ВЕСТНИК

ЮЖНО-УРАЛЬСКОГО ГОСУДАРСТВЕННОГО УНИВЕРСИТЕТА

2017
Т. 6, № 2

ISSN 2305-9052

СЕРИЯ

«ВЫЧИСЛИТЕЛЬНАЯ МАТЕМАТИКА И ИНФОРМАТИКА»

Решением ВАК включен в Перечень научных изданий, в которых должны быть опубликованы результаты диссертаций на соискание ученых степеней кандидата и доктора наук

Учредитель — Федеральное государственное автономное образовательное учреждение высшего образования «Южно-Уральский государственный университет (национальный исследовательский университет)»

Тематика журнала:

- Вычислительная математика и численные методы
- Математическое программирование
- Распознавание образов
- Вычислительные методы линейной алгебры
- Решение обратных и некорректно поставленных задач
- Доказательные вычисления
- Численное решение дифференциальных и интегральных уравнений
- Исследование операций
- Теория игр
- Теория аппроксимации
- Информатика
- Математическое и программное обеспечение высокопроизводительных вычислительных систем
- Системное программирование
- Перспективные многопроцессорные архитектуры
- Облачные вычисления
- Технология программирования
- Машинная графика
- Интернет-технологии
- Системы электронного обучения
- Технологии обработки баз данных и знаний
- Интеллектуальный анализ данных

BULLETIN
OF THE SOUTH URAL STATE UNIVERSITY
SERIES "COMPUTATIONAL MATHEMATICS AND SOFTWARE ENGINEERING"

2017
vol. 6, no. 2

ISSN 2305-9052

Vestnik Yuzhno-Ural'skogo Gosudarstvennogo Universiteta.
Seriya "Vychislitel'naya Matematika i Informatika"

South Ural State University

The scope of the journal:

- Numerical analysis and methods
- Mathematical optimization
- Pattern recognition
- Numerical methods of linear algebra
- Reverse and ill-posed problems solution
- Computer-assisted proofs
- Numerical solutions of differential and integral equations
- Operations research
- Game theory
- Approximation theory
- Computer science
- High performance computing
- System software
- Advanced multiprocessor architectures
- Cloud computing
- Software engineering
- Computer graphics
- Internet technologies
- E-learning
- Database processing
- Data mining

Содержание

Contents

УДК 519.63 DOI: 10.14529/cmse170201

О ПОСТРОЕНИИ ДВУМЕРНЫХ ЛОКАЛЬНО-МОДИФИЦИРОВАННЫХ КВАЗИСТРУКТУРИРОВАННЫХ СЕТОК И РЕШЕНИИ НА НИХ КРАЕВЫХ ЗАДАЧ В ОБЛАСТЯХ С КРИВОЛИНЕЙНОЙ ГРАНИЦЕЙ

© 2017 г. А.Н. Козырев, В.М. Свешников

Институт вычислительной математики и математической геофизики СО РАН
(630090 Новосибирск, пр. Акад. Лаврентьева, д. 6)
E-mail: kozyrev@inbox.ru,victor@lapasrv.sscc.ru

Поступила в редакцию: 01.05.2017

Разработаны новые подходы к локальной модификации квазиструктурированных сеток, которые позволяют отследить неоднородности краевой задачи в расчетной области и адаптивны к криволинейным границам, а также просты в использовании и не требуют хранения большого объема данных, как это необходимо в неструктурированных сетках. Такие сетки предлагается использовать для эффективного моделирования широкого класса электрофизических приборов. Экспериментально показана необходимость локальной модификации прямоугольных сеток при расчетах в областях с криволинейной границей. Разработаны двухшаговые алгоритмы локальной модификации рассматриваемых квазиструктурированных сеток. На первом шаге проводится модификация приграничных узлов путем их сдвига на границу области по нормали к ней, а на втором – преобразование тех сеточных элементов, которые не удовлетворяют критериям качества, в качественные сеточные элементы. Разработаны специальные алгоритмы проведения таких преобразований, которые не нарушают структурированности подсеток в подобластях. Даны рекомендации по построению сеток на границах сопряжения подобластей (интерфейсе), которые содержат несогласованные сетки. Разработаны алгоритмы локальной модификации сеток на интерфейсе между подобластями, одна из которых содержит отрезок границы расчетной области. Проведены серии численных экспериментов по решению модельной задачи, результаты которых показали обоснованность предлагаемых подходов.

Ключевые слова: квазиструктурированные сетки, локальная модификация, метод декомпозиции, задачи сильноточной электроники

ОБРАЗЕЦ ЦИТИРОВАНИЯ

Козырев А.Н., Свешников В.М. О построении двумерных локально-модифицированных квазиструктурированных сеток и решении на них краевых задач в областях с криволинейной границей // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2017. Т. 6, № 2. С. 05–21. DOI: 10.14529/cmse170201.

Введение

В работе [1] были намечены основные принципы построения квазиструктурированных сеток для решения задач сильноточной электроники [2], состоящих в расчете интенсивных пучков заряженных частиц, движущихся в электромагнитных полях. Преимущественная ориентация на данные задачи сохраняется и в настоящей работе. Важную роль в них играет расчет потенциала электрического поля в разномасштабных областях с криволинейной границей, приводящий к решению уравнения Пуассона. Рассматриваемые квазиструктурированные сетки являются адаптивными. Настройка внутренних подсеток, входящих в квазиструктурированные

сетки, осуществляется путем регулировки плотности узлов в них, а подсетки вблизи криволинейных границ требуют специального рассмотрения, что и делается в настоящей статье. Основой построения служат прямоугольные подсетки. Прямоугольные сетки плохо приближают решение вблизи криволинейных границ, что, в частности, следует из численных экспериментов, проведенных в настоящей работе. В связи с этим, осуществляется их локальная модификация, основные принципы которой были заложены в [1], а здесь существенно доработаны. Кроме того, разработаны критерии построения сеток на границе сопряжения подобластей (интерфейсе). Основой проверки качества построенных сеток служили результаты численных экспериментов по решению модельных задач в областях с криволинейными границами, имеющие аналитическое решение.

В настоящее время существует несколько способов построения адаптивных неструктурированных сеток: дифференциальные [3], в которых координаты узлов определяются путем решения дифференциальных уравнений в частных производных, алгебраические, в которых координаты узлов вычисляются из алгебраических соотношений [4], вариационные, основанные на минимизации функционала качества сеток [5, 6]. Мы не рассматриваем здесь неструктурированные сетки, так как решение задач сильноточной электроники на них чрезвычайно трудоемко. Дело в том, что при наличии большого числа заряженных частиц на первый план выходит задача, которая скрыта при формулировке численного алгоритма, а именно задача определения сеточного элемента, которому принадлежит заданная точка. Наиболее просто и эффективно она решается на структурированных прямоугольных сетках. Структурированные адаптивные сетки для сложных расчетных областей рассматривались в работе [7], но такие сетки содержат большое число непрямоугольных элементов, что негативно скажется на решении задач сильноточной электроники.

Локальная модификация прямоугольных сеток наиболее полно подходит для поставленных целей, так как непрямоугольные шаблоны строятся лишь в узкой полосе вблизи криволинейных границ, а большинство шаблонов внутри области, где проходит пучок заряженных частиц, остаются прямоугольными. Идея локальной модификации была изложена в работе [8]. Свое развитие она получила в работах [9, 10], включая реализацию в программах для решения задач сильноточной электроники. В [11] рассматривалась локальная модификация прямоугольных сеток, но для ликвидации некачественных сеточных элементов добавлялись узлы, нарушающие структурированность сеток. Общим недостатком работ по локальной модификации структурированных сеток является отсутствие эффективных средств локальной регулировки плотности узлов в подобласти прохождения пучка. Все это навело на мысль о разработке квазиструктурированных локально модифицированных сеток, состоящих из прямоугольных подсеток, которые адаптируются к внешней границе путем локальной модификации и к пучку заряженных частиц путем регулировки плотности узлов подсеток. Квазиструктурированные сетки с подсетками различных типов рассматривались в работе [12]. Их основным недостатком являлось то, что подсетки должны быть согласованными, что приводило к введению большого количества вспомогательных подобластей, чтобы удовлетворить требованию согласованности. Предлагаемые в настоящей работе сетки могут быть несогласованными. Сшивка решений краевых подзадач на несогласованных подсетках

осуществляется при помощи метода декомпозиции, основанном на прямой аппроксимации уравнения Пуанкаре—Стеклова на интерфейсе [13].

В разделе 1 настоящей работы приводится постановка задачи и кратко излагаются алгоритмы ее решения. Раздел 2 содержит описание алгоритмов локальной модификации. В разделе 3 приводятся результаты численных экспериментов на согласованных и несогласованных сетках. В заключении кратко излагаются полученные результаты и перспективы дальнейших исследований.

1. Постановка задачи и алгоритмы ее решения

Придерживаясь ориентации на задачи сильноточной электроники, сосредоточим внимание на влиянии криволинейных границ на расчет потенциала электрического поля. Для этих целей рассмотрим следующую задачу.

Пусть в замкнутой двумерной (плоской или осесимметричной, допускающей двумерную постановку) области $\overline{G} = G \bigcup \Gamma$ с границей Γ требуется решить краевую задачу для уравнения Пуассона

$$\Delta u = g_1, \quad l u\big|_{\Gamma} = g_2 . \tag{1}$$

Здесь $u = u(x, y)$ — искомая функция, $g_1 = g_1(x, y)$, $g_2 = g_2(x, y)$ — заданные функции, где x, y — декартовы или цилиндрические координаты, причем в последнем случае $x = r, y = z$, Δ — оператор Лапласа, l — оператор граничных условий, включающих условия Дирихле или Неймана. Будем предполагать без существенного ограничения общности, что Γ является кусочно-гладкой границей, состоящей из отрезков прямых и дуг окружностей, и условие Неймана задано на отрезках прямых, параллельных координатным осям. Такие условия могут быть поставлены при расчете электрофизических приборов, причем условия Дирихле ставятся на электродах, а условия Неймана — на линиях симметрии. Отметим, что задача Неймана (без границ с условиями Дирихле) не входит в рассмотрение, а задача Дирихле (без границ с условиями Неймана) допускается.

Опишем вокруг расчетной области прямоугольник $\overline{R} = \left\{0 \le x \le D_x,\ 0 \le y \le D_y\right\}$, где D_x, D_y — заданы, ($\overline{G} \subset \overline{R}$), в котором построим прямоугольную равномерную макросетку

$$\overline{\Omega}_H = \left\{ X_I = IH_x, Y_J = JH_y, \quad I = \overline{0, N_x}, J = \overline{0, N_y}, \quad H_x = \frac{D_x}{N_x}, H_y = \frac{D_y}{N_y} \right\},$$

где з N_x, N_y — аданные целые числа, с шагами $H_x, H_y >> h$ (h — максимальный шаг сетки, на которой аппроксимируется задача (1)). Фактически мы тем самым проводим декомпозицию G на подобласти $G_{I,J}$. Среди них будем различать $G_{I,J}^{(0)}$ — *внешние*, $G_{I,J}^{(1)}$ — *внутренние*, $G_{I,J}^{(2)}$ — *граничные* подобласти, которые соответственно не содержат точек G, содержат только точки G, содержат точки G и Γ. Точки пересечения координатных линий $x = X_I$, $y = Y_J$ макросетки являются макроузлами, которые обозначим как T_p, $p = \overline{1, P}$, где P — число макроузлов, а отрезки координатных линий, принадлежащие расчетной области, образуют границу сопряжения подобластей γ или интерфейс.

В замкнутых подобластях $\overline{G}_{I,J}^{(1)}$, $\overline{G}_{I,J}^{(2)}$ построим равномерные прямоугольные подсетки

$$\overline{\Omega}_{h,k} = \left\{ x_{i_k} = X_I + i_k h_{x,k},\, y_{j_k} = Y_J + j_k h_{y,k},\quad i_k = \overline{0, n_{x,k}},\, j_k = \overline{0, n_{y,k}} \right\}$$

с шагами $h_{x,k} = \dfrac{X_{I+1} - X_I}{n_{x,k}}$, $h_{y,k} = \dfrac{Y_{J+1} - Y_J}{n_{y,k}}$. Здесь введена единая нумерация подобластей и подсеток по индексу $k = \overline{1,K}$, где K — известное целое число. Допускаются несогласованные подсетки, то есть шаги в смежных подсетках могут быть разными. Будем предполагать без существенного ограничения общности, что $n_{x,k}$, $n_{y,k}$ — есть 2 в целой степени и шаги в смежных несогласованных подсетках отличаются в два раза. По аналогии с подобластями будем различать подсетки двух типов *внутренние* $\overline{\Omega}_{h,k}^{(1)}$ и *граничные* $\overline{\Omega}_{h,k}^{(2)}$.

Граничные подсетки $\overline{\Omega}_{h,k}^{(2)}$ подвергаются локальной модификации, состоящей в сдвиге приграничных узлов на границу Γ, в результате которой они преобразуются в подсетки $\overline{\Omega}_{h,k}^{(2,m)}$. Локальная модификация узлов сетки на интерфейсе может затронуть и внутренние подсетки $\overline{\Omega}_{h,k}^{(1)}$, соседствующие с $\overline{\Omega}_{h,k}^{(2,m)}$, в результате чего они преобразуются в подсетки $\overline{\Omega}_{h,k}^{(1,m)}$.

На границе сопряжения введем сетку ω_h, не содержащую макроузлы T_p: $\omega_h = \{(x_i, y_i) \in \gamma, (x_i, y_i) \neq T_p,\ i = \overline{1,M}\}$, где M — известное целое число. Дополнив ω_h макроузлами T_p, образуем сетку $\overline{\omega}_h = \{\omega_h, T_1, T_2, \ldots T_P\}$. Объединение подсеток $\overline{\Omega}_{h,k}^{(1,m)}$, $\overline{\Omega}_{h,k}^{(2,m)}$ и $\overline{\omega}_h$ образует результирующую квазиструктурированную сетку $\overline{\Omega}_h$, на которой решается краевая задача (1).

Решение исходной краевой задачи (1) проводится методом декомпозиции расчетной области на подобласти, сопрягаемые без наложения. Сшивка решений в подобластях осуществляется путем решения уравнения Пуанкаре—Стеклова

$$\left(\frac{\partial u}{\partial \vec{n}}\right)_{\gamma}^{(+)} - \left(\frac{\partial u}{\partial \vec{n}}\right)_{\gamma}^{(-)} = 0 \qquad (2)$$

на интерфейсе γ, где $\vec{n}$ — нормаль к γ, а знаки $+,-$ указывают на принадлежность объекта разным сторонам интерфейса. Для его решения строится внешний итерационный процесс по отысканию приближенных значений искомой функции на интерфейсе, которые мы обозначим через v_h. Внешним данный процесс назван потому, что на каждой его итерации внутренним итерационным методом находятся приближенные значения искомой функции u_h в подобластях. В узлах сетки ω_h уравнение (2) аппроксимируется системой линейных алгебраических уравнений [13]

$$Av_h + b = 0,$$

где $A = \left\{a_{i,j,}, i, j = \overline{1,M}\right\}$ — квадратная матрица, $b = \left\{b_i, i = \overline{1,M}\right\}$ — известный, а $v_h = \left\{v_{h,i}, i = \overline{1,M}\right\}$ — искомый векторы. Решение данной системы осуществляется итерационным методом вида

$$v_h^{(\eta+1)} = \Lambda\left(v_h^{(\eta)}, Av_h^{(\eta)}\right),\quad \eta = 0,1,\ldots \qquad (3)$$

где Λ — функция, определяющая конкретный алгоритм. Смысл формулы (3) заключается в том, что в данном итерационном процессе используются лишь сам вектор $v_h^{(\eta)}$ и действие $Av_h^{(\eta)}$ матрицы на вектор (здесь вместо v_h может быть какой-либо другой вспомогательный вектор). Этим требованиям удовлетворяет, например, семейство быстросходящихся итерационных методов в подпространствах Крылова [14], примерами которых могут служить метод сопряженных градиентов, метод сопряженных невязок, метод обобщенных минимальных невязок.

Вычисление произведений $Av_h^{(\eta)}$, как показано в [13], реализуется по формуле

$$Av_h^{(\eta)} = f_h^{(\eta)} - b,$$

где компоненты $f_{h,i}^{(\eta)}$ вектора $f_h^{(\eta)}$ определяются как

$$f_{h,i} = \left(d_h^{(+)}u_h^{(+)}\right)_i - \left(d_h^{(-)}u_h^{(-)}\right)_i, \quad i = \overline{1,M}. \tag{4}$$

Здесь $d_h^{(+)}, d_h^{(-)} = d_h$ — операторы аппроксимации нормальных производных, входящих в уравнение Пуанкаре—Стеклова (2), то есть

$$\left(\frac{\partial u}{\partial \vec{n}}\right)_h \approx d_h u_h. \tag{5}$$

Для вычисления вектора b функция v_h на интерфейсе полагается равной нулю, решаются краевые подзадачи в подобластях, вычисляются разности производных (4) в узлах сетки ω_h на интерфейсе, которые и являются компонентами вектора b.

Сходимость итерационного процесса (3) считается достигнутой, если норма невязки $\left\|f_h^{(\eta)}\right\| = \max\left|f_{h,i}^{(\eta)}\right|$ удовлетворяет неравенству $\left\|f_h^{(\eta)}\right\| \le \delta$, где δ — заданная малая величина.

