu, Zi-Wei Lin, Wei-Teng Wong. Application of QR-Code Steganography Using Data Embedding Technique. // Lecture Notes in Electrical Engineering. Volume 253. -2013, C. 597-605.
11. Digital Watermarking and Steganography. / Ingemar J. Cox [и др.]. Burlington. Morgan Kaufmann Publishers. 2008. 587 с.
12. Andreas Westfeld, Gritta Wolf. Steganography in a Video Conferencing System. Lecture Notes in Computer Science Volume 1525, 1998, pp 32-47.
13. Daniel Socek, Hari Kalva, Spyros S. Magliveras, Oge Marques, Dubravko Culibrk, Borko Furht. New approaches to encryption and steganography for digital videos. September 2007, Volume 13, Issue 3, pp 191-204.
14. Steganography software. http://www.jjtc.com/Steganography/tools.html (дата обращения - 10.10.13).

УДК 004.057.4

ВИРТУАЛИЗАЦИЯ КАНАЛА ПЕРЕДАЧИ ДАННЫХ В КОМПЬЮТЕРНЫХ СЕТЯХ УДАЛЁННЫХ ЛАБОРАТОРИЙ

И.Е. Титов, Г.С. Иванова

Ключевые слова: *удалённые лаборатории, клиент-серверная архитектура, сетевые протоколы, HTTP(S), web socket, виртуализация канала.*

Key words: Remote laboratories, client-server architecture, network protocols, HTTP(S), web socket, virtualized channel.

Аннотация: *Существующие в настоящее время удалённые лаборатории используют один протокол передачи данных для всех типов данных. В типичной удалённой лаборатории передаётся информация с разными характеристиками и к каждому из типов данных предъявляются различные требования. В данной работе мы предлагаем использовать специализированные протоколы для каждого из типов данных (текстовой информации, изображений и потокового видео) и объединить их в один виртуальный протокол в пределах одной удалённой лаборатории. Показывается, что подобное соединение может быть установлено по стандартным портам (80 или 443), быть эффективным с точки зрения трафика и задержки, и требует для реализации три сервера. Описывается опытная реализация такого решения на базе платформы удалённых лабораторий Labicom.net.*

Введение

В настоящее время при подготовке инженерных кадров всё большую значимость приобретают информационные технологии, в частности, удалённые и виртуальные лаборатории, позволяющие студентам получать доступ к лабораторному оборудованию посредством компьютера, подключенного к сети Интернет. Не смотря на всё разнообразие инженерных и научных задач, изучаемых в таких лабораториях, существует большое количество организационных, технических и методологических черт, одинаковых для всех on-line лабораторий. К таковым, в том числе, относится необходимость передачи данных в сети Интернет между лабораторным сервером и конечным пользователем (как правило, студентом). Однако передаваемые по сети данные в удалённых лабораториях характеризуются большой разнородностью и рядом ограничений на использование сетевых протоколов. В данной работе мы будем исходить из клиент-серверной архитектуры удалённой лаборатории, в которой клиентская часть выполняется в веб-браузере [1]. Типичная удалённая лаборатория, построенная по клиент-серверной

архитектуре (такая, как например RLL [2]), состоит из экспериментального оборудования, лабораторного сервера, веб-сервера (такого как Labicom.net [3]) и веб-клиента. Лабораторным сервером является компьютер, к которому по различным интерфейсам (USB, RS-232, GPIB, IP, Wi-Fi) непосредственно подключено экспериментальное оборудование.

В типичной удалённой лаборатории используется как минимум три разных типа данных: статические данные (содержимое веб-страницы, стили, скрипты, медиа и т.д.), динамические данные (команды от клиента к лабораторному серверу и обновления от лабораторного сервера к клиенту) и потоковое видео с лабораторного стенда. Важным ограничивающим фактором при построении компьютерной сети удалённой лаборатории и написании клиентской части приложения является гарантированная доступность только двух стандартных портов для передачи данных между лабораторным сервером и клиентом через веб-сервер – 80 (протокол HTTP) и 443 (протокол HTTPS). Данные протоколы подходят для передачи статических данных и могут быть использованы для передачи потокового видео, однако плохо приспособлены для передачи динамических данных. HTTP/HTTPS является запросным, вследствие чего сервер не может самостоятельно обратиться к клиенту [4, 5]. Для поддержания актуальности данных приложение-клиент должно постоянно опрашивать сервер на предмет обновлений. Чтобы создать видимость обновлений в реальном времени, клиент должен опрашивать сервер несколько раз в секунду. HTTP(S) состоит из заголовка (HEADER) и тела (BODY) запроса. При каждом запросе (POST) и ответе HEADER может достигать нескольких тысяч байт даже при отсутствии полезной нагрузки в BODY. Таким образом, при отсутствии действий пользователей и обновлений параметров лабораторного стенда передача данных по сети является крайне неэффективной с точки зрения сетевого трафика и задержки при передаче данных.

В типичной удалённой лаборатории, построенной по клиент-серверной архитектуре с веб-браузером в качестве клиента, все данные передаются по протоколу HTTP(S), запросы которого обрабатываются одним веб-сервером. В данной работе предлагается использовать несколько специализированных веб-серверов, а различные (специализированные) каналы передачи данных на время подключения приложения-клиента объединять в один виртуализированный канал.

Использование протокола HTTP(S) для передачи динамических данных

Передаваемые в сетях удалённых лабораторий экспериментальные данные как правило представляют собой текстовые данные. Данные, идущие от клиента к серверу, представляют собой команды, выполняемые лабораторным сервером. Данные идущие в противоположную сторону представляют собой обновление состояния конфигурации экспериментального оборудования и полученные экспериментальные данные. Последние могут быть как скалярными величинами, так и массивами чисел. Таким образом, в среднем от сервера к клиенту в типичной клиент-серверной архитектуре удалённой лаборатории (рисунок 1) передаётся больше данных, чем от клиента к серверу.

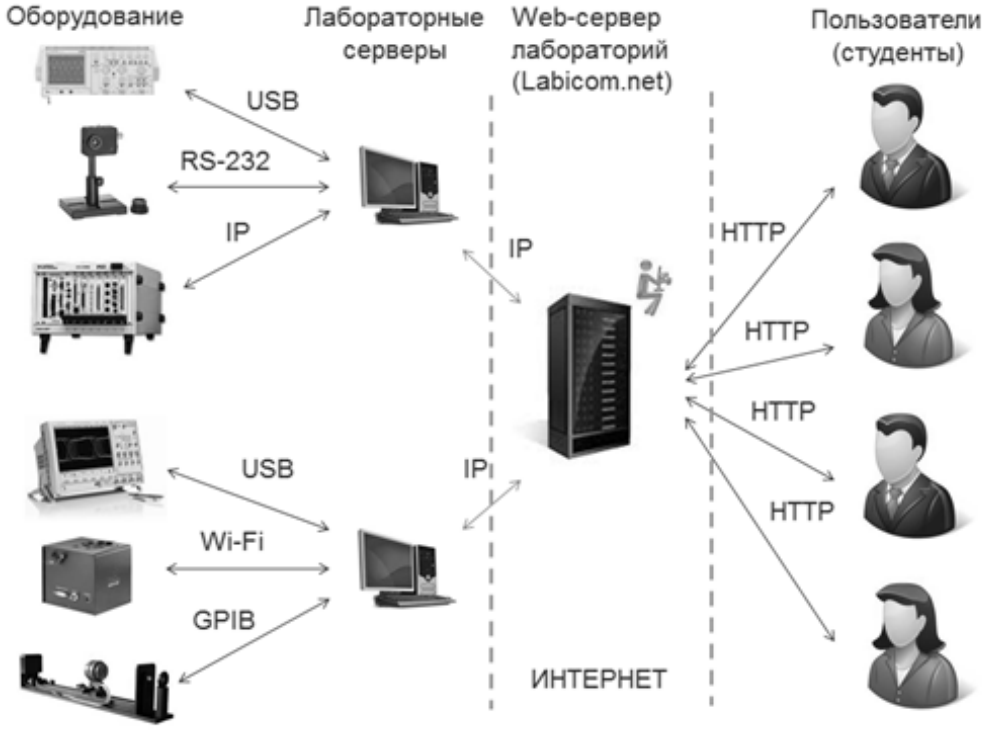


Рисунок 1. Типичная архитектура обмена данными в удалённых лабораториях

Чтобы повысить эффективность обмена данными HTTP(S) в реальном времени, можно использовать набор техник под собирательным названием Comet. Comet включает в себя Streaming, Hidden iframe, XMLHttpRequest, Long Polling Ajax и Long Polling XHR. Они сводятся к тому, что сервер не закрывает соединение, а ждёт обновления информации. Соединение закрывается браузером по наступлению timeout. Данный подход позволяет улучшить эффективность протокола HTTP(S) при обновлении данных удалённой лаборатории, однако не решает проблему полностью. Более того, перечисленные выше техники Comet сложны в реализации и не имеют одинаковой поддержки во всех веб-браузерах.

Использование протокола web socket для передачи динамических данных

В интернет-приложениях (RIA – Rich Internet Application) в настоящее время всё более распространённым становится протокол web socket [6]. Достоинством данного протокола является двусторонняя дуплексная коммуникация между клиентом и сервером. В отличие от HTTP(S), web socket не требует выполнения запроса от клиента к серверу и не содержит такого заголовка (HEADER). Таким образом, web socket является более эффективным, чем HTTP(S) с точки зрения потребляемого Интернет трафика. Web socket handshake (запрос и ответ при установлении соединения) является HTTP-совместимым, что позволяет использовать открытые по умолчанию порты 80 и 443.

Однако на практике использование порта 80 (протокола HTTP) оказывается невозможным, т.к. маршрутизаторы и proxy-серверы распознают трафик, не являющийся HTTP после завершения «рукопожатия» (handshake). В свою очередь, через 443 порт (HTTPS) после «рукопожатия» данные передаются в зашифрованном виде (SSL), и у proxy-серверов нет возможности блокировать трафик удалённой лаборатории по признаку передаваемых данных. По этой причине, а также по причине обеспечения безопасности в удалённых лабораториях платформы Labicom.net для передачи динамических данных всегда используется 443 порт.

Использование протокола WSS (web socket secure) в удалённых лабораториях позволяет передавать экспериментальные данные от лабораторного сервера клиенту с минимальной задержкой и максимальной эффективностью с точки зрения потребляемого трафика.

Использование нескольких специализированных серверов для передачи разнородных данных

На время лабораторной сессии (подключения клиента к лабораторному серверу и работы с экспериментальным оборудованием) предлагается использовать три различных сервера и три различных протокола передачи данных (рисунок 2). При таком построении программной и сетевой архитектуры удалённой лаборатории веб-сервер состоит из трёх физически различных серверов: сервера статических данных, сервера динамических данных и медиа сервера для транслирования потокового видео и других медиа ресурсов.

Сервер статических данных осуществляет организацию обмена данными по протоколу HTTP(S) как от лабораторного сервера к клиенту, так и от клиента к серверу. Так как пользовательский интерфейс (HTML/CSS/JavaScript) доставляется через данный протокол, то имеет смысл сделать сервер статических данных ведущим (Master), а остальные сервера, которые организуют обмен других типов данных, ведомыми (Slave).

Благодаря такому разделению организация серверов будет повторять организацию RIA, что упрощает процесс разработки. Главным достоинством такого разделения обязанностей между протоколами является их специализация – каждый из протоколов используется по своему прямому назначению, используя все свои сильные стороны при работе со своими данными. При этом рассматривая WSS как протокол транспортного уровня, становится возможным поместить в него другие протоколы (такие как протоколы для передачи потокового видео), используя открытый HTTPS порт 443.