Дополнив v_h значениями в макроузлах T_p, получим вектор $\overline{v}_h$, определенный на сетке $\overline{\omega}_h$. Компоненты данного вектора в макроузлах определяются из уравнений

$$\left(\Delta_h^{(p)}\overline{v}_h\right)_j = 0 \quad j = \overline{M+1, M+P}, \tag{6}$$

где $\Delta_h^{(p)}$ — аппроксимация оператора Лапласа на сеточном шаблоне, включающем узлы сетки ω_h, значения функции в которых предполагаются известными на текущей η-ой итерации (3), и один из макроузлов.

В целом решение краевой задачи (1) можно представить в виде последовательности следующих шагов.

Шаг 1. Задается начальное приближение $\overline{v}_h^{(0)}$ в узлах сетки $\overline{\omega}_h$.

Шаг 2. Во внутренних и граничных подобластях на сетках $\overline{\Omega}_{h,k}^{(1,m)}$, $\overline{\Omega}_{h,k}^{(2,m)}$ методом конечных объемов решаются краевые подзадачи, причем на отрезках интерфейса ставятся граничные условия Дирихле, определяемые по значениям $\overline{v}_h^{(\eta)}$.

Шаг 3. Рассчитываются разности производных по формулам (4).

Шаг 4. Делается очередной $(\eta+1)$-ый шаг по решению уравнения Пуанкаре—Стеклова при помощи итерационного процесса (3), в результате чего находятся значения $v_h^{(\eta+1)}$ в узлах сетки ω_h (без макроузлов).

Шаг 5. Из формулы (6) находятся значения искомой функции на текущей внешней итерации в макроузлах, что дает $\overline{v}_h^{(\eta+1)}$.

Шаг 6. Если сходимость внешнего итерационного процесса (3) достигнута, то считается, что приближенное решение исходной задачи получено, если же это не так, то повторяются шаги 2–5.

2. Модификация подсеток вблизи границ

Как уже отмечалось, настоящая статья содержит развитие работы [1] в части локальной модификации квазиструктурированных сеток. Основные требования, которым должны соответствовать построенные в настоящей работе сетки, были следующими: 1) сохранение структурированности подсеток, 2) соблюдение порядка точности решения на квазиструктурированных сетках.

2.1. Модификация внутренних приграничных узлов

Локальной модификации подвергаются подсетки $\overline{\Omega}_{h,k}$ путем сдвига узлов, отстоящих от границы на расстояние

$$\delta \le \frac{h}{2} \tag{7}$$

где h — шаг подсетки, на границу области.

При проведении локальной модификации подсеток помимо сохранения структурированности подсеток мы будем стремиться к построению сеточных шаблонов, удовлетворяющих условию Делоне [15]. Для этого достаточно построения нетупоугольных сеточных элементов.

Локальная модификация осуществляется в два этапа. На первом из них узлы, удовлетворяющие условию (7), сдвигаются на границу по нормали к ней. Прямоугольный шаблон при этом может трансформироваться как в треугольный, так и в произвольный четырехугольный шаблон, который кратчайшей диагональю разбивается на два треугольника. В результате не исключено появление тупоугольных треугольников. Для их ликвидации (второй этап) поступаем следующим образом.

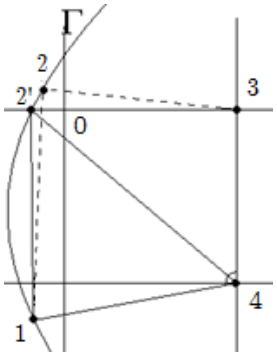


Рис. 1. Два тупых угла на одной стороне четырехугольного элемента

На рис. 1 изображена ситуация, при которой появляются тупоугольные треугольники из-за наличия двух тупых углов на одной стороне четырехугольного шаблона (на рис. 1 это сторона 34). Их ликвидация осуществляется по следующим правилам, приведенным в работе [1]. Меньший тупой угол 234 заменяется прямым 2'34. Это фактически означает модификацию сетки в данной окрестности не по нормали, а

вдоль координатной линии, то есть вместо сдвига 02 применяется сдвиг 02'. Если же два тупых угла появляются на стороне, примыкающей к Γ, то аналогичная процедура сдвига вдоль координатных линий применяется к обоим узлам.

Рассмотрим два варианта устранения тупых углов в треугольных элементах, которые предлагаются в настоящей работе (алгоритмы, приведенные в [1] не давали требуемой точности).

Вариант 1 наиболее простой. Он изображен на рис. 2, где угол 123 тупой. Биссектрисой 24, проведенной до пересечения с границей Γ, разобьем его на два остроугольных треугольника 124 и 423 (пунктиром здесь и на рис. 3 изображены вспомогательные линии). Появление точки 4 не влияет на структурированность сетки. Дело в том, что по предположению на рассматриваемом криволинейном куске границы Γ должно быть задано условие Дирихле, которое учитывается при построении сеточного уравнения в узле 2. В результате точка 4 исключается из расчетов, то есть данная процедура разбиения выполняется только при расчете коэффициентов сеточных уравнений.

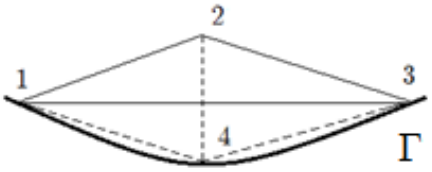


Рис. 2. Вариант 1 тупоугольных треугольников

Вариант 2 показан на рис. 3, где угол 236 тупой. В этом случае применить предыдущую процедуру нельзя, так как в результате разбиения получатся также тупоугольные треугольники, поэтому поступим следующим образом. Из точки 3 продолжим линию сетки до пересечения с границей Γ в точке 1. Получим два элемента: 1) 136 прямоугольный треугольник и 2) 24531 пятиугольник. Последний разобьем линиями 51 и 52 на прямоугольные треугольники 153, 245 и остроугольный треугольник 251. Так же, как в предыдущем варианте, точка 1 является вспомогательной. Она не нарушает структурированности сеток, так как исключается из сеточных уравнений в узлах 3 и 5 за счет учета граничных условий.

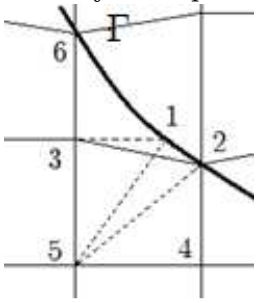


Рис. 3. Вариант 2 тупоугольных треугольников

2.2. Построение сетки на интерфейсе и аппроксимация уравнения Пуанкаре— Стеклова

Предлагается следующий способ построения сетки ω_h на интерфейсе γ. На рис. 4 изображено два возможных варианта сопряжения смежных подсеток $\overline{\Omega}_{h,+}$ и $\overline{\Omega}_{h,-}$: слева согласованные подсетки, справа — несогласованные подсетки.

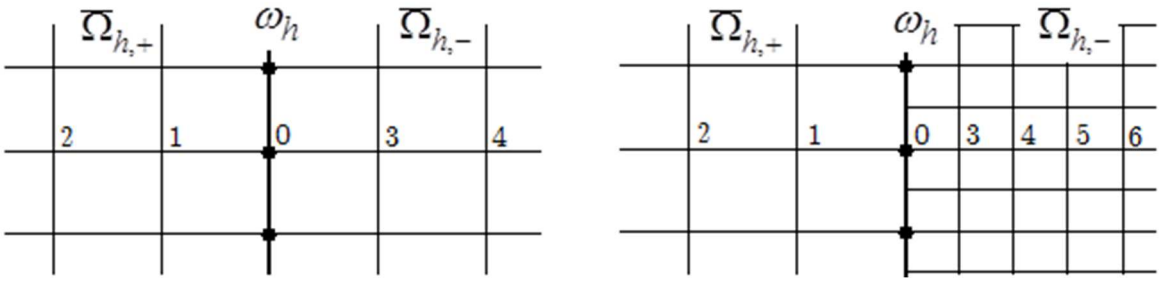


Рис. 4. Построение сетки ω_h на интерфейсе

Для согласованных подсеток предлагается в сетку ω_h включать узлы подсеток $\overline{\Omega}_{h,+}$ и $\overline{\Omega}_{h,-}$, лежащие на γ (на рис. 4 это выделенные узлы). В случае несогласованных подсеток, если следовать этой же технологии, возникает вопрос: узлы какой подсетки включать в ω_h — густой или редкой? Ответ на него дали результаты численных экспериментов по решению задачи, приведенной ниже в п.3, которые показали, что достаточно включить в ω_h узлы редкой подсетки, так как включение узлов густой подсетки не увеличивает точности расчетов. Это обстоятельство позволяет обойтись малым числом уравнений, аппроксимирующих уравнение Пуанкаре—Стеклова, что положительно сказывается на времени решения задачи в целом.

Операторами $d_h^{(+)}$, $d_h^{(-)}$ аппроксимирующими нормальные производные (5) в уравнении Пуанкаре—Стеклова, служили трехточечные схемы второго порядка, которые для соседних равноотстоящих с шагами h_+, h_- узлов, обозначенных цифрами 0, 1, 2, и 0, 3, 4, имеют вид

$$\left(d_h^{(+)}u_h\right)_0 = \frac{3\left(u_h\right)_0 - 4\left(u_h\right)_1 + \left(u_h\right)_2}{2h_+},\ \left(d_h^{(-)}u_h\right)_0 = \frac{-3\left(u_h\right)_0 + 4\left(u_h\right)_3 - \left(u_h\right)_4}{2h_-}. \qquad (8)$$

Применение данных формул для смежных внутренних согласованных подсеток при $h_+ = h_- = h$ не вызывает затруднений. Иначе обстоит дело, во–первых, с несогласованными внутренними подсетками и, во–вторых, с граничными подсетками.

Рассмотрим случай несогласованных внутренних подсеток, изображенный справа на рис. 4. Непосредственное применение формул (8) приводит к тому, что берутся неравные шаги: шаг h_+ в подсетке $\overline{\Omega}_{h,+}$ и шаг h_- в подсетке $\overline{\Omega}_{h,-}$, а также соответственно узлы 0, 1, 2 и 0, 3, 4. Численные эксперименты, проведенные с такой аппроксимацией производных-с различными шагами, не дали удовлетворительной точности (см. табл. 4 в п.3). Тогда было решено взять при вычислении производных в $\overline{\Omega}_{h,-}$ шаг h_+ и узлы 0, 4, 6, то есть провести расчеты производных с одним и тем же шагом h_+ в разных подобластях $\overline{\Omega}_{h,+}$ и $\overline{\Omega}_{h,-}$. В этом случае точность расчетов

повысилась, что показано в табл. 4. Объяснение этому следующее. Непосредственным разложением в ряд Тейлора можно убедиться в том, что

$$u'_{\pm}(0)=\left(d_h^{(\pm)}u\right)_0+r_{\pm}(0),$$

где остаточный член $r_{\pm}(0)$ равен

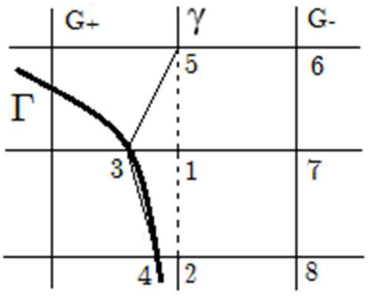
$$r_{\pm}(0)=\frac{1}{3}u'''(0)h_{\pm}^2+O\left(h^3\right).$$

Здесь штрихи означают производные по нормали, а $h=\max(h_+,h_-)$. Тогда для разностей производных справедливы соотношения

$$f(0)=u'_+(0)-u'_-(0)=\left(d_h^{(+)}u\right)_0-\left(d_h^{(-)}u\right)_0+\frac{1}{3}\left(h_+^2-h_-^2\right)+O\left(h^3\right).$$

Отсюда следует, что при равных шагах $h_+=h_-$ разность производных имеет на порядок выше точность, чем в случае неравных шагов.

Перейдем теперь к рассмотрению граничных подсеток. В работе [1] узлы сетки $\overline{\omega}_h$ на интерфейсе не подвергались модификации. Это могло привести к тому, что в приграничных узлах интерфейса γ между граничной подобластью G_+ и внутренней подобластью G_- на рис. 5, например, в узле 1 производная (5) в уравнении Пуанкаре—Стеклова аппроксимировалась по двум точкам 1 и 3 с первым порядком точности. Как показали численные эксперименты, тогда и решение имело первый порядок. Для повышения порядка точности было предложено проводить локальную модификацию узлов интерфейса путем их сдвига вдоль координатных линий на границу, что отражается на смежной подсетке в подобласти G_-, не являющейся граничной (не содержащей внешнюю границу). То есть узел 1 предлагается сдвинуть в точку 3, тогда в нем уже не решается уравнение Пуанкаре—Стеклова, так как он становится граничным. При этом ставится ограничение: смежные подсетки, имеющие интерфейс



вблизи внешней границы должны быть согласованными. Такая модификация, как следует из результатов численных экспериментов, приведенных ниже, повышает порядок точности до второго.

Рис. 5. Модификация узлов интерфейса вблизи внешней границы

3. Численные эксперименты

Основным критерием оценки качества построенных сеток служат результаты численных экспериментов по решению следующей модельной краевой задачи. В цилиндрическом конденсаторе, ограниченном двумя концентрическими окружностями с радиусами $R_1=0{,}1$ и $R_2=1$ с заданными на них потенциалами $g(R_1)=1$, $g(R_2)=2$

требуется рассчитать электрическое поле, то есть найти решение задачи (1). Ее аналитическое решение есть

$$u(r) = \ln^{-1} R_2 / R_1 \ln r R_2 / R_1^2 . \quad (9)$$

Численное решение данной задачи проводилось в двумерной постановке в декартовых координатах. Постановка рассматриваемой задачи во всей области была сделана с целью контроля симметрии сетки и значений функции. Симметрия была подтверждена экспериментально. Кроме того, с целью выяснения влияния условия Неймана на точность данная задача рассматривалась для четверти области, ограниченной линиями $x = 0$ и $y = 0$, на которых задавалось условие симметрии $\frac{\partial u}{\partial \vec{n}} = 0$. Результаты численных экспериментов показали, что условие симметрии не оказывает существенного влияния на точность.

Поставленная выше задача решалась как на согласованных, так и на несогласованных квазиструктурированных сетках. Принципы построения несогласованных сеток были следующими. Как видно из (9), рассматриваемая задача имеет особенность при $r \to 0$. В связи с этим, в граничных подобластях, содержащих границу с меньшим радиусом R_1, строились более густые подсетки, имеющие шаг в 2 раза меньший, чем шаг в остальных подобластях. Макросетка и подсетки были квадратными, имеющие параметры: $N_x = N_y = N$, $n_{x,k} = n_{y,k} = n$ — для согласованных подсеток и $n^{(1)}_{x,k} = n^{(1)}_{y,k} = n_1$, $n^{(2)}_{x,k} = n^{(2)}_{y,k} = n_2$ — для несогласованных подсеток. Конкретные значения параметров приводятся в нижеследующих таблицах. Краевая задача (1) аппроксимировалась на пятиточечных шаблонах в регулярных узлах и, в общем случае, на девятиточечных шаблонах в локально модифицированных узлах со вторым порядком точности. В численных экспериментах подсчитывалась максимальная относительная погрешность ε по формуле

$$\varepsilon = \max_{i,j \in \Omega_h} \left| \frac{u_{i,j} - (u_h)_{i,j}}{u_{i,j}} \right| . \quad (10)$$

Расчеты проводились при помощи пакета прикладных программ ЭРА-DD [16]. Каждая серия численных экспериментов была направлена на решение конкретного вопроса, которому ниже посвящается отдельный пункт.

3.1. Необходимость локальной модификации

Для ответа на вопрос о необходимости локальной модификации были проведены эксперименты по решению поставленной выше модельной задачи на равномерной квадратной сетке (простейший вариант квазиструктурированной сетки). Сетка не подвергалась модификации. Граничные условия Дирихле учитывались в точках пересечения координатных линий сетки с границей, что приводило к появлению сеточных шаблонов с неравномерными шагами.

Результаты расчетов на различных сетках при $R_1 = 0{,}1$ приведены в табл. 1, где ε — погрешность, вычисленная по формуле (10).

Таблица 1

Погрешности ε без локальной модификации

Сетка	32×32	64×64	128×128
ε	1,66	0,85	0,48

Из данной таблицы видно, что погрешность линейно, а не квадратично, как предсказывает теория [17], убывает с уменьшением шага сетки. Причина этого в появлении шаблонов с сильно неравномерными шагами, от которых необходимо избавляться. Этого мы достигаем при помощи локальной модификации приграничных узлов.

3.2. Численные эксперименты на согласованных сетках

Поставленная выше методическая задача решалась на согласованных сетках, которые подвергались локальной модификации по алгоритмам, изложенным выше. Расчеты проводились при различных параметрах N и n соответственно макросетки и подсеток, конкретные значения которых приведены в нижеследующих таблицах. На рис. 6 изображена сетка для $N = 4, n = 8$.