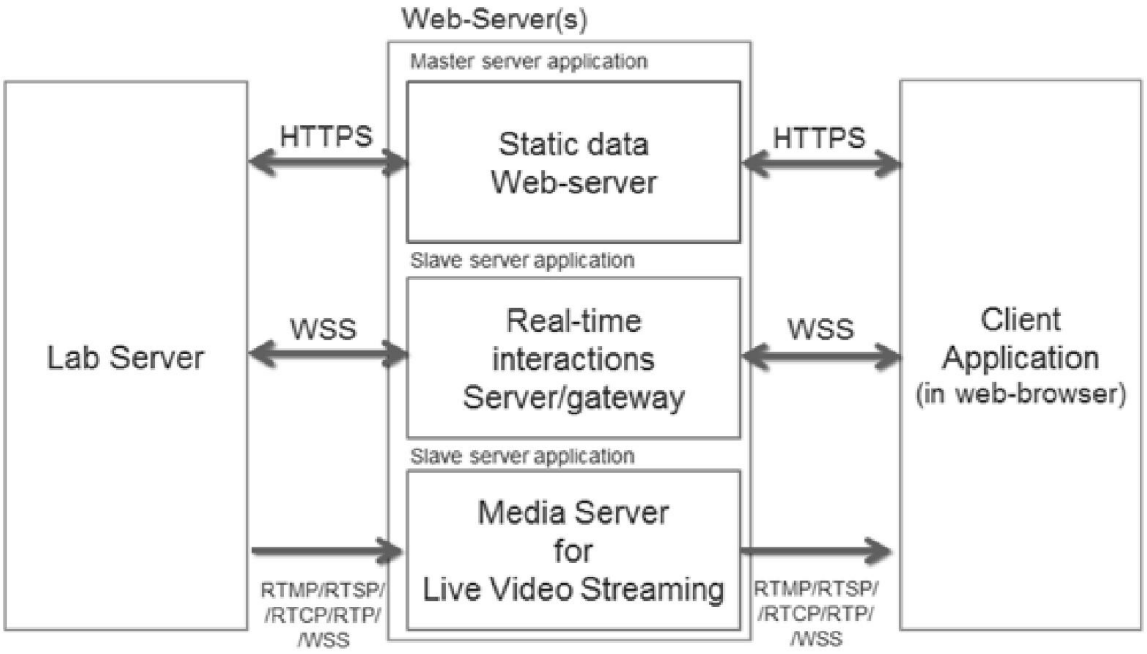


Рисунок 2. Три слоя веб-сервера удалённых лабораторий

Выводы

Рассмотренная в данной статье виртуализация нескольких разных каналов передачи данных и протоколов в один виртуальный канал, использующий стандартный порт 443, позволяет обойти все ограничения, существующие в сетях удалённых лабораторий. Данное решение показывает свою высокую эффективность с точки зрения количества передаваемых данных, задержек при передаче данных и простоты разработки лабораторного сервера и клиента. Показано, что единственным пригодным портом для нормального функционирования удалённой лаборатории является порт HTTPS 443, используемый WSS, который в свою очередь может быть использован в качестве протокола транспортного уровня для всех остальных протоколов.

Литература

[1] I. Titov, E. Smirnova "Network architectures of remote laboratories: Proposal of a new solution and comparative analysis with existing ones" // International Journal of Online Engineering (IJOE), Volume 9, Issue 6, November 2013, pp. 41-44 (http://dx.doi.org/10.3991/ijoe.v9i6.3142)

[2] I. Titov, O. Smirnova, A. Glotov and A. Golovin, "Remote Laser Laboratory: lifebuoy for laser engineering curriculum (a safe and cost-effective solution for hazardous lab environment)" // International Journal of Online Engineering (IJOE), May 2012, vol. 8, issue 2, pp. 23-27 (http://dx.doi.org/10.3991/ijoe.v8i2.1761).

[3] I. Titov, "Labicom.net – The on-line laboratories platform: optimal solution for deploying cost-effective remote and virtual laboratories" // IEEE Xplore, Global Engineering Education Conference (EDUCON), 2013 IEEE, Berlin, 13-15 March 2013, PP. 1137-1140 (http://dx.doi.org/10.1109/EduCon.2013.6530251)

[4] Network Working Group [электронный ресурс] / IETF Request for comments, "Hypertext Transfer Protocol HTTP 1.1" - 1999. - Режим доступа: http://tools.ietf.org/search/rfc2616, свободный (дата обращения: 01.12.2013).

[5] Network Working Group [электронный ресурс] / IETF Request for comments, "HTTP Over TLS" - 2000. - Режим доступа: http://tools.ietf.org/search/rfc2818, свободный (дата обращения: 01.12.2013).

[6] Network Working Group [электронный ресурс] / IETF Request for comments, "The WebSocket Protocol" - 2000. - Режим доступа: http://tools.ietf.org/html/rfc6455, свободный (дата обращения: 01.12.2013).

УДК 004.021

МОДИФИКАЦИЯ МЕТОДА РУТИСХАУЗЕРА ДЛЯ РАЗБОРА МАТЕМАТИЧЕСКИХ ВЫРАЖЕНИЙ, СОДЕРЖАЩИХ ФУНКЦИИ

А.К. Халайджи, Г.С. Иванова

Ключевые слова: разбор, выражение, метод Рутисхаузера, функция.

Key words: parsing, expression, Rutishauzer's method, function.

Аннотация: Проведён анализ недостатков классического метода Рутисхаузера применительно к задаче разбора математических выражений, содержащих функции.

Предложена модификация рассмотренного метода, позволяющая решать поставленную задачу. Проведён сравнительный анализ классического и модифицированного метода Рутисхаузера с классическим алгоритмом Бауэра-Замельзона. Принцип работы модифицированного метода показан на примере.

При разработке программного обеспечения достаточно часто выполняется разбор математической записи различных выражений. Такая задача, например, ставится при анализе SQL-запросов, при создании пакетов экономических программ и при написании компилирующих или транслирующих программ.

В основном для разбора выражений применяют алгоритмы, использующие обратную польскую запись (ОПЗ), или их модификации [1]. Эти алгоритмы базируются на теории грамматик и реализуют обобщённый механизм разбора выражений: не только математических, но и конструкций языков программирования.

Помимо указанных алгоритмов разбора математических выражений существует не менее известный метод Рутисхаузера [2]. Он позволяет вычислять значение корректно сформированного математического выражения, записанного в полной скобочной форме. Разбираемое данным методом выражение может содержать только простейшие арифметические операции: +, -, *, /, собственно операнды и скобки.

Процедура приведения выражения к полной скобочной форме не представляет особой сложности. К тому же существуют готовые алгоритмы и реализации на различных языках, позволяющие это сделать. Одним из таких алгоритмов является алгоритм, реализующий метод Вогана-Пратта [3].

При разборе математического выражения классическим методом Рутисхаузера также может возникнуть проблема, связанная с унарными операциями «+» и «-«. В этом случае необходимо предварительно перед составлением полной скобочной формы записи данного выражения добавить ко всем унарным операциям левый недостающий операнд 0. Тогда все операции станут бинарными. Так, например, «- 5» => «0 – 5». Данная процедура также не является сложной, так как существует ограниченное количество допустимых вариантов использования унарной операции: она может начинаться с начала выражения или после открытой скобки.