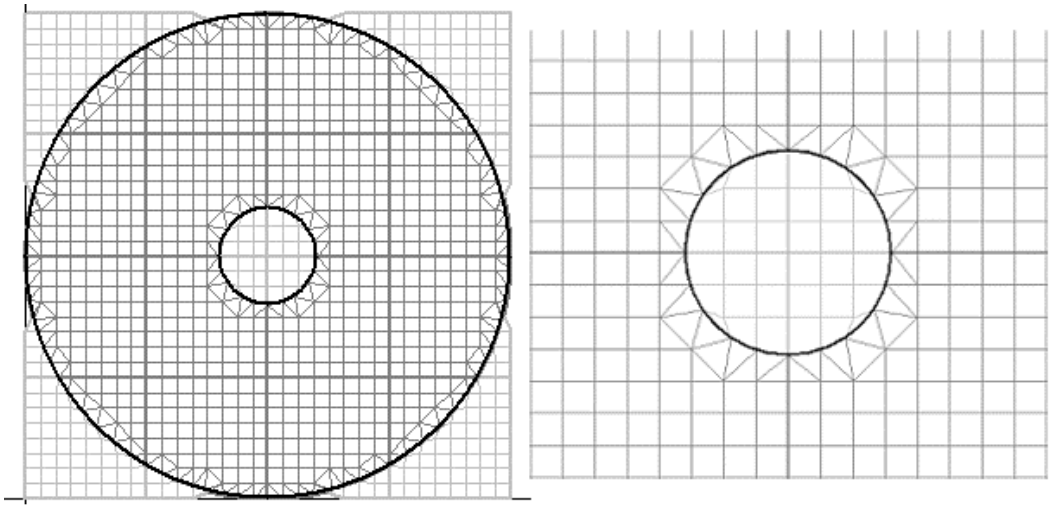


Рис. 6. Согласованная локально модифицированная сетка во всей области и вблизи границы

Кроме значения $R_1 = 0{,}1$, для выяснения характера влияния особенности проводились расчеты для $R_1 = 0{,}2$. Результаты расчетов приведены в табл. 2, причем верхние подстроки соответствуют $R_1 = 0{,}1$, а нижние — $R_1 = 0{,}2$.

Таблица 2

Погрешности ε расчетов на согласованных сетках с различными радиусами внутренней окружности

n \ N	4	8	16
8	9,88E-01	1,59E-01	3,91E-02
	2,13E-01	5,46E-02	1,24E-02
16	1,56E-01	3,51E-02	1,06E-02
	4,97E-02	1,49E-02	4,84E-03
32	3,52E-02	1,03E-02	3,29E-03

	1,45E-02	4,89E-03	1,36E-03
64	1,03E-02	3,27E-03	8,24E-04
	4,88E-03	1,36E-03	3,86E-04

Из табл. 2 можно сделать следующие выводы.

1. Погрешность с уменьшением шага сетки уменьшается приблизительно квадратично (особенно это заметно на густых сетках), что подтверждает второй порядок точности.
2. Сравнение результатов табл. 2, полученных с локальной модификацией, с результатами табл. 1, полученных без локальной модификации (надо сравнивать табл. 1 с верхними подстроками первого столбца табл. 2), говорит о том, что в результате локальной модификации значительно (от 1,6 до 14 раз) повысилась точность расчетов.
3. Сравнение результатов расчетов с различными радиусами говорит о том, что при удалении от особенности, как и следовало ожидать, точность расчетов увеличивается.

3.3. Расчеты на несогласованных сетках

Как уже отмечалось, для несогласованных смежных подсеток в сетку ω_h включаются узлы редкой подсетки. Тогда на очередном шаге решения уравнения Пуанкаре—Стеклова граничные значения функции на интерфейсе для решения краевой подзадачи на смежной густой подсетке определяются интерполяцией. Были рассмотрены два метода интерполяции: 1) сплайновая кубическая интерполяция [18] и 2) линейная интерполяция, то есть в данном случае полусумма соседних значений. Для первого варианта результаты расчетов приведены в верхних подстроках табл. 3, а для второго – в нижних подстроках. Расчеты проводились на различных квазиструктурированных несогласованных сетках, параметры которых указаны в нижеследующих таблицах

Таблица 3

Погрешности ε расчетов на несогласованных сетках
с различными вариантами интерполяции

n_1-n_2 \ N	4	8	16
8-16	1,57E-01	3,39E-02	1,05E-02
	0,20	0,23	9,14E-02
16-32	3,51E-02	1,03E-02	3,31E-03
	6,93E-02	8,84E-02	3,98E-02
32-64	1,03E-02	3,29E-03	8,31E-04
	2,84E-02	3,90E-02	1,88E-02

Сравнение результатов в подстроках данной таблице говорит о том, что погрешность линейной интерполяции оказывает негативное влияние на точность решения, а сплайновая интерполяция значительно повышает точность, поведение погрешности становится адекватным (соблюдается второй порядок точности).

Сравнение расчетов на несогласованных сетках (табл. 3) и расчетов на согласованных сетках (табл. 2) показывает хорошее совпадение результатов (сравниваются результаты верхних подстрок при $n = n_2$). Этот факт говорит о преимуществах квазиструктурированных несогласованных сеток, которые сгущаются только в районе особенности. Действительно, согласованные сетки имеют значительно (почти до 4 раз) большее число узлов по сравнению с соответствующими несогласованными сетками для обеспечения той же точности. То есть, согласованные структурированные сетки содержат лишние узлы, которые необходимы только для поддержки структурированности, а для обеспечения требуемой точности достаточно использовать квазиструктурированные несогласованные сетки, содержащие значительно меньшее число узлов.

В табл. 4 приведены результаты решения задачи с двумя вариантами аппроксимации производных на границе сопряжения несогласованных подсеток: 1) с одинаковыми шагами $h_+ = h_-$, 2) с различными шагами $h_+ \neq h_-$. В табл. 4 приведены значения погрешности для обоих вариантов (верхние подстроки – первый вариант, нижние – второй).

Из результатов данной таблицы следует, что расчеты с одинаковым шагом дают более высокую точность (особенно это заметно на густых сетках), что подтверждает теоретический вывод, приведенный в п. 2.2 настоящей работы.

Таблица 4

Погрешности ε расчетов на границе сопряжения
с различными вариантами аппроксимации производных

n_1-n_2 \ N	4	8	16
8-16	1.57E-01	3.39E-02	1.05E-02
	1.50E-01	4.68E-02	1.78E-02
16-32	3.51E-02	1.03E-02	3.31E-03
	3.62E-02	1.77E-02	5.20E-03
32-64	1.03E-02	3.29E-03	8.31E-04
	1.02E-02	5.28E-03	1.29E-03

Заключение

В работе предлагаются новые способы построения квазиструктурированных локально-модифицированных сеток. Такие сетки ориентированы преимущественно на решение задач сильноточной электроники с интенсивными пучками заряженных частиц. Данные задачи характеризуются сложной конфигурацией внешних границ и неоднородностью пучка. Для адаптации сеток к внешним границам разработана локальная модификация, а адаптации сеток к неоднородностям пучка проводится путем регулировки плотности узлов подсеток в подобластях. Качество построенных сеток проверено на примерах решения модельной краевой задачи в области с криволинейными границами. Проведены серии численных экспериментов, которые показали, что погрешность решения убывает со вторым порядком при дроблении шага сетки, в то время как расчеты без локальной модификации демонстрируют только первый порядок.

Исследованы вопросы о выборе сетки на интерфейсе, точности интерполяции и аппроксимации уравнения Пуанкаре—Стеклова при расчетах на несогласованных сетках. Показано преимущество квазиструктурированных сеток по сравнению с расчетами на структурированных согласованных сетках.

Дальнейшее направление исследований связано с улучшением качества квазиструктурированных сеток путем построения специальных треугольных подсеток в граничных подобластях, включающих сеточные элементы с криволинейными сторонами.

Работа выполнена при финансовой поддержке РФФИ (проект № 16-01-00168)

Литература

1. Свешников В.М., Беляев Д.О. Построение квазиструктурированных локально-модифицированных сеток для решения задач сильноточной электроники // Вестник ЮУрГУ. Серия: Математическое моделирование и программирование. 2012. № 40(299). С. 130–140.
2. Сыровой В.А. Введение в теорию интенсивных пучков заряженных частиц. Москва: Энергоатомиздат. 2004. 522 с.
3. Liseikin V.D. Grid generation methods. Berlin, Springer-Verlag, 1999. 363 p.
4. Шокин Ю.И., Данаев Н.Т., Хакимзянов Г.С., Шокина Н.Ю. Лекции по разностным схемам на подвижных сетках. Часть 2. Алматы: Изд–во КазНУ. 2008. 184 с.
5. Сидоров А.Ф. Об одном алгоритме расчета оптимальных разностных сеток // Труды математического института им. В.А. Стеклова. 1966. Т. 74. С. 147–151.
6. Ушакова О.В. Алгоритм построения двумерных оптимальных адаптивных сеток // Математическое моделирование. 1997. Т. 9, № 2. С. 88–90.
7. Анучина А.И., Артемова Н.А., Бронина Т.Н., Гордейчук В.А., Ушакова О.В. О разработке алгоритма построения сеток в конструкциях, образованных объемами вращения, в том числе и при их деформации // Тезисы докладов VIII Всероссийской конференции, посвященной памяти академика А.Ф. Сидорова и Всероссийской молодежной школы-конференции (Абрау–Дюрсо, 5–10 сентября 2016 г.). Екатеринбург: Изд–во Института математики и механики им. Н.Н. Красовского УрО РАН, 2016. С. 7–9.
8. Мацокин А.М.. Автоматизация триангуляции областей с гладкой границей при решении уравнений эллиптического типа // Препринт. Новосибирск: Изд–во ВЦ СО АН СССР, 1975. № 15. 15 с.
9. Ильин В.П., Ицкович Е.А., Куклина Г.Я., Сандер И.А., Сандер С.А., Свешников В.М. Расчет электростатических полей на локально-модифицированных сетках // Вычислительные методы и технология решения задач математической физики. Новосибирск: Изд–во ВЦ СО РАН, 1993. С. 63-72.
10. Свешников В.М. Численный расчет пучков заряженных частиц на локально модифицированных сетках // Препринт. Новосибирск: Изд–во ВЦ СО РАН, 1997. № 1109. 28 с.
11. Sander I.A. The program of Delaunay triangulation construction for the domain with the piecewise smooth boundary // Bulletin of the Novosibirsk Computing Center. Series: Numerical Analysis. Novosibirsk: NCC Publisher, 1998. P. 71-79.

12. Ильин В.П., Свешников В.М., Сынах В.С. О сеточных технологиях для двумерных краевых задач // Сибирский журнал индустриальной математики. 2000. Т. 3, № 1. С. 124–136.
13. Свешников В.М. Построение прямых и итерационных методов декомпозиции // Сибирский журнал индустриальной математики. 2009. Т. 12, № 3(39). С. 99–109.
14. Ильин В.П. Методы и технологии конечных элементов. Новосибирск: Изд–во ИВМиМГ (ВЦ) СО РАН. 2007. 371 с.
15. Скворцов А.В. Обзор алгоритмов построения триангуляции Делоне // Вычислительные методы и программирование. 2002. Т. 3, № 1. С. 18–43.
16. Беляев Д.О., Козырев А.Н., Свешников В.М. Пакет прикладных программ ЭРА-DD для решения двумерных краевых задач на квазиструктурированных сетках // Вестник НГУ. 2010. Т. 8, № 1. С. 3–11.
17. Самарский А.А. Теория разностных схем. Москва: Наука. 1977. 656 с.
18. Марчук Г.И. Методы вычислительной математики. Москва: Наука. 1977. 456 с.

Свешников Виктор Митрофанович, д.ф.-м.н., зав. лаб., лаборатория вычислительной физики, Институт вычислительной математики и математической геофизики СО РАН (г. Новосибирск, Россия)

Козырев Александр Николаевич, м. н. с., лаборатория вычислительной физики, Институт вычислительной математики и математической геофизики СО РАН (г. Новосибирск, Россия)

DOI: 10.14529/cmse170201

ON THE CONSTRUCTION OF TWO-DIMENSIONAL LOCAL-MODIFIED QUASISTRUCTURED GRIDS AND SOLVING ON THEM TWO-DIMENSIONAL BOUNDARY VALUE PROBLEM IN THE DOMAINS WITH CURVILINEAR BOUNDARY

A.N. Kozyrev, V.M. Sveshnikov

Institute of Computational Mathematics and Mathematical Geophysics SB RAS
(pr. Akademika Lavrentjeva 6, Novosibirsk, 630090 Russia)
E-mail: kozyrev@inbox.ru,victor@lapasrv.sscc.ru

Received: 01.05.2017

New approaches to local modification quasistructured grids, which allow to track the inhomogeneous boundary value problems in the computational domain and adaptable to curved boundaries, as well as easy to use and does not require the storage of large amounts of data as required in unstructured grids are developed. Such grids are proposed to use for the efficient simulation of a wide class of electro physical devices. It is experimentally shown the need for a local modification of the rectangular grid in calculations in domains with curvilinear boundary. The two-step algorithms for local modifications of considered quasistructured grids are developed. On the first step modification of the near boundary nodes is carried out by the its shift along the normal to boundary and on the second step the transformation of the grid elements that do not meet the quality criteria in a quality grid elements is carried out. Special algorithms for such transformations, which do not violate the structuring subgrids in subdomains are developed. Recommendations for the construction of grids on the interface of subdomains that contain the uncoordinated grids have been done. Algorithms local modification of grids on the

interface between the subdomains, one of which contains a segment of the computational domain boundaries, have been developed. The series of numerical experiments on solving a model problem are carried out. The results of numerical experiments showed the validity of the proposed approaches.

Keywords: quasistructured grids, local modification, domain decomposition method, problems of high current electronics

FOR CITATION

Kozyrev A.N., Sveshnikov V.M. On the construction of two-dimensional local-modified quasistructured grids and solving on them two-dimensional boundary value problem in the domains with curvilinear boundary. *Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering.* 2017. vol. 6 no 2. pp. 05–21. (in Russian). DOI: 10.14529/cmse170201.

References

1. Sveshnikov V.M., Belyaev D.O. Construction of Quasi-Structured Locally Modified Grids for Solving Problems of High Current Electronics. *Vestnik Yuzho-Uralskogo gosudarstvennogo universiteta. Seriya "Vychislitelnaya matematika i informatika"* [Bulletin of South Ural State University. Series: *Computational Mathematics and Software Engineering*. 2012. no. 40(299). pp. 130–140. (in Russian)
2. Syrovoj V.A. *Vvedenie v teoriyu intensivnykh puchkov zaryazhennykh chastits* [Introduction to the Theory of Intense Charged Particle Beams]. Moscow, Energoatomizdat, 2004. 552 p.
3. Liseikin V.D. Grid Generation methods. Berlin, Springer-Verlag, 1999. 363 p.
4. Shokin Yu.I., Danaev N.T., Hakimzyanov G.S., Shokina N.Yu. *Lektsii po raznostnym skhemam na podvizhnykh setkakh. Chast' 2* [Lectures on the Difference Scheme on Moving Grids.Part 2]. Almaty, Kazakh National University, 2008. 184 p.
5. Sidorov A.F. An Algorithm for Calculating the Optimal Difference Grids. *Trudy matematicheskogo instituta im. V.A. Steklova.* [Proceedings of the Mathematical Steklov Institute]. 1966. vol. 74. pp. 147–151. (in Russian)
6. Ushakova O.V. An Algorithm for Constructing Two-dimensional Optimal Adaptive Grids. *Matematicheskoe modelirovanie* [Mathematical modeling]. 1997. vol. 9, no 2. pp. 88–90. (in Russian)
7. Anuchina A.I., Artemova N.A., Bronina T.N., Gordeychuk V.A., Ushakova O.V. On the development of the algorithm for constructing grids in the constructions formed by volumes of rotation, including when their deformation. *Tezisy dokladov VIII Vserossiiskoi konferentsii, posvyashchennoi pamyati akademika A.F. Sidorova i Vserossiiskoi molodezhnoi shkoly-konferentsii (Abrau–Dyurso, 5–10 sentyabrya 2016 g.).* [Abstracts of the VIII All–Russian conference dedicated to the memory of A.F. Sidorov and Russian youth school–conference (Abrau–Durso, 5–10 September 2016)]. Ekaterinburg: Publishing House of the Krasovsky Institute of Mathematics and Mechanics. UB RAS, 2016. pp. 7–9. (in Russian)
8. Matsokin A.M.. Automation triangulation of domains with smooth boundary for solving elliptic equations. Preprint. Novosibirsk: Computing Center of USSR Academy of Sciences, 1975. no. 15. 15 p.
9. Ilin V.P., Itskovich E.A., Kuklina G.Y., Sander I.A., Sander S.A., Sveshnikov V.M. Calculation of Electrostatic Fields on Locally Modified Grids. *Vychislitel'nye metody i tekhnologiya resheniya zadach matematicheskoi fiziki* [Computational methods and

technology solutions of problems of mathematical physics]. Novosibirsk: Computing Center of SB RAS, 1993. pp. 63–72. (in Russian)

10. Sveshnikov V.M. The numerical calculation of charged particle beams on locally modified grids. Preprint. Novosibirsk: Computing Center of Russian Academy of Sciences, 1997. Vol. 1109. 28 p.
11. Sander I.A. The program of Delaunay triangulation construction for the domain with the piecewise smooth boundary. Bulletin of the Novosibirsk Computing Center. Series: Numerical Analysis. Novosibirsk: NCC Publisher, 1998. pp. 71–79.
12. Ilin V.P., Sveshnikov V.M., Synakh V.S. On Grid Technologies for Two-Dimensional Boundary Value Problems. *Sibirskii zhurnal industrial'noi matematiki* [Journal of Applied and Industrial Mathematics]. 2000. vol. 3, no. 1. pp. 124–136. (in Russian)
13. Sveshnikov V.M. Construction of Direct and Iterative Methods of Decomposition. *Sibirskii zhurnal industrial'noi matematiki* [Journal of Applied and Industrial Mathematics]. 2009. vol. 12, no. 3(39). pp. 99–109. (in Russian)
14. Ilin V.P. *Metody i tekhnologii konechnykh elementov* [Methods and Technologies of Finite Elements]. Novosibirsk, ICM&MG SBRAS, 2007. 370 p.
15. Skvortcov A.V. Overview of Algorithms for Constructiong Delanau Triangulation. *Vychislitel'nye metody i programmirovanie* [Numerical Methods and Programming]. 2002. vol. 3, no. 1. pp. 18–43. (in Russian)
16. Belyaev D.O., Kozyrev A.N., Sveshnikov V.M. Program Package ERA-DD for Solving Two-Dimensional Boundary Value problems on Quasi-Structured Grids. *Vestnik NGU* [Novosibirsk State University Journal of Information Technologies]. 2010. vol. 8, no. 1. pp. 3–11. (in Russian)
17. Samarskiy A.A. *Teoriya raznostnykh skhem* [The Theory of Difference Schemes]. Moscow, Nauka, 1977. 656 p.
18. Marchuk G.I. *Metody vychislitel'noy matematiki* [Methods of Computational Mathematics]. Moscow, Nauka, 1977. 456 p.