Хотя ранее метод применялся в компиляторах ранних языков программирования, в настоящее время в литературе отсутствует описание алгоритмов, реализующих данный метод для разбора сложных математических выражений. Это связано с несколькими причинами:

1) рассматриваемый метод изначально был разработан для простейших математических выражений, которые были небольшими и содержали только простейшие арифметические операции;

2) по сравнению с методами, использующими грамматику, рассматриваемый метод имеет худшую надёжность, то есть способность обнаружения ошибок в выражении, поскольку взаимосвязь между символами выражения не учитывается.

Тем не менее у метода Рутисхаузера есть преимущества по сравнению со стековыми методами. К ним можно отнести то, что:

1) метод самостоятельный, то есть не использует внешних понятий, таких как грамматика, следовательно проще для понимания;

2) во многих случаях разбора арифметических выражений без функций может показывать большую эффективность, чем стековые алгоритмы, так как отсутствует необходимость вести дополнительные стеки.

Область применения классического метода Рутисхаузера – простейшие математические выражения с числами, скобками и арифметическими операциями. Поэтому рассматриваемый метод не подходит для анализа выражений, содержащих функции. Так, например, при попытке нумерации символов строки «F(P)» получается, что в конце выражения – не 0, а 1, что свидетельствует о том, что выражение некорректно. Это связано с тем, что при нумерации символ F считается как операнд, а не как функция.

В дальнейшем предполагается, что метод применяется к строке токенов, полученной на этапе лексического анализа. При этом выражение приведено к полной скобочной форме, а все унарные операции приведены к бинарным.

В процессе синтеза модифицированного метода рассматривались функции без параметров, с одним и с двумя параметрами. Соответственно, в строке токенов они отображались как: F(), F(<параметр>), F(<параметр1>;<параметр2>).

Основной точкой отправления служило создание новых типов "троек" и их свёртки. Так, если встречается комбинация 'F(', то индекс скобки совпадает с индексом символа функции, так как не может быть символа функции без скобки. Если встречается последовательность '<параметр1>;<параметр2>', то у символа ';' индекс такой же, как и у обоих параметров, так как, если необходимо передать несколько параметров, то данный символ обязателен. Таким образом, исследуя всевозможные комбинации, можно выделить несколько групп «троек»:

1) n:n:n+1:n - F(<параметр>). Например, sin(a);

2) n+1:n:n+1 - <операнд 1><операция><операнд 2>. Например, a+b;

3) n-1:n:n+1:n - (<значение>). Например, (a) - первый параметр необходим для того, чтобы отличать данную тройку от тройки первого типа;

4) n-1:n-1:n:n:n:n-1 - F(<параметр1>;<параметр2>). Например, log(n;a);

5) n-1:n:n:n - F(). Например, e() - исключение (у второй скобки индекс тот же).

Использование таких вариантов «троек» позволяет анализировать математические выражения с функциями. Следует отметить, что он не эффективен при наличии функций с большим количеством параметров, потому что выражение "F(<параметр 1>;<параметр 2>;...;<параметр n>) будет иметь следующие индексы: n-1:n:n:n:n:.....:n:n-1, что естественно не эффективно, хотя и позволяет анализировать выражения при условии, что они составлены верно.

Видно, что во всех вариантах есть характерные черты, а также, что первый индекс блока каждого типа совпадает с последним (кроме последнего), то есть вставка или удаление одного такого блока не влияет на остальную нумерацию «тройками». Это показывает то, что, если выражение корректно составлено, то его можно разбить на такие блоки. Анализ следует начинать так же с поиска максимального индекса, но в отличие от классического метода необходимо анализировать сложные «тройки», что является «узким» местом алгоритма, так как данная операция не является очевидной с точки зрения реализации.

Перебором всех возможных вариантов можно показать, что не существует других троек. При этом, как и в классическом методе, если не удаётся проверить какую-то комбинацию, необходимо пропустить её и анализировать следующую. Перед тем, как находить и сворачивать «тройки», как в классическом методе, необходимо избавиться от неоднозначности, которая проявляется в том, что тройка пятого типа полностью совпадает со внутренностью тройки четвёртого типа, но такой не является. Поэтому тройки пятого типа необходимо все свернуть до начала работы основной части алгоритма. Свёртка таких функций происходит просто, потому что это функции без параметров, то есть в строке токенов всегда отображаются одинаково: "F()".

После удаления всех троек пятого типа необходимо найти первый элемент i с максимальным индексом n. Необходимо проверить i+1 элемент. Если он также имеет индекс n, то это может быть либо ошибка, либо тройка четвёртого типа. Для проверки на наличие ошибки необходимо проверить все индексы, образующие эту тройку, а при свёртке проконтролировать, что данная функция может иметь два параметра. (потому что, например, выражение «sin(x;y)» некорректно, когда «log(x;y)» корректно).

Если индекс элемента i+1 не равен n, то он не может быть равен n+1 (т.к. максимальный индекс в строке n), поэтому он может быть равен только n-1. Тогда

необходимо проверить i+2 элемент. Если его индекс равен n, то тем самым этот элемент замыкает тройку-операцию. Если он равен n-1, то это сигнал об ошибке, потому что тогда получается тройка вида: «n:n-1:n-1», где в качестве «n-1:n-1» может быть или «<параметр>;», или «F». Но так как и параметр, и функция являются символами, которые не уменьшают значение счётчика, то они не могут быть в строке, потому что тогда бы на i-ом месте должен бы стоять индекс n-2, а там находится индекс n, то есть индекс уменьшается от i-го до (i+1)-го элемента. Если индекс (i+2)-го элемента равен n-2, то это ни о чём явно не говорит (например, это может быть последовательность закрывающихся скобок). Поэтому необходимо проверить i-1 элемент. Его индекс не может быть равным n, потому что тогда бы (i-1)-й элемент был бы первым максимальным элементом строки. По той же причине, его индекс не может быть больше, чем n. Поэтому единственным возможным значением индекса является n-1. Видно, что теперь выстроилась тройка, которая есть как в первом типе, так и в третьей. Поэтому необходимо проверить также и (i-2)-й символ для того, чтобы точно определить, к какой группе «троек» относится данная «тройка». Если индекс этого элемента n-1, то это функция с одним аргументом. При свёртке необходимо проверить, допустим ли у этой функции только один параметр. Если нет – то это сигнал об ошибке. Если же индекс равен не n-1, то он не может быть равным n, так как i-й элемент является первым элементом с максимальным индексом. Поэтому он может быть равен только n-2. Тогда i-1:i:i+1 элементы образуют тройку-скобки.

Таким образом, можно свести весь алгоритм к символьному виду:

1) F() (e(); sin()->ОШИБКА!);

2) Max индекс n – i-й символ;

3) (i+1)-> n? => F(a;b) (log(a;b); sin(a;b)->ОШИБКА!);

4) (i+1)->n-1; (i+2)->n? => операция;

5) (i+1)->n-1; (i+2)->n-2; (i-1)->n-1; (i-2)->n-2? => скобки;

6) (i+1)->n-1; (i+2)->n-2; (i-1)->n-1; (i-2)->n-1? => F(a) (sin(a); log(a)->ОШИБКА!).

Пример. Пусть дана строка токенов в полной скобочной форме: (F((P*F())/F(P;P)))

Номер шага																					
1		(	F	(	(	P	*	F	(	)	)	/	F	(	P	;	P	)	)	)	
	0	1	2	2	3	4	3	4	4	4	3	2	3	3	4	4	4	3	2	1	0
2		(	F	(	(	P	*	C	)	/	F	(	P	;	P	)	)	)			
	0	1	2	2	3	4	3	4	3	2	3	3	4	4	4	3	2	1	0		

3			(	F	(	(	C	)	/	F	(	P	;	P	)	)	)
	0	1	2	2	3	4	3	2	3	3	4	4	4	3	2	1	0
4			(	F	(	C	/	F	(	P	;	P	)	)	)		
	0	1	2	2	3	2	3	3	4	4	4	3	2	1	0		
5			(	F	(	C	/	C	)	)							
	0	1	2	2	3	2	3	2	1	0							
6			(	F	(	C	)	)									
	0	1	2	2	3	2	1	0									
7			(	C	)												
	0	1	2	1	0												
8 - результат			C														
	0	1	0														

Таким образом, предлагаемый модифицированный алгоритм Рутисхаузера позволяет без существенного изменения сложности алгоритма анализировать математические выражения, содержащие функции. Однако модификация не решает проблем с обнаружением ошибок. Как и в классическом методе, недостатком также остаётся необходимость приведения исходного выражения к полной скобочной форме перед непосредственным использованием алгоритма.

Литература

1. Илья Кантор. Обратная польская нотация // ALGOLIST.RU:Алгоритмы. Методы. Исходники. 2000 – 2013.URL.http://algolist.ru/syntax/revpn.php(дата обращения: 31.12.2013).

2. Илья Кантор. Алгоритм Рутисхаузера // ALGOLIST.RU:Алгоритмы. Методы. Исходники. 2000 – 2013.URL. http://algolist.ru/syntax/parsear.php (дата обращения: 31.12.2013).

3. Prett Parsing – метод Вогана Пратта для разбора выражений // HABRAHABR.RU:2009.URL. http://habrahabr.ru/post/50349/ (дата обращения 31.12.2013).

УДК 001.87

ПРОГРАММНО-АППАРАТНЫЙ КОМПЛЕКС ДЛЯ РЕАЛИЗАЦИИ ЛАБОРАТОРНЫХ ПРАКТИКУМОВ ПО КУРСУ «СХЕМОТЕХНИКА ЭВМ»

А.А. Чибисов, Е.А. Яковенко, А.С. Черников

Ключевые слова: Лабораторный практикум, моделирование, удаленный доступ, схемотехника

Key words: Laboratory workshop, modeling, remote access, circuitry

Аннотация: Рассматриваются проблемы проведения лабораторных практикумов по техническим дисциплинам. Описывается разработанный программно-аппаратный комплекс, совмещающий в себе возможности физического моделирования электрических схем и удаленного доступа к ним. Сравнивается реализация схем без и с использованием ПЛИС. Рассматриваются возможности и преимущества удаленного доступа к управлению схемами, перспективы развития разработанного комплекса.

В рамках учебного процесса учебных заведений для лучшего усвоения материала студентами требуется обеспечить возможность проведения практических занятий по изучаемым техническим дисциплинам. На деле организация таких лабораторных работ требует определенных материальных и временных затрат.

Здесь возникает и ряд других проблем. Из-за быстрого совершенствования техники, созданные лабораторные работы, а чаще их аппаратная часть, устаревают и перестают заинтересовывать студентов. Также, интенсивное развитие сетевых технологий сделало возможным дистанционный доступ студентов к различным курсам, что в перспективе может привести к переводу большей части образовательного процесса на дистанционный режим. Решить эти и многие другие задачи позволяет разработанный программно-аппаратный комплекс.

Обычно, все лабораторные практикумы относятся к одному из двух видов:

– готовый макет лабораторной работы, когда от студента требуется пошагово выполнить работу, предложенную в методических указаниях. Минусом таких макетов является их неуниверсальность, невозможность их усовершенствования и трудоемкость ремонта;

– моделирование, когда не требуется никаких аппаратных затрат, но при этом проделанная работа не отражает реальных процессов в собранных схемах. При данном подходе невозможно оценить все аспекты функционирования реальной схемы, и

интерес со стороны студента здесь минимальный, поскольку нельзя «пощупать» результат.

Совмещение обоих видов лабораторных работ – основная идея данного проекта. Безусловно, это требует аппаратных затрат, однако можно выявить и ряд плюсов:

– выполненная работа может быть модернизирована частично или под совершенно другую работу;

– используемая элементная база может быть заменена (по причине поломки или при необходимости использования других элементов);

– появляется возможность сравнить данные, полученные при моделировании и физической реализации, в рамках одного проекта;

На данный момент многие вузы и специализированные Интернет-порталы во всем мире предлагают воспользоваться услугами онлайн-курсов, которые обычно включают в себя видеозаписи лекций и выкладки теоретических материалов в виде текстовых документов или презентаций. Однако, для изучения многих технических дисциплин этого не достаточно, так как необходимо не только проработать теорию, но и получить некоторые практические навыки, пройдя серию лабораторных работ. Особенностью разработанного комплекса является обеспечение возможности удаленного выполнения таких работ. Таким образом, можно выделить дополнительные преимущества комплекса:

– реализация удаленного доступа позволит сэкономить время преподавателям, а студентам выполнять практикум в удобное им время и из удобного места;

– также удаленный доступ разгрузит лабораторные классы (снижается необходимое количество оборудования, и выполнение работ не привязано к общему расписанию студентов);

– реализация анализа данных по успеваемости и связи комплекса с электронными системами университетов позволит автоматизировать процесс оценки успеваемости студентов.