УДК 004.272.34 DOI: 10.14529/cmse170202

МОДУЛЯРНО-ЛОГАРИФМИЧЕСКИЙ СОПРОЦЕССОР ДЛЯ МАССОВЫХ АРИФМЕТИЧЕСКИХ ВЫЧИСЛЕНИЙ*

© 2017 г. И.П. Осинин

Саровская лаборатория имитационного моделирования
(607190 Саров, ул. Маяковского, д. 42)
E-mail: iposinin@mail.ru

Поступила в редакцию: 01.05.2017

Предлагаемый сопроцессор представляет собой самостоятельный сложнофункциональный (intellectual property — IP) блок системы-на-кристалле, позволяющий проводить математические вычисления над вещественными числами в уникальной модулярно-логарифмической системе счисления. Обеспечены два уровня преобразования исходных чисел: в модулярную систему счисления вместо традиционной позиционной и в логарифмическую систему счисления вместо плавающей точки. Благодаря этому сопроцессор обладает более высоким быстродействием, точностью и надежностью вычислений по сравнению с известными аналогами. Он состоит из набора одинаковых вычислительных ядер, каждое из которых выполняет однотактовые скалярные или векторные операции. В результате проведенных исследований и разработок предложены новые научные и технические решения, реализующие предложенные способы вычислений и кодирования данных. При этом преобразование кодов в модулярно-логарифмическую систему счисления и обратно не вносит значительных временных задержек при большом потоке входных данных за счет предложенных аппаратных решений, конвейеризирующих процесс интерполяции функции логарифма и преобразования кодов системы остаточных классов. Реализован прототип устройства на базе программируемой логической интегральной схемы в виде IP-блока. Целевой рынок решения — компании разработчики универсальных процессоров.

Ключевые слова: сопроцессор, реконфигурируемая архитектура, система остаточных классов, логарифмическая система счисления, высоконадежные вычисления.

ОБРАЗЕЦ ЦИТИРОВАНИЯ

Осинин И.П. Модулярно-логарифмический сопроцессор для массовых арифметических вычислений // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2017. Т. 6, № 2. С. 22–36. DOI: 10.14529/cmse170202.

Введение

В течение последних десятилетий наблюдается экспоненциальный рост вычислительных возможностей ЭВМ, благодаря чему высокопроизводительные вычисления стали важной технологией при проведении исследований и разработок развитых государств.

При этом помимо скорости вычислений, все большее внимание уделяется точности расчетов и надежности супер-ЭВМ, что связано с постоянным увеличением количества вычислительных ядер. Например, у TaihuLight — лидера списка Top500 [1] за июнь 2016 года их число превысило 10 млн. штук.

Традиционно для решения подобных задач используются вычисления с плавающей точкой по стандарту IEEE-754 [2], который обладает рядом существенных недостатков,

* Статья рекомендована к публикации программным комитетом Международной научной конференции «Параллельные вычислительные технологии (ПаВТ) 2017».

например, критические ошибки округления зачастую приводят к неверным результатам расчетов.

С другой стороны, в распространенных на рынке процессорах никак не задействовано распараллеливание самих арифметических операций на уровне системы счисления. Раскрытие этого потенциала несет в себе дополнительный потенциал повышения скорости вычислений.

Современные отечественные микропроцессоры не лишены упомянутых особенностей, при этом у них отсутствует блок векторной обработки вещественных чисел. В связи с этим актуально создание IP-блока сопроцессора, дополняющего основную систему-на-кристалле и расширяющего набор команд математическими скалярными и векторными инструкциями. Применение такого сопроцессора обеспечит дополнительные конкурентные преимущества компаниям на рынке супер-ЭВМ.

В данной работе рассмотрена концепция самостоятельного IP-блока системы-на-кристалле, позволяющего проводить математические расчеты с изменением традиционных принципов вычислений. Продукт не имеет прямых аналогов за счет применения модулярно-логарифмической системе счисления. Сопроцессор является конкретной реализацией научных и технических решений, приведенных в общем виде в [3].

Статья структурирована следующим образом. Первый раздел посвящен обобщению результатов исследования актуальности данной разработки. Второй содержит описание базовой технологии. В третьем приведены конкурентные преимущества предложенного сопроцессора и сценарии его внедрения. В заключении обобщены результаты, полученные в ходе данного проекта.

1. Исследование актуальности модулярно-логарифмических вычислений

Высокопроизводительные вычисления стали одной из важнейших технологией проведения исследований и разработок в области моделирования космоса, климата, макромира, микромира и многих других. Проведенный анализ выявил [4], что все они используют формат машинной арифметики IEEE-754. Его достоинства и недостатки, а также основные способы повышения скорости и надежности вычислений рассмотрены в данном пункте.

1.1. Способы повышения точности вычислений

IEEE 754 — широко распространенный стандарт, описывающий формат чисел с плавающей точкой, в котором вещественные числа представлены в виде знака, показателя степени и мантиссы.

Преимущество данного формата состоит в существенно большем диапазоне для 64-битных чисел, который составляет порядка 10^{308} значений по сравнению с фиксированной точкой (10^{19} значений). Однако ему присущи и серьезные недостатки, среди которых, например, наличие округлений, возникающих в ходе вычислений, что приводит к нарушению алгебраических свойств коммутативности и дистрибутивности. Это приводит к тому, что традиционные процессоры зачастую выдают ошибочный результат вычислений на следующих типах задач [4]:

– плохо обусловленные;

– с разномасштабными коэффициентами;
– чувствительные к классу эквивалентных преобразований.

Для их верного решения требуются высокоточные вычисления, основанные на:
– программных подходах;
– повышении разрядности обрабатываемых вещественных чисел (512 бит и более);
– вычислениях в логарифмической системе счисления (ЛСС).

Первые два подхода являются традиционными. Также возможно выполнение арифметических операций с дробями без округления результата (пакеты Wolfram Mathematica, MatLab и др.) [4]. Их реализация в современных процессорах сопровождается увеличением временных затрат из-за их алгоритмической сложности и отсутствия аппаратной поддержки в процессорах, что является существенным недостатком.

Третий вариант является перспективной альтернативой формату IEEE-754, так как в ЛСС нет выравниваний порядков. Вещественное число a представлено в ней двоичным логарифмом $L(a)$ его абсолютной величины и знаком S.

Преобразование числа из формата IEEE-754 в ЛСС происходит следующим образом (рис.1):
– знак S переписывается без изменений;
– характеристика числа E становится целой частью числа в ЛСС, т.к. $\log_2(2^E)=E$;
– вычисляется логарифм мантиссы M, который является дробной частью числа в ЛСС.

Обратное преобразование выполняется аналогично, но на последнем этапе вычисляется антилогарифм дробной части логарифмического числа.

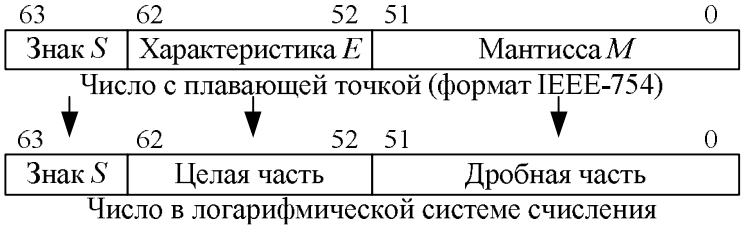


Рис. 1. Преобразование числа двойной точности формата IEEE-754 в ЛСС

Важной особенностью ЛСС является упрощение выполнения следующих операций:
– умножение заменяется сложением;
– деление заменяется вычитанием;
– возведение в степень заменяется умножением;
– извлечение корня заменяется делением.

Данное преимущество, продиктованное свойствами логарифмов, позволит повысить быстродействие предложенного сопроцессора на классе задач, где преобладают упомянутые операции. Однако за это приходится расплачиваться более сложным выполнением операций сложение и вычитание. Например, сложение можно определить следующей формулой:

$$\log_2(a+b)=\log_2 a+\log_2(1+2^{\log_2 b-\log_2 a}), \qquad (1)$$

где a — наименьший операнд, b — наибольший операнд. Из формулы видно, что сложение состоит из последовательности следующих преобразований:
– вычисление антилогарифма разности чисел в ЛСС;

– прибавление числа 1 к данному антилогарифму;
– вычисление логарифма от найденного числа на предыдущем шаге;
– суммирование логарифма с наименьшим из операндов.

Поскольку точка, отделяющая целую часть числа в ЛСС от дробной, является фиксированной, то выравнивание порядков не требуется, что является ключевым преимуществом по сравнению с плавающей точкой.

Основополагающий вклад в изучение ЛСС внесли ученые Дж. Коулман (J. Coleman), М. Арнолд (M. Arnold), Е. Честер (E. Chester), Д. Льюис (D. Lewis) [3, 4]. Однако, несмотря на достаточно обширный объем проведенных исследований, законченных технических реализаций на сегодняшний день несколько. Одной из них является логарифмический процессор European Logarithmic Microprocessor (ELM) [5], который имеет 32-разрядную RISC архитектуру. Однако остается малоизученным вопрос использования разрядности данных в ЛСС свыше 32 бит. Это связано с экспоненциально возрастающими аппаратными затратами на преобразование кодов из формата IEEE-754 в ЛСС и обратно, а также реализацию операции суммирования, что является недостатком данной системы счисления.

Описанные в п. 2.1 технические решения, основанные на проведенном автором исследовании, позволяют сгладить данный недостаток за счет использования более высокого порядка интерполяции, применяемого при преобразовании кодов, в чем заключается отличие от рассмотренного логарифмического процессора ELM. Кроме того, отличием является дополнительный уровень кодирования данных в системе остаточных классов. Ее особенности приведены далее.

1.2. Способы повышения скорости вычислений

На сегодняшний день наиболее распространены следующие пути повышения быстродействия:

– увеличение числа вычислительных ядер в составе супер-ЭВМ, а также повышение частоты процессора (например, за счет уменьшения размеров транзистора при создании СБИС);
– увеличение количества команд, выполняемых за такт, например, с помощью широкого командного слова в процессорах Эльбрус АО «МЦСТ» (до 26 операций за такт) или векторных операций (до 8 команд двойной точности за такт в сопроцессоре Intel AVX);
– совершенствование алгоритмов программ, например, за счет распараллеливания вычислений.

При этом перспективным резервом повышения производительности ядра процессора является распараллеливание выполнения самих арифметических операций на уровне разрядов системы счисления. В позиционной системе счисления (ПСС), на базе которой функционирует подавляющее большинство известных процессоров, этому препятствует последовательное распространение единиц переноса. Традиционное решение кроется в использовании вычислительного конвейера, однако, его аппаратная сложность значительно возрастает по мере увеличения числа ступеней, а положительный эффект достижим лишь при непрерывном потоке исходных данных.

Перспективной альтернативой является модулярная система счисления, или, по-другому, система остаточных классов (СОК), где вычисления над каждой цифрой числа

из базиса оснований $\{p_1,p_2,...,p_n\}$ осуществляются независимо друг от друга, где n — количество остатков.

Пусть СОК состоит из двух оснований $\{p_1=5,\ p_2=7\}$. Диапазон чисел ограничен произведением модулей. Преобразуем числа — найдем остатки от деления на p_1 и p_2, например, кортеж $\{2,3\}$ является уникальным представлением числа 17. Остатки, обладая существенно меньшей разрядностью, могут суммироваться, либо умножаться параллельно:

$$+\begin{cases}17_{10}=\{2_{\mathrm{mod}\,5},3_{\mathrm{mod}\,7}\}\\ 10_{10}=\{0_{\mathrm{mod}\,5},3_{\mathrm{mod}\,7}\}\end{cases}$$
$$27_{10}=\overline{\{2_{\mathrm{mod}\,5},6_{\mathrm{mod}\,7}\}} \qquad (2)$$

Предложенные технические решения на базе СОК используются в основном для цифровой обработки сигналов, так как применение плавающей точки затруднено ввиду сложной реализации деления на степени числа 2. Научные работы Н.И. Червякова, И.Я. Акушского, А. Сасаки (A. Sasaki), Г. Гарнер (H. Garner), А. Омонди (A. Omondi) [5, 6] посвящены данному направлению.

Известный модулярный процессор общего назначения, функционирующий на базе искусственных нейросетей, был разработан в Ставрополе в 2005 г. [7]. Однако недостаточное внимание в научных работах уделено возможности совместного использования преимуществ СОК и ЛСС, в чем состоит ключевое отличие предложенного сопроцессора.

1.3. Способы повышения надежности вычислений

Одновременно с тенденцией экспоненциального роста производительности супер-ЭВМ, все большую важность приобретает безошибочность вычислений, так как даже одиночная ошибка может привести к неверному результату многодневных расчетов. Вероятность возникновения таких ошибок увеличивается пропорционально числу вычислительных ядер. Их появление при сотнях тысяч ядер становится достаточно частым явлением.

Кроме того, существует множество задач, где помимо скорости вычислений, необходима отказоустойчивость аппаратуры. Среди них системы, работающие в реальном времени. Все их объединяет то, что процесс счета не должен прерываться ни при каких обстоятельствах. На сегодняшний день это достигается с помощью следующих средств:

– резервирование на уровне компонент системы;
– дублирование или троирование вычислений;
– программная обработка исключительных ситуаций и т.п.

Недостаток перечисленных подходов состоит в том, что часть оборудования простаивает, либо дублирует работу, вплоть до момента отказа одного из компонентов системы. Кроме того, применяются корректирующие коды, например, контроль четности регистров процессора, но они могут обнаружить лишь ошибку, возникшую при чтении/записи памяти и не приспособлены для выявления ошибок, появившихся в арифметическом устройстве.

Иначе обстоит дело при использовании системы остаточных классов, которая, помимо потенциала естественного параллелизма вычислений, обладает средствами повышения надежности вычислений на уровне системы счисления. При этом

используется единый помехоустойчивый код для обнаружения ошибок, возникающих при обработке информации в арифметическом устройстве и передаче ее по каналам связи.

В этом случае полный диапазон представления чисел P делится на рабочий диапазон P_n из n модулей и контрольный P_k из k модулей. Данные названия условны, так как модули совершенно равноправны. При выполнении любой арифметической операции над двумя числами из рабочего диапазона результат также принадлежит этому диапазону P_n, в противном случае произошла ошибка. Подробнее алгоритм описан в [5, 7] отечественными учеными И.Я. Акушским, И.А. Калмыковым, Н.И. Червяковым, которые внесли значительный вклад в развитие данного направления.

2. Описание базовой технологии

В данном пункте приведено описание предложенной модулярно-логарифмической системы счисления и технических решений сопроцессора, реализующих арифметические операции на ее базе.

Фиксированная точка в ЛСС позволяет работать с вещественным операндом, как с целым числом, что позволяет ввести следующий уровень преобразования — в СОК, образуя при этом новую модулярно-логарифмическую систему счисления (МЛСС), объединяющую преимущества обоих систем. По количеству разрядов формат числа в МЛСС может совпадать с форматом IEEE-754, обеспечивая эквивалентный диапазон представления вещественных чисел, например, двойную точность при использовании 64 разрядов.

Предлагаемый набор $P=\{p_1=2^{15}\text{-}1,\ p_2=2^{16}\text{-}1,\ p_3=2^{16},\ p_4=2^{16}+1\}$ обеспечивает минимальное снижение (порядка 0,003%) диапазона представления 63 разрядного числа (64-й отведен под знак). Выбранные основания выполняют условие взаимной простоты и позволяют упростить преобразование кодов из ПСС в СОК и обратно, благодаря модулям вида $2^n\pm1$, где n — разрядность модуля [7].