В качестве основы для разработки практикума был выбран курс «Схемотехника ЭВМ», а в рамках этого курса – лабораторные работы на темы «Счетчик» и «Регистр».

В качестве аппаратной основы комплекса используется учебная станция NI ELVIS II от компании National Instruments. NI является одним из мировых лидеров в технологии виртуальных приборов и в разработке и изготовлении аппаратного и программного обеспечения для систем автоматизированного тестирования. Платформа NI ELVIS II представляет собой настольную лабораторную станцию для подключения к ПК с графической средой программирования LabVIEW. Объединение этих компонентов в рамках единой системы делает NI ELVIS II мощной и гибкой контрольно-измерительной платформой [1]. Для

создания схем на реальной элементной базе используются подсоединяемые к ELVIS макетные платы (с наборным полем или со встроенной ПЛИС).

При разработке проекта решалась проблема управления работой собираемых схем как в статическом, так и в динамическом режимах, и наблюдения результатов их функционирования. Для создания практикума на макетной плате были выбраны элементы серий КР1533 и К555. Взаимодействие со схемой в статическом режиме обеспечивает программное обеспечение, созданное в среде LabVIEW [2]. С его помощью можно управлять работой собранной схемы в режиме реального времени, используя подключенный к учебной станции компьютер. Для наблюдения функционирования схемы в динамическом режиме (определение временных задержек распространения сигналов, возможных гонок сигналов) используются возможности среды Multisim [1, 3]. В нее встроена возможность эмулирования учебной станции NI ELVIS II, при этом можно управлять приборами (генератор, осциллограф и т.д.), расположенными на реальной, подключенной к компьютеру станции.

При создании программно-аппаратного комплекса были изучены два возможных подхода к созданию лабораторных работ. Кроме построения схем лабораторного практикума на обычном наборном поле, где требуется физически размещать все необходимые для функционирования схемы элементы, был реализован вариант с использованием Программируемой Логической Интегральной Схемы (ПЛИС). Для учебной станции NI ELVIS II существует специальная макетная плата (DE FPGA Board) с наборным полем и встроенной ПЛИС типа FPGA [4]. Для программирования этой ПЛИС используется специальный программный модуль LabVIEW FPGA, который дает возможность применить ПЛИС для создания схем лабораторного практикума. Использование ПЛИС имеет большое преимущество по сравнению со стандартной сборкой схем на макетной плате: все компоненты, используемые в комбинационных схемах (логические элементы), а также коммутацию различных выводов микросхем между собой можно реализовать программным способом в модуле LabVIEW FPGA и затем загрузить созданную программу в ПЛИС. Таким образом, ПЛИС будет управлять работой только подключенными к плате DE FPGA Board микросхемами. Это позволяет значительно упростить построение схем на реальных физических элементах. Но при этом имеется и недостаток: процессы, протекающие в элементах, реализованных на ПЛИС, не будут соответствовать реальным процессам в физических элементах. Кроме того, для некоторых схем (например, описывающих функционирование регистров) практически не требуется дополнительных комбинационных элементов, т.о., преимущество использования ПЛИС в таких случаях исчезает. Дополнительно к этому, усложняется

разработка программного обеспечения для управления собираемыми схемами. Таким образом, можно сделать вывод, что в лабораторных практикумах целесообразнее использовать обычные наборные платы.

Использование среды LabVIEW для создания программного обеспечения по управлению схемами лабораторного практикума позволяет обеспечить дистанционное взаимодействие обучающегося и предлагаемых схем. Разработанные в LabVIEW приложения легко интегрируются с веб-сайтами образовательных учреждений, таким образом, что студент, получив доступ к необходимой странице, видит перед собой интерфейс приложения и может в режиме реального времени управлять и получать результаты работы схемы лабораторного практикума, физически собранной и подключенной к компьютеру, расположенному в лаборатории. В данном случае сохраняется возможность работы с реальными устройствами, при этом требуется минимальное количество рабочих макетов и специальных помещений для обеспечения учебного процесса.

Использование разработанного комплекса для проведения лабораторных практикумов возможно в двух вариантах:

– полностью дистанционное обучение – студент работает удаленно с заранее собранной в лаборатории схемой;

– разделение выполнения лабораторной работы на два этапа – студент лично собирает схему на макетной плате и тестирует правильность ее работы; затем, уже удаленно, проводит дополнительные исследования ее функционирования (снятие временных характеристик и т.п.)

Таким образом, данный проект может дать следующие преимущества, как для студента, так и для преподавателя:

– возможность создания функционального узла своими руками на основе элементной базы;

– проведение лабораторных работ с разделением на этапы: в условиях лаборатории (конструкция стенда) и в домашних условиях (разработка макета, получение результатов);

– возможность использования уникального оборудования;

– возможность применения в рамках данного комплекса методов электронного обучения;

– возможность быстрой модернизации существующих и создания новых лабораторных работ.

Реализация удаленного доступа добавляет еще ряд достоинств:

– эффективное использование оборудования, так как лаборатории разгрузятся;

– удобство работы студентов и исследователей с точки зрения планирования времени;

– возможность автоматизации анализа данных по успеваемости и связи комплекса с информационной системой университета контроля текущей успеваемости;

– возможность реализации совместных учебных программ и НИР с вузами как российскими, так и зарубежными;

– объединение комплекса с онлайн-лекциями и системами контроля знаний позволяет получить полноценный дистанционный обучающий курс по технической дисциплине;

Кроме всего выше изложенного, при использовании специального оборудования для автоматизации процесса сборки схем на наборных полях макетных плат, появится возможность как обеспечения проведения большого числа различных лабораторных работ при использовании минимума оборудования (учебных станций NI ELVIS II), так и удаленного управления сборкой схем из реальных элементов для лабораторных или исследовательских работ.

Литература

1. NI ELVIS II. Учебный курс. [Электронный ресурс]. – URL: http://www.ros-group.ru/content/data/store/images/f_3484_26925_1.pdf (дата обращения: 20.12.2013).