Прямое преобразование из IEEE-754 в МЛСС происходит в два этапа (рис. 2):

– характеристика исходного числа с плавающей точкой становится целой частью логарифма, одновременно с этим вычисляется логарифм по основанию два от мантиссы исходного числа, разряд знака не изменяется; точка, отделяющая целую часть вычисленного логарифма от дробной, отбрасывается;
– вычисляются остатки от деления полученного целого числа на основания СОК.

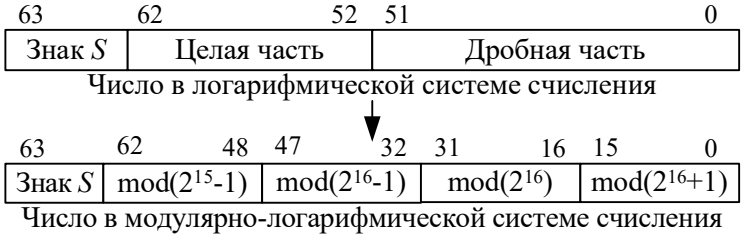


Рис. 2. Преобразование числа двойной точности из ЛСС в МЛСС

Преобразование из МЛСС в формат IEEE-754 проходит в обратной последовательности:

– по известным остаткам вычисляется позиционное представление логарифма; точка, отделяющая целую часть от дробной, восстанавливается на прежней позиции;

– вычисляется антилогарифм дробной части (мантисса), целая часть становится порядком искомого числа с плавающей точкой, разряд знака не изменяется.

Следует учитывать, что при одинаковом числе двоичных разрядов ПСС и СОК, диапазон представления чисел P последней будет меньше, так как в общем случае n-разрядный модуль числа меньше чем 2^n чисел, кодируемых тем же количеством разрядов ПСС. В связи с этим выбор набора оснований СОК является первостепенной задачей.

Перевод чисел из IEEE-754 в МЛСС и обратно осуществляется в предлагаемом сопроцессоре с помощью преобразователя кодов, рассмотренного далее.

2.1. Преобразователь кодов

Первый этап кодирования числа в МЛСС состоит в вычислении логарифма мантиссы, которая принимает значения на интервале от 1 до 2. В известных технических решениях [3, 4] использована линейная интерполяция с дополнительной коррекцией ошибки, что позволило обойтись относительно малыми аппаратными затратами, обеспечивая погрешность преобразования порядка E=10^{-7}, достаточную при вычислениях с одинарной точностью, но совершенно не достаточную для преобразования чисел двойной точности.

Уникальность предложенного решения состоит в применении квадратичной интерполяции с линейной интерполяцией ошибки каждого подинтервала, что позволяет обеспечить погрешность преобразования порядка E=10^{-16}, что достаточно при вычислениях с двойной точностью. Таким образом, логарифм y мантиссы x вычисляется по следующей формуле:

$$y = a \cdot x^2 + b \cdot x + c + (al \cdot xl + bl) \cdot maxerr, \tag{3}$$

где a, b, c — константы интерполяции интервала, которому принадлежит x;

xl — младшие 32 разряда мантиссы, определяющие подинтервал;

al, bl, $maxerr$ — константы интерполяции ошибки подинтервала.

Техническое решение, реализующее рассмотренное преобразование в аппаратуре, обладает параллелизмом конвейерного типа и состоит из сумматоров и умножителей. Его применение позволит уменьшить ошибку интерполяции в миллиард раз (с 10^{-7} до 10^{-16}) по сравнению с известным решением, оперирующем одинарной точностью при повышении аппаратных затрат в 8 раз.

Второй этап кодирования числа в МЛСС состоит в вычислении остатков от деления логарифмического представления числа на основания СОК. Благодаря специально выбранным основаниям, находящимся вблизи числа 2^n, вычисление остатков происходит с помощью сумматоров, разрядность которых не превышает 16 бит, что значительно сократит аппаратные затраты данного блока по сравнению с известными решениями [5].

Помимо основной функции преобразования кодов из формата IEEE-754 в МЛСС и обратно, аппаратный преобразователь кодов выполняет следующие операции:

– вычисление позиционной характеристики модулярного числа, что необходимо, например, при сравнении чисел, обнаружении переполнения разрядной сетки и машинного нуля;
– модулярно-логарифмическое сложение и вычитание вещественных чисел.

Остальные арифметические операции производятся в вычислительном ядре.

2.2. Вычислительное ядро

Каждое вычислительное ядро сопроцессора ведет независимую обработку по определенному модулю СОК. При этом операции умножение, деление и возведение в степень выполняются в векторном или скалярном виде.

В общем случае длина вектора ограничена лишь аппаратными затратами и скоростью выборки данных. Например, в рассмотренном варианте сопроцессора вектор имеет разрядность 128 бит, что составляет 8 упакованных в строку остатков, принадлежащих разным числам в МЛСС. Данный выбор обусловлен соотнесением с векторами, обрабатываемыми расширением Intel AVX2, в котором за один такт работы выполняется операция сразу над 8 числами.

Для обработки векторов в вычислительном ядре предусмотрено восемь 16-разрядных сумматоров и столько же умножителей. Устройство работает следующим образом. Каждый такт из кэш-памяти на обработку поступают два исходных вектора. В зависимости от команды над остаточными представлениями логарифмов выполняется одна из следующих операций:

– сложение (соответствует умножению исходных вещественных чисел);
– вычитание (соответствует делению)
– умножение (соответствует возведению в степень).

Результирующий вектор записывается в кэш-память. Причем, для устранения задержек, связанных с извлечением и записью данных, целесообразно применить трех-портовую кэш-память, расположенную на кристалле. За один такт такая память может выполнить чтение двух строк и запись одной строки.

Таким образом, в отличие от арифметико-логических устройств известных универсальных процессоров, малая разрядность обрабатываемых остатков и регулярность структуры вычислительного ядра позволяет распараллелить вычисления на макро-уровне операндов, составляющих вектор, и микро-уровне независимого счета остатков каждого операнда. При этом арифметические операции в ядре выполняются за один такт без использования конвейеризации.

2.3. Реконфигурируемая архитектура сопроцессора

В предложенной СОК количество оснований равно четырем, соответственно, сопроцессор состоит из четырех независимых ядер, которые подключены к преобразователю кодов и оперативной памяти (рис. 3). Кэш-память каждого ядра хранит информацию не обо всем числе, а лишь об остатке, по которому ведется обработка. Это позволяет организовать независимые вычисления в каждом ядре, а также некогерентную кэш-память.

Ключевой особенностью сопроцессора является динамическая реконфигурация, позволяющая управлять разрядностью вычислений, что бывает актуально для задач, не требующих высокой точности, но обладающих высокой степенью векторного параллелизма, например, к ним относится глубокое обучение нейросетей [2].

Возможны следующие варианты реконфигурации сопроцессора:

– 32 векторные операции половинной точности;
– 16 векторных операций одинарной точности;
– 8 векторных операций двойной точности.

В случае, когда на одном кристалле размещено n сопроцессоров, максимальная разрядность обрабатываемых целых чисел равна $64 \cdot n$. Например, при использовании восьми сопроцессоров в одной системе, диапазон обработки целых чисел достигает 512 бит с сохранением параллельной обработки по каждому остатку.

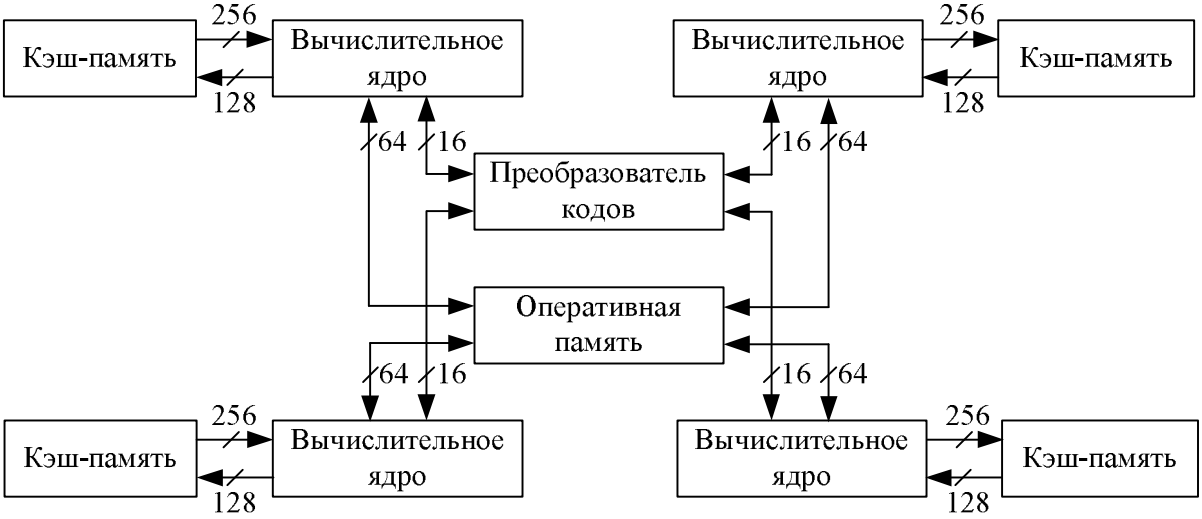


Рис. 3. Преобразование числа двойной точности формата IEEE-754 в ЛСС

Помимо изменения разрядности вычислений, реконфигурация позволяет отключить одно или несколько ядер в случае их отказа. Способ обнаружения в сопроцессоре одиночных и двойных ошибок, а также ситуаций отказа ядра базируется на свойствах СОК рассмотренных в п. 1.3. Для этого в сопроцессор необходимо добавить контрольное ядро. Диапазон представления чисел при этом делится на рабочий интервал и запрещенный интервал в отличие от универсальных процессоров, где требуется дублирование/троирование аппаратуры или процесса вычислений.

Доказано [5, 7], что при возникновении ошибки результат окажется в запрещенном интервале, причем предусмотрен механизм, локализующий ядро, где произошел сбой. Если ошибки появляются постоянно в одном и том же ядре, возникает ситуация отказа и ядро блокируется. Сопроцессор при этом продолжает работу, сокращается лишь диапазон представления чисел. Если при этом нельзя снижать точность вычислений, то можно использовать вычислительные ядра другого сопроцессора, в том числе и контрольные. Такая реконфигурация позволяет создать процессор, устойчивый к постепенной деградации.

3. Описание базовой технологии

На данный момент модулярно-логарифмический процессор существует в виде прототипа — IP-блока на базе программируемой логической интегральной схемы (ПЛИС). Данный выбор обусловлен простотой реализации, так как можно использовать готовые функциональные блоки софт-процессора Altera NIOS, дополнив основную систему команд инструкциями сопроцессора и использовав готовый компилятор для написания и отладки программ. Проектирование функциональных схем процессора проводилось в САПР Quartus II Web Edition фирмы Altera. Отладка и тестирование выполнялось на базе платы с ПЛИС Altera Cyclone V.

Аппаратные затраты на создание предлагаемого IP-блока составляют около 2 млн. транзисторов, тогда как для реализации блока векторных расширений AVX2 процессора Xeon семейства Haswell требуется около 21 млн. транзисторов [1], что на порядок выше. Далее приведено их сравнение по таким параметрам, как быстродействие, точность и надежность вычислений.

3.1. Быстродействие

Сравнение быстродействия модулярно-логарифмического сопроцессора (МЛС) с сопроцессором Intel AVX2 при предполагаемой равной частоте работы устройств (2,6 ГГц) приведено на рис. 4, где арифметические операции над числами двойной точности, упакованными в вектора размерностью 8 элементов, обозначены следующим образом:

- SUM — суммирование соответствующих пар операндов векторов;
- HSUM — редукционное (мультиоперандное) суммирование элементов вектора;
- MUL/DIV— умножение/деление пар операндов векторов соответственно;
- POW — возведение операндов первого вектора в степень, хранимую во втором векторе;
- $\log_2 x$ — логарифм по основанию 2 от каждого элемента вектора;
- matmul — перемножение плотных матриц размерностью 8*8 элементов (состоит из 64 операций умножения MUL и 64 редукционных суммирований HSUM);
- polinom — вычисление многочлена восьмой степени (состоит из одной операции POW, одного умножения MUL и одного редукционного суммирования HSUM).

Время выполнения операции на универсальном процессоре замерялось с помощью функции clock() библиотеки time.h. Для гарантированного исполнения векторных инструкций Intel применялись функции из набора арифметических операций библиотеки immintrin.h.

Наибольшее ускорение в предложенном сопроцессоре достигается при использовании операций, использующих свойства логарифмов. Например, для операций возведение в степень разница составляет три порядка в пользу сопроцессора, что объясняется отсутствием аппаратной реализации этих операций в сопроцессорах Intel.

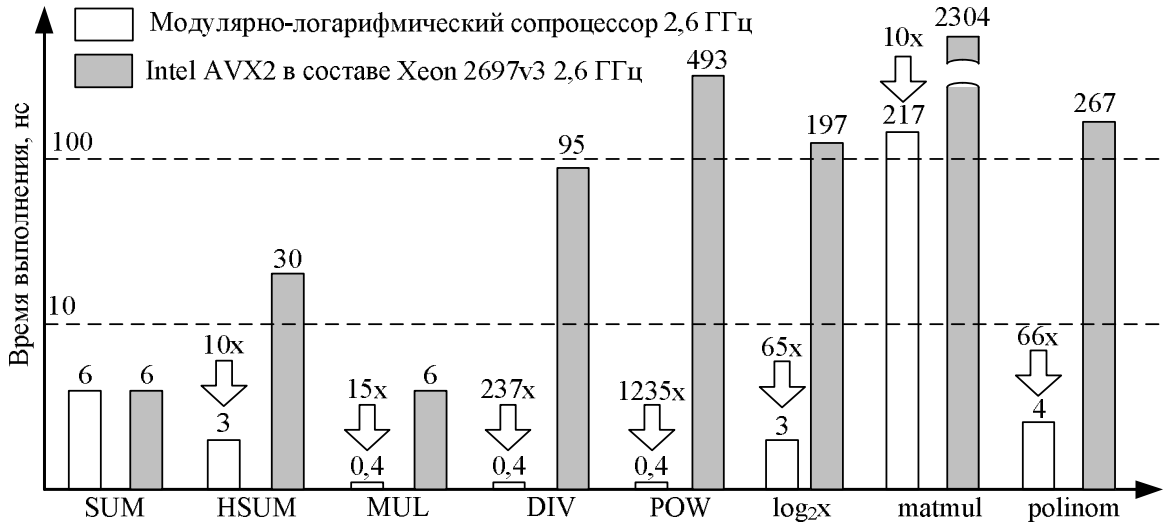


Рис. 4. Преобразование числа двойной точности формата IEEE-754 в ЛСС

Однотактовое выполнение векторных операций умножение, деление и возведение в степень позволяет получить значительное преимущество по скорости выполнения определенного класса задач, где таких операций большинство. К ним относится вычисление многочленов n-ой степени, например, разложение в ряд Тейлора, перемножение матриц и ряд других.

В данном тестировании принято, что преобразование в МЛСС и обратно аппаратно конвейеризировано и не вносит значительных временных задержек в вычислительный процесс.

3.2. Точность вычислений

Одним из вариантов тестирования точности вычислений является решение задачи Коши методом Эйлера. Рассмотрим дифференциальное уравнение $x'(t)=t$, где $x_0=0$, $t_0=0$. Аналитическое решение этого уравнения известно, а значит можно вычислить относительную погрешность решения, полученного численным путем в арифметике с плавающей точкой. Шаг интегрирования равен $1/2^i$, где i — натуральное число.

Результаты расчетов с двойной точностью в МЛС и формате IEEE-754 на рис. 5, где показана зависимость относительной погрешности решения задачи Коши от шага интегрирования.

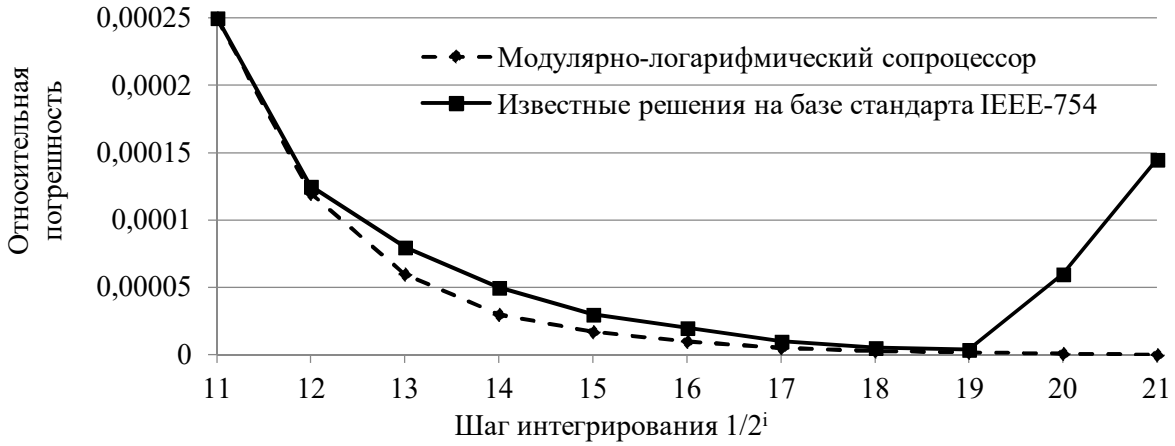


Рис. 5. Результаты тестирования точности вычислений

Из рис. 5 видно, что при уменьшении шага интегрирования относительная погрешность вычислений по стандарту IEEE-754 уменьшается, однако, после значения 19 резко возрастает, что обусловлено ошибками округления. Подобный эффект отсутствует при расчетах с помощью МЛС, что позволяет сделать вывод о более высокой точности вычислений в данном случае.

3.3. Оценка надежности

При построении надежностных характеристик предположим, что:

- в начальный момент времени в сопроцессоре имеется $n=k+r$ ядер, где $k=3$ — количество информационных ядер, $r=1$ — количество контрольных ядер;
- минимальное количество каналов для функционирования сопроцессора n-$(r+1)=2$;
- отказавшие каналы не восстанавливают работоспособность;
- отказы каналов являются статистически независимыми событиями.

Таким образом, надежностная структура МЛС соответствует модели скользящего резервирования, где резервные элементы находятся в нагруженном состоянии [3]. Вероятность его безотказной работы вычисляется следующим образом:

$$R_{МЛП}= \sum_{0}^{3\cdot k+4\cdot r} P^{4\cdot(k+r)-i}\cdot(1-P)^{i}, \tag{4}$$

где $P(t)=e^{-\lambda\cdot t}$ — вероятность безотказной работы канала, λ — интенсивность его отказа.

В традиционных системах высокой надежности определение и оперативная коррекция ошибок возможна лишь при условии одновременной работы нескольких устройств по принципу голосования, например, «2 из 3». Вероятность безотказной работы такой системы:

$$R_{ПСС}=3\cdot P_{ПСС}^{2}-2\cdot P_{ПСС}^{3}\ , \tag{5}$$

где $P_{\text{ПСС}}(t)=e^{-n\cdot\lambda_{\text{ПСС}}\cdot t}$, $\lambda_{\text{ПСС}}$ — интенсивность отказа одного разряда, n — разрядность позиционного процессора.

Результаты расчета надежности обоих вариантов при интенсивности отказов λ=10^{-5} ч$^{-1}$ представлены графически на рис. 6.

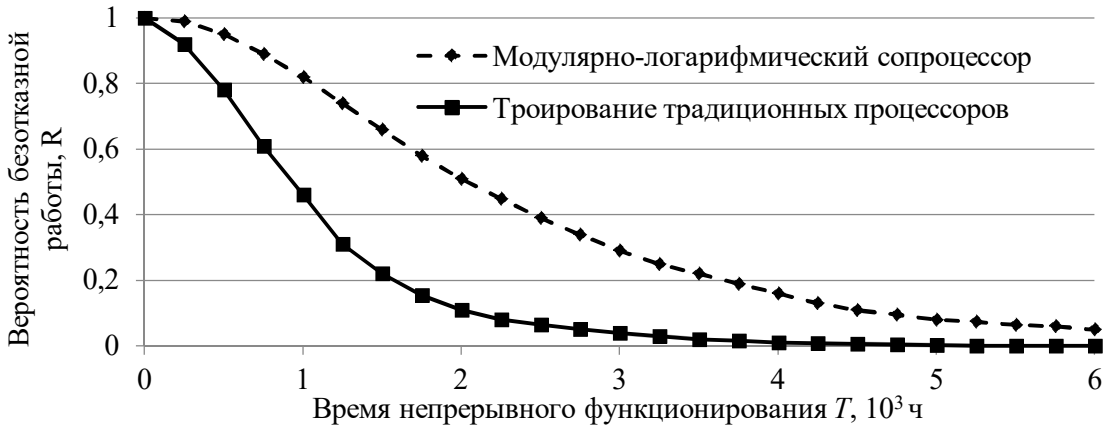


Рис. 6. Результаты тестирования точности вычислений

Анализ полученных зависимостей свидетельствует о преимуществе в надежности МЛС. Например, в течение t=2000 часов функционирования он обеспечивает вероятность безотказной работы R=0,5; тогда как позиционный процессор R=0,1. Избыточность аппаратурных затрат составляет 25% и 66% соответственно.

Заключение

Анализ приведенных в данном отчете способов повышения скорости, точности и надежности вычислений выявил затруднения на пути дальнейшего улучшения характеристик без внесения изменений на фундаментальном уровне самих принципов вычислений.

Перспективными системами счисления, призванными внести упомянутые изменения, являются модулярная система счисления как замена позиционной системы счисления и логарифмическая система счисления в качестве замены плавающей точки по стандарту IEEE-754. Таким образом, актуальной является разработка вычислительного

устройства, функционирующего в базисе совмещенной модулярно-логарифмической системы счисления.

В результате проведенных исследований и разработок предложены новые научные и технические решения, реализующие предложенные способы вычислений и кодирования данных, что позволило добиться следующих конкурентных преимуществ.

Высокое быстродействие по сравнению с известными аналогами достигается за счет следующих факторов:

– отсутствие переносов между модулями одного числа;
– замена умножение/деление на сложение/вычитание;
– замена возведение в степень/извлечение корня на умножение/деление.

Например, вычисление произвольного полинома восьмой степени в прототипе сопроцессора на ПЛИС выполнено втрое быстрее (90 нс против 267 нс) по сравнению с серверным процессором Intel Xeon 2697 v3 при разнице тактовой частоты в 26 раз в пользу Intel (0,1 ГГц против 2,6 ГГц).

Более высокая точность вычислений по сравнению с использованием плавающей точки достигается благодаря следующим факторам:

– отсутствие округления результата при выполнении операций умножение, деление, возведение в степень;
– отсутствие выравнивания порядков, в результате чего происходит потеря значащих разрядов.

Это позволяет верно решать ряд задач, критичных к ошибкам округления в перечисленных операциях и содержащих разномасштабные данные, в отличие от универсальных процессоров.

В отличие от известных вычислительных устройств, высокая надежность обеспечена на аппаратном уровне за счет следующих механизмов:

– выявление одиночных и двойных ошибок в вычислительном устройстве;
– реконфигурация ядер сопроцессора при отказе одного или нескольких из них с продолжением процесса счета.

Стоит отметить, что преобразование кодов в МЛСС и обратно не вносит значительных временных задержек при большом потоке входных данных за счет предложенных аппаратных решений, конвейеризирующих процесс интерполяции функции логарифма и преобразования кодов системы остаточных классов.

Одним из перспективных направлений внедрения разработки являются отечественные процессоры, например, Эльбрус-8С, Байкал-М, KOMDIV-64. Применение сопроцессора в этих системах-на-кристалле позволило бы дополнить их конкурентные преимущества рассмотренными, что, в конечном счете, сыграло бы важную роль в импортозамещении вычислительных компонент.

Литература

1. Top 10 Sites for November 2016 URL: www.top500.org/lists/2016/11 (дата обращения: 26.01.2017).
2. 754-2008 — IEEE Standard for floating-Point Arithmetic. Revision of ANSI/IEEE Std 754-1985 URL: ieeexplore.ieee.org/document/4610935, 2008 (дата обращения: 26.01.2017).

3. Осинин И.П. Высоконадежный модулярно-логарифмический процессор с реконфигурируемой архитектурой // Параллельные вычислительные технологии (ПаВТ'2016): труды международной научной конференции (Архангельск, 28 марта – 1 апреля 2016 г.). Челябинск: Изд-во ЮУрГУ, 2016. С. 642–654.
4. Осинин И.П., Князьков В.С. Направления развития архитектуры реконфигурируемых вычислительных платформ. Математическое моделирование развивающейся экономики, экологии и технологий (ЭКОМОД-2016): Труды IX Всероссийской научной конференции (Киров, 4–9 июля 2016 г.). Киров: Изд-во ВятГУ, 2016. С. 486–495.
5. Coleman J.N., Chester E.I. Arithmetic on the European Logarithmic Microprocessor // IEEE Transactions on Computers. 2000. Vol. 49, No. 7. P. 702–715. DOI: 10.1109/12.863040.
6. Ismail R.C., Hussin R., Muzad S.A. Interpolator Algorithms for approximating the LNS addition and substraction // IEEE International Conference on Circuits and Systems (ICCAS). Kuala Lumpur, 3 – 4 Oct. 2012. P. 174–179. DOI: 10.1109/iccircuitsandsystems.2012.6408336.
7. Червяков Н.И. Модулярные параллельные вычислительные структуры нейропроцессорных систем. Москва: Изд-во Физматлит, 2003. 288 с.
8. Omondi A. Residue Number System: Theory and Implementation. London: Imperial College Press, 2007. 312 p.
9. Калмыков И.А. Теоретические основы вычислений в полиномиальной системе классов вычетов, ориентированных на построение отказоустойчивых систем. Ставрополь: Изд-во СКФУ, 2006. 347 с.

Осинин Илья Петрович, к.т.н., н.с. ООО «Саровская лаборатория имитационного моделирования» (Саров, Российская Федерация)

DOI: 10.14529/cmse170202

MODULAR-LOGARITHMIC COPROCESSOR FOR MASSIVE ARITHMETIC CALCULATIONS

© 2017 I.P. Osinin

Sarov Laboratory of Simulation Modeling (Mayakovskogo 42, Sarov, 607190 Russia)

E-mail: iposinin@mail.ru

Received: 01.05.2017

The paper presents a conceptual design of an IP module of mathematical coprocessor. It consists of a set of processing cores of the same kind which perform single-cycle scalar, or vector operations with real numbers. The processed data is represented in the modular logarithmic format that provides two levels of translating the original numbers, namely: the modular level instead of the conventional positional system and the logarithmic level instead of the floating point format. As a result of the research and development, new scientific and technical solutions are proposed that implement the proposed methods of computing and coding data. Owing to this feature a coprocessor has a higher performance, a higher accuracy and a higher level of reliability, as compared to the known analogs. Convert codes in modular-logarithmic number system and vice versa does not introduce significant time delays in a large stream of input data by offering hardware solutions pipelined process of interpolation of the logarithm function and conversion of residual classes system codes. A prototype coprocessor is an FPGA-based IP module. Companies developing general-purpose processors are the target market for this design.

Keywords: residue number system, logarithmic number system, reconfigurable architecture, highly reliable computing.

FOR CITATION

Osinin I.P. Modular-logarithmic Co-processor for Massive Arithmetic Calculations. *Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering.* 2017. vol. 6, no. 2. pp. 22–36. (in Russian) DOI: 10.14529/cmse170202.

References

1. *Top 10 Sites for November 2016*. Available at: www.top500.org/lists/2016/11 (accessed: 26.01.2017).
2. *754-2008 — IEEE Standard for floating-Point Arithmetic*. Revision of ANSI/IEEE Std 754-1985 URL: ieeexplore.ieee.org/document/4610935, 2008 (accessed: 26.01.2017).
3. Osinin I.P. The Highly-modular Logarithmic Processor with a Reconfigurable Architecture. *Parallelnye vychislitelnye tekhnologii (PaVT'2016): trudy mezhdunarodnoj nauchnoj konferentsii (Arkhangel'sk, 28 marta – 1 aprelya 2016).* [Parallel Computational Technologies (PCT'2016): Proceedings of the International Scientific Conference (Arkhangelsk, Russia, March, 28 – April, 1, 2016)]. Chelyabinsk, Publishing of the South Ural State University, 2016. pp. 642–654. (in Russian)
4. Osinin I.P., Knyaz'kov V.S. Directions of Development of Architecture of Reconfigurable Computing Platforms. *Matematicheskoe modelirovanie razvivayushcheysya ekonomiki, ekologii i tekhnologiy (EKOMOD-2016): Trudy IX Vserossiyskoy nauchnoy konferentsii (Kirov, 4–9 iyulya 2016).* [Mathematical modeling of a developing economy, ecology and technology (ECOMOD-2016): Proceedings of the Scientific Conference (Kirov, Russia, July 4–9, 2016)]. Kirov: Publishing of the Vyatka State University, 2016. pp. 486–495. (in Russian)
5. Coleman J.N., Chester E.I. Arithmetic on the European Logarithmic Microprocessor // *IEEE Transactions on Computers*. 2000. vol. 49, no. 7. pp. 702–715. DOI: 10.1109/12.863040
6. Ismail R.C., Hussin R., Muzad S.A. Interpolator Algorithms for Approximating the LNS Addition and Substraction // *IEEE International Conference on Circuits and Systems (ICCAS)*. Kaula Lumpur, 3 – 4 Oct. 2012. pp. 174–179. DOI: 10.1109/iccircuitsandsystems.2012.6408336
7. Chervyakov N.I. *Modulyarnye parallel'nye vychislitel'nye struktury neyroprotses-sornykh sistem* [Modular structure of the parallel computing systems neuroprocessor]. Moscow, Publishing Fizmatlit, 2003. 288 p.
8. Omondi A. *Residue Number System: Theory and Implementation*. London: Imperial College Press, 2007. 312 p.
9. Kalmykov I.A. *Teoreticheskie osnovy vychisleniy v polinomial'noy sisteme klassov vychetov, orientirovannykh na postroenie otkazoustoychivykh system.* [Theoretical bases of calculations in polynomial system of residue classes, focused on building a fault-tolerant systems]. Stavropol, Publishing North-Caucasian Federal University, 2006. 347 p.

DOI: 10.14529/cmse170203

VIRTUALIZATION OF HETEROGENEOUS HPC-CLUSTERS BASED ON OPENSTACK PLATFORM*

© 2017 A.G. Feoktistov, I.A. Sidorov, V.V. Sergeev,
R.O. Kostromin, V.G. Bogdanova
Matrosov Institute for System Dynamics and Control Theory of SB RAS
(Lermontova str., 134, Irkutsk, 664033, Russia)
E-mail: agf@icc.ru, ivan.sidorov@icc.ru, vvsergeev@mail.ru, kostromin@icc.ru, bvg@icc.ru
Received: 01.05.2017

The paper addresses to the problem of integration of heterogeneous computing clusters to the united environment based on a virtualization technology. OpenStack software is selected as a platform for managing the virtual environment. The OpenStack platform provides a wide range of components and solutions to a functional interaction with different hypervisors. These include KVM, XEN, ESXi, QEMU and other systems. In addition to the OpenStack platform, we developed a specialized hypervisor shell. It helps to start virtual machines using queues of the traditional resource management systems, such as PBS, SLURM, LSF, or SGE, that are used on clusters of a center of collective usage. The developed model of the resource allocation for virtual machines allowed us to use the knowledge about job requests, resource characteristics and current state of the environment, and the expertise of it administrators. The realized tools provide the capability for the "painless" integration of heterogeneous clusters with the preinstalled local resource managers for creating the virtual cluster with the required configuration. Extensive modeling shows that the hypervisor shell can improve efficiency of integrated environment nodes through reallocating virtual machines to queues of the traditional resource management systems.

Keywords: computer clusters, HPC, virtualization technologies, OpenStack, simulation modeling

FOR CITATION

Feoktistov A.G., Sidorov I.A., Sergeev V.V., Kostromin R.O., Bogdanova V.G. Virtualization of Heterogeneous HPC-clusters Based on OpenStack Platform. *Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering.* 2017. vol. 6, no. 2. pp. 37–48. DOI: 10.14529/cmse170203.

Introduction

In the field of high-performance computing (HPC), computational clusters have become an essential element used to carry out various large problems within fundamental and applied studies. The nature of the target problems calls for a variety of hardware and software platforms, architectures and communication environments of HPC-clusters, which in turn significantly affects their performance, efficiency and reliability.

A variety of cluster systems makes the combined usage of HPC-clusters reasonable as it increases the total available computing power and reduces the mean waiting time for starting a job. The latter is achieved by using extended planning strategies, which consider the variations in time of the loading of different clusters and at the same time the flexibility in sharing the resources [1]. Jobs can be redirected from the currently most loaded cluster to the idle cluster, and a higher priority and other privileges can be assigned to them.

* The paper is recommended for publication by the Program Committee of the International Scientific Conference "Parallel Computational Technologies (PCT) 2017".

Jobs specify processes of problem solving. They include the information about the required computational resources, executable programs, input and output data, and other required items.

We assume that the integrated cluster environment (ICE) is based on the cluster systems that differ in their computational characteristics of nodes and parameters of administrative policies. There are different approaches to the creation of the ICE [2].

Our contribution to the solution of this problem is the development of methods and tools for creating heterogeneous distributed computing environments for different purposes with multi-agent management of computations [3–6]. These developments are based on the interaction with the traditional middleware Condor, Globus Toolkit, gLite, PBS Torque and other known systems. The creation of the environments was implemented in the Center of collective usage SB RAS "Irkutsk supercomputer center" [7].

Nevertheless, there remain many unresolved problems as follows:

– Providing in operating system of the allocated node the full set of libraries that are necessary to correctly launch and execute the instance of the distributed application whether it has the form of an executable file or a piece of program code;
– Ensuring the reliability for computing processes in the nodes of the environment;
– Difficulty of tracing the real causes of faults in computing processes in debugging applications due to a lack of direct access to the computational nodes;
– Security of the distributed computing due to a lack of reliable mechanisms of screening the executable files for the presence of malicious code.