2. LabVIEW. Вводный курс. [Электронный ресурс]. – URL: http://www.novsu.ru/file/1065868 (дата обращения: 20.12.2013).

3. Введение в Multisim. Трехчасовой курс. [Электронный ресурс]. – URL: http://www.all-library.com/obrazovanie/uchebnye-posobiya/49330-vvedenie-v-multisim-trexchasovoj-kurs.html (дата обращения: 20.12.2013).

4. Отладочная плата NI Digital Electronics FPGA Board. Руководство по эксплуатации. [Электронный ресурс]. – URL: ftp:// ftp.ni.com/pub/branches/russia/ni_elvis/prototyping_board_with_fpga_design.pdf. (дата обращения: 20.12.2013).

УДК 004.891

ПРИМЕНЕНИЕ МЕТОДОВ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА В ЗАДАЧАХ РАЗВИТИЯ И СОВЕРШЕНСТВОВАНИЯ ТВОРЧЕСКОГО МЫШЛЕНИЯ

Э.Н. Голубева, Е.К. Пугачев

Введение

В наше время очень популярны методики, которые помогают развивать и совершенствовать навыки мышления. В связи с тем, что думать и принимать решения людям всех профессий, специальностей и возрастов приходится ежедневно, повышение эффективности использования данных методик является актуальной задачей.

Чтобы освоить ту или иную методику совершенствования мышления, приходится перечитывать большое количество литературы, вникать в предложенный метод и стараться применить его в соответствии с рекомендациями автора. Важно отметить, что процесс освоения методики носит итерационный характер. Связано это с тем, что часто приходится уточнять те, или иные положения с целью более детальной проработки методики.

В связи с вышесказанным, важной задачей является создание программного комплекса, в котором реализованы различные положения методов, позволяющие совершенствовать навыки мышления пользователя.

Одна из известных методик развития мышления описана в работе «Шесть шляп мышления» Эдварда де Боно. Она интересна тем, что предлагает рассмотрение вопроса с разных точек зрения.

В данной статье для реализации программного комплекса предлагается использовать методы систем искусственного интеллекта. В частности, предлагается подход к созданию экспертной системы, позволяющей консультировать и выдавать рекомендации для пользователей на основе метода «Шесть шляп мышления».

Цель и задачи исследования

Цель исследования – получить все необходимые данные, которые позволили бы создать модель экспертной системы для метода «Шесть шляп мышления» с целью более быстрого изучения и приобретения навыков по применению описанного метода.

Задачи исследования:

- детально проработать метод Эдварда де Боно;
- выделить особенности каждой шляпы мышления, т.е. определить в какой ситуации требуется использовать каждую шляпу и определить очередность использования каждой из шляп;

- оценить применимость известных методов искусственного интеллекта к данной предметной области.

Основные положения метода «Шесть шляп мышления»

Основой метода «Шесть шляп мышления» является шесть сторон рассмотрения вопроса, что обеспечивает всестороннее изучение поставленной задачи.

Определенную сторону изучения поставленной задачи характеризует «одетая» на частника рассмотрения шляпа. Всего шляп шесть. Каждая из них имеет свои особенности.

1. Белая шляпа.

Участник обсуждения, находящийся «в белой шляпе» не должен выказывать свои чувства или эмоции, а должен рассматривать вопрос с объективной точки зрения.

В этой шляпе имеет место рассмотрение только фактов, сводок, проверенных исследований и повторяющихся закономерностей.

2. Красная шляпа

«Надевший» эту шляпу высказывает все, что он чувствует, что его тревожит в рассматриваемой проблеме.

Он может озвучивать как опасения, так и хорошие предчувствия. Приветствуется высказывание интуитивной информации, так как эта шляпа не только эмоций, но и интуиции.

Высказанную в этой шляпе информацию можно не аргументировать, так как природа наших чувств иногда необъяснима.

Эта шляпа используется при возникновении споров или при окончательном рассмотрении вопроса.

3. Черная шляпа

Как и обладатель белой шляпы, участник в черной шляпе должен проявлять точность в изложении фактов, но точность эта должна доходить до въедливости. Это шляпа критики и негативного рассмотрения всех сторон вопроса.

При высказывании отрицательных качеств рассматриваемой проблемы не нужно чувственное рассмотрение вопроса, а лишь аргументированные выводы.

4. Желтая шляпа

«Шляпа радужных настроений», как называет ее сам Эдвард де Боно.

Участник обсуждения, надевший желтую шляпу, при рассмотрении вопроса должен искать положительные его стороны, воспринимать их как благоприятные возможности для дальнейшего осуществления.

Эта шляпа – шляпа энтузиазма и оптимизма.

5. Зеленая шляпа

Зеленая шляпа- шляпа творческого начала.

Участник обсуждения, находящийся в «зеленой шляпе» настроен на поиск новых решений и путей развития проблемы. Приветствуются любые неординарные мысли и решения.

Для творческого рассмотрения проблемы существует огромное количество разнообразных методик, которыми может пользоваться обладатель «зеленой шляпы».

6. Синяя шляпа

Участник в «синей шляпе» играет управляющую функцию, осуществляет контроль и дисциплину происходящего процесса мышления.

Он следит за течением обсуждения, разрешает конфликтные ситуации, предлагая участникам «надеть» красные шляпы и высказаться, не доводя до ссор и разбирательств. Также он формулирует задачу обсуждения, может изменять ее, также ставит задачи и области для творческого рассмотрения, формирует перечень идей.

Кроме того, обладатель «синей шляпы» подводит итоги всего обсуждения и выбирает окончательную стратегию решения поставленного вопроса.

Все участники обсуждения могут «надеть» ту или иную шляпу в зависимости от состояния обсуждения решаемой проблемы. Совершенно необязательно, что в шляпе одного цвета должен находиться только один участник, их может быть и несколько, в зависимости от ситуации. Недопустимо, чтобы участник обсуждения находился все время в шляпе одного цвета, так как это означает однобокость рассмотрения вопроса и противоречит данному методу.

Некоторые рекомендации по использованию шляп:

- Рекомендуется рассмотрение вопроса не начинать с черной шляпы.
- Перед рассмотрением вопроса в зеленой шляпе, рассмотреть его с позиции белой шляпы и высказать все, что уже сделано или обычно делается при решении аналогичных задач.
- При возникновении споров или предчувствий следует воспользоваться красной шляпой, но контролировать критическую ситуацию следует обладателю синей шляпы.
- Обладатель синей шляпы должен заносить все возникающие идеи в перечень идей, с целью дальнейшего их обсуждения.
- Начинать следует с постановки задачи обсуждения, формулирование цели обсуждения - задача обладателя синей шляпы.

Отдельно следует отметить последовательность действий при рассмотрении сформированного перечня идей:

1) Выбрать одну предложенную идею;
2) Рассмотреть идею с позиции обладателя желтой шляпы;
3) Рассмотреть идею с позиции обладателя белой шляпы;
4) Рассмотреть идею с позиции обладателя зеленой шляпы;
5) Рассмотреть идею с позиции обладателя черной шляпы;
6) Исправить недочеты, выявленные в состоянии черной шляпы, с использованием желтой и зеленой шляп;
7) Отбор идей с использованием желтой и черной шляп;
8) Выработка окончательной стратегии при помощи синей шляпы.

При возникновении споров требуется воспользоваться синей шляпой – для контролирования критической ситуации, и красной шляпой – для высказывания эмоций.

Разработка модели экспертной системы метода «Шесть шляп мышления»

На основе рекомендаций и характеристик, изложенных в книге «Шесть шляп мышления», были определены особенности модели экспертной системы.

Цель такой экспертной системы – выработка окончательной стратегии по решаемому вопросу.

Эту цель можно разбить на ряд подцелей:

1. Составление перечня идей для обсуждения и выбора;
2. Рассмотрение идей;
3. Исправление недочетов идей;
4. Отбор идей
5. Разрешение разногласий.

Подцели 1-4 должны быть выполнены по порядку, их выполнение обязательно для достижения цели.

Подцель 5 может не выполняться, если при обсуждении вопроса разногласий не возникло.

При дальнейшем разбиении можно получить:

1. Составление перечня идей:
   - 1.1 Постановка задачи
     - 1.1.1 Обсуждение проблемы
     - 1.1.2 Формулирование задачи (синяя шляпа)
   - 1.2 Возникновение идеи

1.3 Рассмотрение идеи (одиночной)

1.3.1 С положительной стороны (желтая шляпа)

1.3.2 С отрицательной стороны (черная шляпа)

1.3.3 Рассмотрение фактов (белая шляпа)

1.3.4 Использование творческого подхода

1.3.4.1 Формулирование задач для творчества (синяя шляпа)

1.3.4.2 Рассмотрение творческих аспектов идеи (зеленая шляпа)

1.3.4.3 Высказывание новых идей (зеленая шляпа)

1.4 Разрешение разногласий

1.4.1 Определение роли управляющего (синяя шляпа)

1.4.2 Высказывание чувств (красная шляпа)

2. Рассмотрение идей (из перечня идей)

2.1 С положительной стороны (желтая шляпа)

2.2 Рассмотрение фактов (белая шляпа)

2.3 Рассмотрение с творческой стороны (зеленая шляпа)

2.4 С отрицательной стороны (черная шляпа)

Пункт 2.3 можно разбить аналогично пункту 1.3.4.

3. Исправление недочетов идей

3.1 Использование недочетов так, чтобы они стали преимуществами (желтая шляпа)

3.2 Использование недочетов как основы для творчества (зеленая шляпа)

4. Отбор идей

4.1 Отбор по положительным качествам (желтая шляпа)

4.2 Отбор по отрицательным качествам (черная шляпа)

4.3 Систематизация результатов (синяя шляпа)

5. Разбиение подцели 4 аналогично разбиению 1.4.

Наименьшей единицей разбиения, в представленном выше способе разбиения, является состояние нахождения в шляпе определенного цвета. При этом, каждое такое состояние должно сопровождаться пояснениями и рекомендациями.

Каждую подцель удобно рассматривать как фрейм, с количеством слотов, равных количеству частей подцели. При этом, в некоторых случаях нужно учитывать очередность выполнения подзадач (означивания слотов фрейма).

Некоторые из слотов могут не выполняться – например, такие, как «разрешение разногласий», потому что в ходе обсуждения, может не возникнуть конфликтных ситуаций.

Таким образом, реализацию метода «Шесть шляп мышления» Эдварда де Боно удобно рассматривать как фреймовую модель экспертной системы, предусмотрев модули накопления данных, знаний, развитый модуль объяснения.

Выводы

В данной работе были сформулированы основные правила и особенности метода «Шести шляп мышления», выявлены ключевые моменты для проектирования.

Определена структура экспертной системы для рассматриваемой предметной области. Предложена модель представления знаний, определены основные процессы механизма логического вывода, а также ряд решений связанных с реализацией сервисный модулей.

Полученные результаты могут быть использованы при реализации экспертной системы, которая помогает осуществлять процесс мышления на основе метода Эдварда де Боно.

В качестве дальнейшей проработки могут быть вопросы, связанные с проектированием модуля для зеленой – «творческой» шляпы, который бы предлагал на выбор методики творческого мышления. К таким методикам относятся биометрия, подвижные столбики, описанные в книге Виктора Папанека «Дизайн для реального мира», а также методы «шиворот-навыворот», случайно загаданное слово, описанные в книге Эдварда де Боно «Шесть шляп мышления», или многие другие методы.

Литература

1. Папанек В. Дизайн для реального мира. - Изд-во Аронов, 2008.-460с.

2. Уильямс Л. Перворот. Проверенная методика захвата рынка.- Изд-во Манн, Иванов, Фербер, 2008. - 208с.

3. Де Боно Э. Шесть шляп мышления.- Изд-во Попурри, 2006.-208с.

4. Х. Уэно, М. Исидзука. Представление и использование знаний: Пер. с япон.- М.:Мир, 1989. - 220с.

СОДЕРЖАНИЕ

СОВРЕМЕННЫЕ КОМПЬЮТЕРНЫЕ СИСТЕМЫ
И ТЕХНОЛОГИИ

Сборник трудов кафедры
«Компьютерные системы и сети»
МГТУ им. Н.Э. БАУМАНА

Подписано в печать 2.06.2014. Формат 60x84/16.

Усл. печ. л. 11,13. Тираж 100 экз.

105005, Москва, 2-я Бауманская ул., д. 5, стр. 1

Изд-во: НИИ Радиоэлектроники и лазерной техники
МГТУ им. Н.Э. Баумана