One of the most promising ways to solve such problems is using virtualization techniques [8], which enable execution of the instances of the distributed and parallel applications in isolated environments with the required hardware and software characteristics. In this case, the physical nodes, which support the virtualization, can significantly differ in performance, hardware characteristics, types of their operating systems and other parameters.

In this regard, we describe the new approach to creating the ICE using virtualization techniques. This approach has the following advantages: the possibility of using complex knowledge about job requests; resource characteristics and current state of the environment; the expertise of its administrators; and the capability for the "painless" integration of heterogeneous clusters with the preinstalled local resource managers PBS, SLURM, or SGE to the virtual cluster with the required configuration.

1. Related work

In this section, we give a brief overview of software for a distributed computing virtualization. There is a wide variety of software tools that support resource virtualization [9]. In particular, they include the following popular tools:

– Docker for deployment and application management in a virtualization environment at the operating system level [10];
– QEMU for emulating hardware for different platforms [11];
– KVM for supporting virtualization in a Linux environment [12];
– Xen for virtual machines with para-virtualization support [13];
– VMware ESXi for enterprise-level virtualization, offered by VMware as a VMware vSphere component [14].

We have practical skills in using these tools for solving the following problems:
- Developing containers for web services with Docker;
- Testing software on various hardware configurations with QEMU;
- Enabling virtual machines for the resource management with KVM;
- Creating virtual computer clusters of various configurations for debugging and testing parallel and distributed applications with XenServer;
- Providing users with virtual machines of the required configuration using VMware vSphere.

When the aforementioned tools are used, problems of the cluster resources virtualization, integration of heterogeneous computer clusters, and providing service-oriented interfaces for integrated cloud infrastructure require additional solutions. There are also other software tools. However, they are based on the above listed tools or highly tailored.

The platforms OpenStack [15], Apache CloudStack [16], Eucalyptus [17] and Open Nebula [18] are package suites for creating cloud services according to Infrastructure-as-a-Service (IaaS) concept. The last three platforms are not widespread. In contrast, the OpenStack platform is leading in the field of the distributed computing virtualization.

There are following advantages of OpenStack:
- Involvement of major players in the cloud industry in the work with this platform;
- Wide range of software components and functional solutions, including service-oriented interface for users;
- Availability of extensive documentation covering components;
- Application of open standards;
- Availability of APIs for interoperability with external software;
- Support of work with different hypervisors (KVM, XEN, ESXi, QEMU and others);
- Extensive capabilities for developing own solutions, etc.

We have selected the OpenStack platform for the ICE virtualization and the hypervisor KVM for launching virtual machines in cluster nodes. However, the selected software requires changing traditional system software in nodes, which is not desirable to do. To solve this problem, we have developed a specialized hypervisor shell for running virtual machines as an additional module for the OpenStack. This hypervisor shell allows running virtual machines in the cluster nodes without the need for significant changes in their system software.

2. Architecture of the integrated cluster environment

We propose an architecture of the ICE that includes the following main components:
- Interfaces of an access level and the environment management node (Fig. 1);
- Platform for the environment virtualization management, specialized system software of a hypervisors level and software, and hardware of a computing resources level (Fig. 2).
- At the access level, there are three main interfaces for the interaction of different user categories with the environment.
- The command line interface (CLI) provides the interaction with a server for management of virtual machines. We use the PBS Torque for its implementation.
- Application programming interface (API) allows the external tools to interoperate with the environment to set up jobs, monitor them and get the results. API is based on the REST approach to creation of web services.

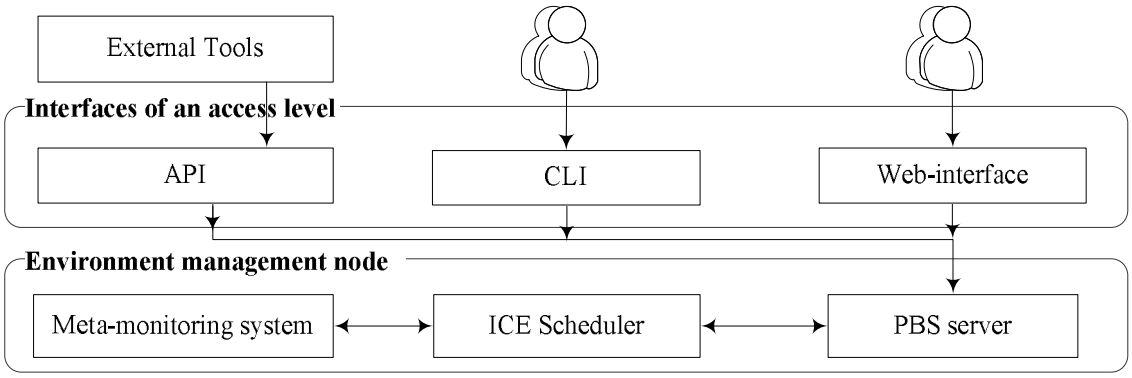


Fig. 1. Architecture of the access level and environment management node

The job for the ICE is the specification of the resource requirements for the execution of parallel or distributed programs. The job specifies the following requirements: number of nodes and cores, RAM size, specialized accelerators (Intel Xeon Phi, GPU), interconnectors, disk space, operating system type and others.

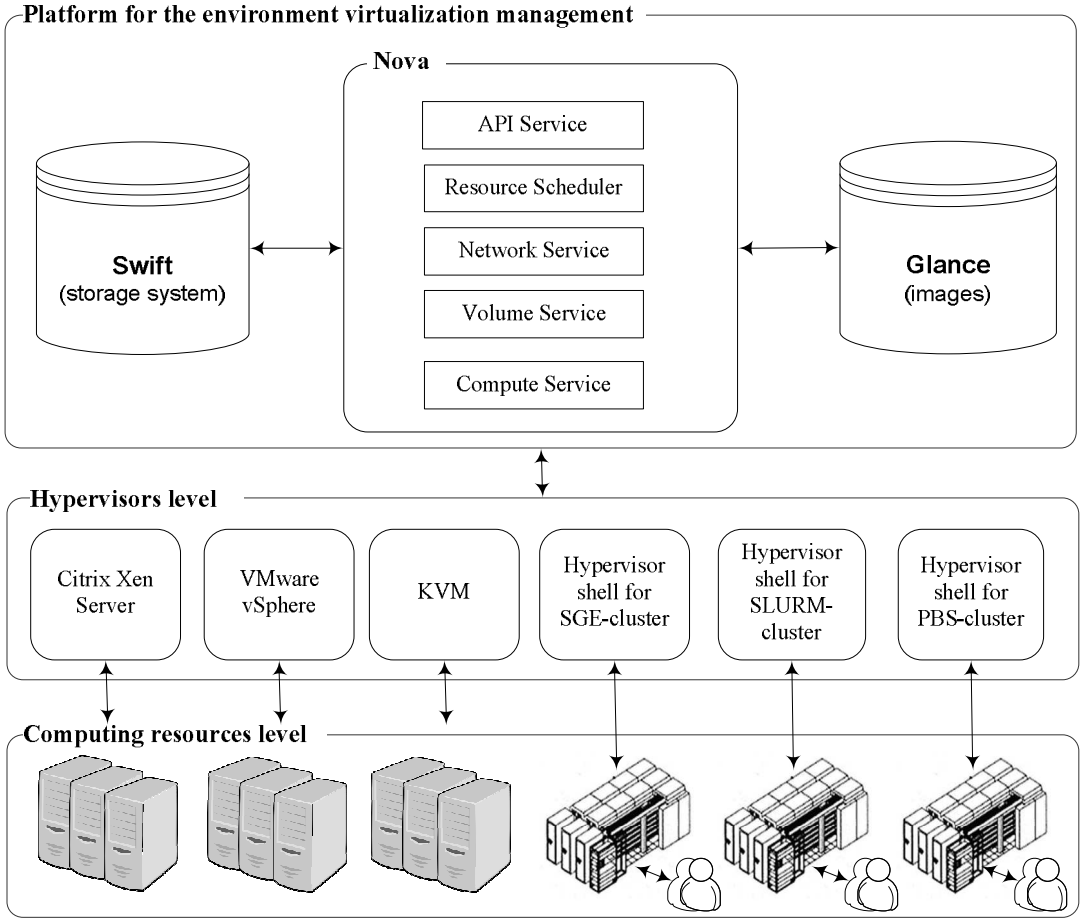


Fig. 2. Architecture of the ICE virtualization

Visual user interface includes the tools for adding jobs in the environment queues, and the tools for creating the virtual dedicated clusters of the required configurations similar to Amazon EC2 cloud services.

The following actions are carried out at the environment management node:

– Servicing the environment queues;

– Planning and distributing the requests for creating virtual machines;
– Managing pools of connected virtual machines created by virtualization management platform;
– Monitoring the environment components.

One of the key components of the ICE is the scheduler which processes jobs, allocates resources, interoperates with job queues of the PBS Torque, OpenStack platform and meta-monitoring system [19, 20]. The meta-monitoring system provides ICE scheduler with extensive data on the state of the cluster queues, the state of physical and virtual nodes and other necessary information. The specialized algorithms for this system provide the reliability of distributed computing [21].

The ICE scheduler defines the number and characteristics of virtual machines taking into account user's requirements. It sends defined information to the virtualization management platform (OpenStack) for forming and launching jobs, represented by virtual machines, in cluster nodes.

Currently, we use the three main components of OpenStack: computing resources controller (Nova), cloud file storage (Swift) and library of virtual machine images (Glance). The OpenStack processes information received from ICE scheduler, prepares virtual machine images and launches virtual machines by means of the traditional hypervisors and special hypervisors shell developed within the proposed approach.

The special hypervisor shell provides the following important additional possibilities:
– Monitoring the current state of cluster queues to spot idle resources;
– Adding a job to the selected cluster queue for launching the virtual machines;
– Configuring the launched virtual machines;
– Monitoring the computations for virtual machines;
– Migrating virtual machines;
– Finishing the virtual machines and cleaning the cluster resources.

The local resource managers PBS Torque SLURM and SGE with open source code are used for the job flow management in the clusters.

3. Model of resource allocation for virtual machines

In this section, we describe a model of the resource allocation (queues selection) for virtual machines. The model is based on the classification of jobs for an execution of virtual machines in nodes (resources) of the environment. The proposed job classification is based on the feature vector method [22].

Let us have a finite set $H = \{h_1, h_2, ..., h_k\}$ of job characteristics (requests). They include problem solving time, RAM and disk space size, number of nodes, processors and cores, program libraries, compilers, their keys, etc.

Each characteristic h_i is described by information structure, including the following components:
– Domain D_i of values with the symbol θ of uncertainty;
– Rank $r_i \geq 1$;
– Weight $w_i \geq 0$.

All elements of the set H are partially ordered descending in accordance with the ranks.

Let us denote by $C = \{c_1, c_2, ..., c_m\}$ a finite set of job classes. Each class c_j is defined by the main (mandatory) and additional (optional) sets of characteristics. The characteristics of the main and additional sets are presented by the Boolean matrixes A and B of dimension $k \times m$. The matrix elements $a_{ij} = 1$ or $b_{ij} = 1$ mean that the characteristic h_i is a member of the main or additional set, used in the definition of the class c_j and has the specialized domain $D_{ij}^* \subseteq D_i \setminus \{\theta\}$ for this class. If the condition $a_{ij} \vee b_{ij} = 0$ is satisfied, then $D_{ij}^* \equiv \{\theta\}$.

The matrixes A and B have satisfied the following conditions: $\bigvee_{j=1}^{m} \bigwedge_{i=1}^{k} \overline{a_{ij}} = 0$ and $\bigvee_{i=1}^{k} \bigvee_{j=1}^{m} (a_{ij} \wedge b_{ij}) = 0$.

An administrator of the environment forms the sets of characteristics and classes, and defines an accordance of resources to job classes.

The scheduler automatically determines characteristics used in a job specification and their domains for the job execution.

Let us denote by x a Boolean vector of dimension k for the job characteristics representation. There is bijection between elements of the vector x and indexes of the characteristics. Element values of the vector x are defined as follows:

$$x_i = \begin{cases} 0, \text{ if } D'_i \equiv \{\theta\}, \\ 1, \text{ if } D'_i \subseteq D_i \setminus \{\theta\}, \end{cases}$$

where D'_i is the domain of the characteristic h_i requested for the job execution, $i \in \{1, 2, ..., k\}$. The vector x has satisfied the following condition: $\bigwedge_{i=1}^{k} \overline{x_i} = 0$.

We use the characteristic function $\chi(x)$ to check the domain D'_i of the characteristic h_i for an accordance to the domain D_{ij}^* of this characteristic of the class c_j. This function is defined as follows:

$$\chi_j(x) = \begin{cases} 0, \text{ if } \exists i : (a_{ij} \vee b_{ij} = 1) \wedge (D'_i \not\subseteq D_{ij}^*), \\ 1 \text{ otherwise}, \end{cases}$$

where $i \in \{1, 2, ..., k\}$, $j \in \{1, 2, ..., m\}$.

The function $\chi(x)$ allows fulfilling of a primary job classification. As the result of this classification, the job can be related to several classes. In this instance, we use additional information such as the probabilistic measure of belonging to class, ranks and weights of characteristics, and computational history of jobs for more detailed classification.

Let us define the functions $\rho_j(x)$, $\sigma_j(x)$, $\omega_j(x)$ и $\phi_j(x, z)$, which calculate the following parameters of the characteristics relative to the class c_j:

- Probabilistic measure of belonging to class;
- Aggregated rank;
- Summary weight;
- Probabilistic measure of belonging to the class, taking into account the computational history z of jobs.

Let us denote by the symbols δ_ρ, δ_σ, δ_ω and δ_ϕ the upper limits of an equivalence for the functions ρ, σ, ω and ϕ, respectively. When the absolute difference of two values for one of the functions ρ, σ, ω or ϕ is less or equal to the upper limit of its equivalence δ_ρ, δ_σ, δ_ω or δ_ϕ, then these values are equivalent.

Let us define the following characteristic functions:

$$\chi_j^\rho(x,y)=\begin{cases}0, \text{ if } \max\limits_{\forall l:y_l=1}\{\rho_l(x)\}-\rho_j(x)>\delta_\rho,\\ 1 \text{ otherwise,}\end{cases}$$

$$\chi_j^\sigma(x,y)=\begin{cases}0, \text{ if } \max\limits_{\forall l:y_l=1}\{\sigma_l(x)\}-\sigma_j(x)>\delta_\sigma,\\ 1 \text{ otherwise,}\end{cases}$$

$$\chi_j^\omega(x,y)=\begin{cases}0, \text{ if } \max\limits_{\forall l:y_l=1}\{\omega_l(x)\}-\omega_j(x)>\delta_\omega,\\ 1 \text{ otherwise,}\end{cases}$$

$$\chi_j^\phi(x,y,z)=\begin{cases}0, \text{ if } \max\limits_{\forall l:y_l=1}\{\phi_l(x,z)\}-\phi_j(x,z)>\delta_\phi,\\ 1 \text{ otherwise,}\end{cases}$$

where $y_j=\chi_j(x)$, $j\in\{1,2,...,m\}$.

These functions allow implementing the various variants of the additional job classification based on the primary one. The job classification is intended for the primary filtration of the resources set. It provides forming the residual set of resources for the job execution. The further filtration of resources from the set V is implemented with help of the lexicographical or majority methods of choice [23].

We use the lexicographic method with the following modified rule of choice:

$$V^*=\{v_s:(\forall v_l\in V\ \exists p:(\hat{q}_{1,s}=\hat{q}_{1,l})\wedge...\wedge(\hat{q}_{p,s}=\hat{q}_{p,l})\wedge(\hat{q}_{p+1,s}>\hat{q}_{p+1,l}))\},$$

where V^* is the filtered set of resources, $v_s\in V$, n_v is the number of resources, q_{js} is the characteristic of the sth resource, $\hat{q}_{js}$ is its estimation, n_q is the number of compared characteristics, $p\in\{1,2,...,n_q-1\}$, $j=1,2,...,n_q$, $s\in\{1,2,...,n_v\}$, $l\in\{1,2,...,n_v\}$ and $v\neq l$.

We use the majority method with the following modified rule of choice:

$$V^*=\left\{v_s:\left(\neg\exists v_l\in V:\sum_{j=1}^{n_q}\text{sign}(\hat{q}_{jl}-\hat{q}_{js})>0\right)\right\},$$

where $\text{sign}(0)=0$.

To estimate the values q_{js}, $s=1,2,...,n_v$, their set is divided into subsets that do not intersect pairwise. They are ordered by ascending or descending. Accordingly, each subset receives its index used as an estimate of the values belonging to the given subset. If the resulting set V^* contains more than one element, then the final choice of the single resource v_s is done randomly.

4. Experimental analysis

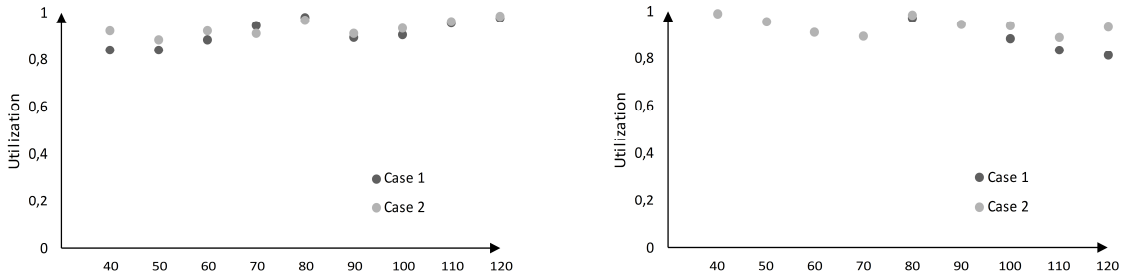
Within experimental analysis, we performed simulation modeling of the ICE using the GPSS World system [24]. The examples of the utilization of ICE nodes using the hypervisor shell and without it are given bellow.

Simulation modeling was performed on the job flow corresponding to the flow of real jobs executed on the cluster that includes 20 nodes. Each node has two quad-core processors Intel Xeon E5345. We processed over 60000 jobs in total.

We varied dedicated and non-dedicated nodes of the cluster in various proportions. The dedicated nodes execute virtual machines and do not execute jobs from common queues of the local resource managers. The non-dedicated nodes execute all jobs, including jobs for executing virtual machines.

The traditional manager of a virtual environment does not accept jobs for execution of virtual machines, whose requirements exceed the possibilities of environment nodes. In contrast, the hypervisor shell allows decomposing such jobs and passing its part to non-dedicated nodes.

Fig. 3 show the utilization of dedicated and non-dedicated nodes under management, respectively, using the hypervisor shell (a) and without it (b).



a) dedicated nodes b) non-dedicated nodes

Fig. 3. Utilization of nodes

The represented results show that the hypervisor shell allows improving the utilization of dedicated nodes by means of allocation of non-dedicated nodes for executing virtual machines. Therefore, the utilization of non-dedicated nodes also increases in many cases.

In the future, we will study the ICE in more detail in the process of its practical use.

Conclusion

In this paper, the approach to the integration and virtualization of heterogeneous clusters in a cloud computing environment was presented. A brief overview of the traditional tools for the management of virtual machines was given; the advantages of the OpenStack platform for an interaction with these hypervisors were highlighted.

Our contribution is multifold. We proposed the technology for the heterogeneous cluster environment virtualization using the OpenStack platform together with the new specialized hypervisor shell for local resource managers such as PBS, SLURM, or SGE. Within this approach, we developed the model of the resource allocation for virtual machines and realized the tools for the integration of heterogeneous clusters.

The developed model of the resource allocation for virtual machines allowed us to use the knowledge about job requests, resource characteristics and current state of the environment, and the expertise of it administrators.

The realized tools provide the capability for the “painless” integration of heterogeneous clusters with the preinstalled local resource managers listed above for creating the virtual cluster with the required configuration.

The practicability and benefits of the represented approach were demonstrated on the model example. All experiments were carried out in the Center of collective usage SB RAS "Irkutsk supercomputer center".

The study was partially supported by RFBR, projects no. 15-29-07955 and no. 16-07-00931, and Program 1.33P of fundamental research of Presidium RAS, project "Development of new approaches to creation and study of complex models of information-computational and dynamic systems with applications".

References

1. Gergel V., Senin A. Metacluster System for Managing the HPC Integrated Environment. Methods and Tools of Parallel Programming Multicomputers. Second Russia-Taiwan Symposium, MTPP 2010 (Vladivostok, Russia, May 16–19 2010). LNCS, vol. 6083. pp. 86–94. DOI: 10.1007/978-3-642-14822-4
2. Mladen A., Eric S., Patrick D. Integration of High-Performance Computing into Cloud Computing Services. Handbook of Cloud Computing. 2010. pp. 255-276. DOI: 10.1007/978-1-4419-6524-0_11
3. Bychkov I.V., Oparin G.A., Novopashin A.P., Feoktistov A.G., Korsukov A.S., Sidorov I.A. High-performance computing resources of ISDCT SB RAS: State-of-the-art, prospects and future trends. Comput. Tech. 2010. vol. 15. pp. 69–81. (in Russian).
4. Bogdanova V.G., Bychkov I.V., Korsukov A.S., Oparin G.A., Feoktistov A.G. Multiagent Approach to Controlling Distributed Computing in a Cluster Grid System. J. Comput. Syst. Sci. Int. 2014. vol. 53. pp. 713–722. DOI:10.1134/S1064230714040030
5. Bychkov I.V., Oparin G.A., Feoktistov A.G., Bogdanova V.G., Pashinin A.A. Service-oriented Multiagent Control of Distributed Computations. Automat. Rem. Contr. 2015. vol. 76. pp. 2000–2010. DOI: 10.1134/S0005117915110090
6. Bychkov I.V., Oparin G.A., Feoktistov A.G., Sidorov I.A., Bogdanova V.G., Gorsky S.A. Multiagent Control of Computational Systems on the Basis of Meta-monitoring and Imitational Simulation. Optoelectron., Instr. and Data Process. 2016. vol. 52, pp. 107–112. DOI: 10.3103/S8756699016020011
7. Irkutsk Supercomputer Center of SB RAS. Available at: http://hpc.icc.ru (accessed: 16.02.2017).
8. Buyya R., Broberg J., Goscinski A.M. Cloud Computing: Principles and Paradigms. Wiley, 2011. 637 p. DOI: 10.1002/9780470940105
9. Sridharan S. A Performance Comparison of Hypervisors for Cloud Computing. University of North Florida, 2012. 269 p.
10. Docker. Available at: http://docker.com (accessed: 16.02.2017).
11. QEMU. Available at: http://qemu.org (accessed: 16.02.2017).
12. KVM. Available at: http://www.linux-kvm.org (accessed: 16.02.2017).
13. Xen. Available at: http://cam.ac.uk/research/srg/netos/projects/archive/xen (accessed: 16.02.2017).
14. vSphere ESXi. Available at: https://vmware.com/support/vsphere-hypervisor.html (accessed: 16.02.2017).
15. Bumgardner V.K. OpenStack in Action. Manning Publications, 2016. 358 p.
16. Apache CloudStack. Available at: https://cloudstack.apache.org/ (accessed: 16.02.2017).

17. Euacalyptus. Available at: http://www.eucalyptus.com/ (accessed: 16.02.2017).
18. OpenNebula. Available at: https://opennebula.org (accessed: 16.02.2017).
19. Bichkov I.V., Oparin G.A., Novopashin A.P., Sidorov I.A. Agent-Based Approach to Monitoring and Control of Distributed Computing Environment. Parallel Computing Technologies: 13th International Conference, PaCT 2015 (Petrozavodsk, Russia, August 31-September 4). LNCS, vol. 9251. pp. 253–257. DOI: 10.1007/978-3-319-21909-7_24
20. Sidorov I.A. Methods and Tools to Increase Fault Tolerance of High-performance Computing Systems. In proc. of the 39th International Convention on information and communication technology, electronics and microelectronics, MIPRO-2016 (Opatija, Croatia, 30 May – 3 June 2016). Riejka: CSICTEM 2016. pp. 242–246. DOI: 10.1109/MIPRO.2016.7522142
21. Feoktistov A.G, Sidorov I.A. Logical-Probabilistic Analysis of Distributed Computing. In proc. of the 39th International Convention on information and communication technology, electronics and microelectronics, MIPRO-2016 (Opatija, Croatia, 30 May-3 June 2016). Riejka: CSICTEM 2016. pp. 247–252. DOI: 10.1109/MIPRO.2016.7522142
22. Hastie T., Tibshirani R., Friedman J. The Elements of Statistical Learning: Data Mining, Inference, and Prediction. 2001, 533 p.
23. Sholomov L.A. *Logicheskie metodi issledovaniya diskretnih modelei vibora* [Logical Research Methods of Discrete Choice Models]. Moscow: Nauka, 1989, 288 p. (in Russian) GPSS World Tutorial Manual. Available at: http://www.minutemansoftware.com (accessed: 16.02.2017).

УДК 004.45, 004.386, 004.382.2 **DOI: 10.14529/cmse170203**

ВИРТУАЛИЗАЦИЯ HPC-КЛАСТЕРОВ С ПОМОЩЬЮ ПЛАТФОРМЫ OPENSTACK

© 2017 г. **А.Г. Феоктистов, И. А. Сидоров, В.В. Сергеев, Р.О. Костромин, В.Г. Богданова**

Институт динамики систем и теории управления имени В.М. Матросова СО РАН (664033, Иркутск, Лермонтова, 134),

E-mail: agf@icc.ru, ivan.sidorov@icc.ru, vvsergeev@mail.ru, kostromin@icc.ru, bvg@icc.ru

Поступила в редакцию: 01.05.2017

Статья посвящена проблеме интеграции разнородных вычислительных кластеров в единую среду на основе технологий виртуализации. В качестве платформы управления виртуальной средой выбран программный комплекс OpenStack. Данный комплекс обеспечивает широкий набор компонентов и функциональных решений для взаимодействия с различными гипервизорами. В их числе KVM, XEN, ESXi, QEMU и другие системы. В дополнение к программному комплексу OpenStack разработана специализированная оболочка гипервизора, обеспечивающая запуск виртуальных машин из очередей традиционных систем управления заданиями, например, PBS, SLURM, LSF или SGE, используемых на кластерах суперкомпьютерного центра коллективного пользования. Разработанная модель распределения ресурсов для виртуальных машин позволяет использовать знания о заданиях, характеристиках ресурсов и текущем состоянии виртуальной вычислительной среды, а также опыт ее администраторов. Реализованные инструменты обеспечивают возможность «безболезненной» интеграции разнородных вычислительных кластеров, использующих различные системы управления заданиями, в виртуальный кластер с требуемой конфигурацией. Модельные эксперименты показывают, что оболочка гипервизора позволяет повысить

коэффициент полезного использования узлов интегрированной среды путем переназначения виртуальных машин в очереди традиционных систем управления заданиями.

Ключевые слова: вычислительные кластеры, высокопроизводительные вычисления, технологии виртуализации, OpenStack, имитационное моделирование

ОБРАЗЕЦ ЦИТИРОВАНИЯ

Feoktistov A.G, Sidorov I.A., Sergeev V.V., Kostromin R.O., Bogdanova V.G. Virtualization of heterogeneous HPC-clusters based on OpenStack platform // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2017. Т. 6, № 2. С. 37–48. DOI: 10.14529/cmse170203.

Литература

1. Gergel V., Senin A. Metacluster System for Managing the HPC Integrated Environment // Methods and Tools of Parallel Programming Multicomputers. Second Russia-Taiwan Symposium, MTPP 2010 (Vladivostok, Russia, May 16-19 2010). LNCS, Vol. 6083. P. 86–94. DOI: 10.1007/978-3-642-14822-4
2. Mladen A., Eric S., Patrick D. Integration of High-Performance Computing into Cloud Computing Services // Handbook of Cloud Computing. 2010. P. 255–276. DOI: 10.1007/978-1-4419-6524-0_11
3. Бычков И.В., Опарин Г.А., Новопашин А.П., Феоктистов А.Г, Корсуков А.С., Сидоров И.А. Высокопроизводительные вычислительные ресурсы Института динамики систем и теории управления со ран: текущее состояние, возможности и перспективы развития // Вычислительные технологии 2010. Т. 15. С. 69–81.
4. Bogdanova V.G., Bychkov I.V., Korsukov A.S., Oparin G.A., Feoktistov A.G. Multiagent Approach to Controlling Distributed Computing in a Cluster Grid System // J. Comput. Syst. Sci. Int. 2014. Vol. 53. P. 713–722. DOI:10.1134/S1064230714040030
5. Bychkov I.V., Oparin G.A., Feoktistov A.G., Bogdanova V.G., Pashinin A.A. Service-oriented multiagent control of distributed computations // Automat. Rem. Contr. 2015. Vol. 76. P. 2000–2010. DOI: 10.1134/S0005117915110090
6. Bychkov I.V., Oparin G.A., Feoktistov A.G., Sidorov I.A., Bogdanova V.G., Gorsky, S.A. Multiagent Control of Computational Systems on the Basis of Meta-monitoring and Imitational Simulation // Optoelectron., Instr. and Data Process. 2016. Vol. 52, P. 107–112. DOI: 10.3103/S8756699016020011
7. Иркутский суперкомпьютерный центр СО РАН. URL: http://hpc.icc.ru (accessed: 16.02.2017).
8. Buyya R., Broberg J., Goscinski A.M. Cloud Computing: Principles and Paradigms. Wiley, 2011. 637 p. DOI: 10.1002/9780470940105
9. Sridharan S. A Performance Comparison of Hypervisors for Cloud Computing. University of North Florida, 2012. 269 p.
10. Docker. URL: http://docker.com (дата обращения: 16.02.2017).
11. QEMU. URL: http://qemu.org (дата обращения: 16.02.2017).
12. KVM. URL: http://www.linux-kvm.org (дата обращения: 16.02.2017).
13. Xen. URL: http://cam.ac.uk/research/srg/netos/projects/archive/xen (дата обращения: 16.02.2017).
14. vSphere ESXi. URL: https://vmware.com/support/vsphere-hypervisor.html (дата обращения: 16.02.2017).
15. Bumgardner V.K. OpenStack in Action. Manning Publications, 2016. 358 p.
16. Apache CloudStack. URL: https://cloudstack.apache.org/ (дата обращения: 16.02.2017).

17. Euacalyptus. URL: http://www.eucalyptus.com/ (дата обращения: 16.02.2017).
18. OpenNebula. URL: https://opennebula.org (дата обращения: 16.02.2017).
19. Bichkov I.V., Oparin G.A., Novopashin A.P., Sidorov I.A. Agent-Based Approach to Monitoring and Control of Distributed Computing Environment // Parallel Computing Technologies: 13th International Conference, PaCT 2015 (Petrozavodsk, Russia, August 31 – September 4 2015). LNCS, Vol. 9251. P. 253–257. DOI: 10.1007/978-3-319-21909-7_24
20. Sidorov I.A. Methods and Tools to Increase Fault Tolerance of High-performance Computing Systems // In proc. of the 39th International Convention on information and communication technology, electronics and microelectronics, MIPRO-2016 (Opatija, Croatia, 30 May-3 June 2016). Riejka: CSICTEM 2016. P. 242–246. DOI: 10.1109/MIPRO.2016.7522142
21. Feoktistov A.G, Sidorov I.A. Logical-Probabilistic Analysis of Distributed Computing // In proc. of the 39th International Convention on information and communication technology, electronics and microelectronics, MIPRO-2016 (Opatija, Croatia, May 30 – June 3 2016). Riejka: CSICTEM 2016. P. 247–252. DOI: 10.1109/MIPRO.2016.7522142
22. Hastie T., Tibshirani R., Friedman J. The Elements of Statistical Learning: Data Mining, Inference, and Prediction. 2001, 533 p.
23. Шоломов Л.А. Логические методы исследования дискретных моделей выбора. М.: Наука, 1989. 288 с.
24. GPSS World Tutorial Manual. URL: http://www.minutemansoftware.com (дата обращения: 16.02.2017).

Феоктистов Александр Геннадьевич, к.ф.-м.н., доцент, лаборатория параллельных и распределенных вычислительных систем, Институт динамики систем и теории управления им. В.М. Матросова СО РАН (Иркутск, Российская Федерация)

Сидоров Иван Александрович, к.ф.-м.н., доцент, лаборатория параллельных и распределенных вычислительных систем, Институт динамики систем и теории управления им. В.М. Матросова СО РАН (Иркутск, Российская Федерация)

Сергеев Вадим Викторович, аспирант, лаборатория параллельных и распределенных вычислительных систем, Институт динамики систем и теории управления им. В.М. Матросова СО РАН (Иркутск, Российская Федерация)

Костромин Роман Олегович, аспирант, лаборатория параллельных и распределенных вычислительных систем, Институт динамики систем и теории управления им. В.М. Матросова СО РАН (Иркутск, Российская Федерация)

Богданова Вера Геннадьевна, к.ф.-м.н., доцент, лаборатория параллельных и распределенных вычислительных систем, Институт динамики систем и теории управления им. В.М. Матросова СО РАН (Иркутск, Российская Федерация)

DOI: 10.14529/cmse170204

PARALLEL ALGORITHMS FOR EFFECTIVE CORRESPONDENCE PROBLEM SOLUTION IN COMPUTER VISION*

© 2017 г. S.A. Tushev, B.M. Sukhovilov
South Ural State University
(Russian Federation, 454080 Chelyabinsk, 76 Lenin avenue)
E-mail: science@tushev.org, sukhovilovbm@susu.ru
Received: 01.05.2017

We propose new parallel algorithms for correspondence problem solution in computer vision. We develop an industrial photogrammetric system that uses artificial retroreflective targets that are photometrically identical. Therefore, we cannot use traditional descriptor-based point matching methods, such as SIFT, SURF etc. Instead, we use epipolar geometry constraints for finding potential point correspondences between images. In this paper, we propose new effective graph-based algorithms for finding point correspondences across the whole set of images (in contrast to traditional methods that use 2-4 images for point matching). We give an exact problem solution via superclique and show that this approach cannot be used for real tasks due to computational complexity. We propose a new effective parallel algorithm that builds the graph from epipolar constraints, as well as a new fast parallel heuristic clique finding algorithm. We use an iterative scheme (with backprojection of the points, filtering of outliers and bundle adjustment of point coordinates and cameras' positions) to obtain an exact correspondence problem solution. This scheme allows using heuristic clique finding algorithm at each iteration. The proposed architecture of the system offers a significant advantage in time. Newly proposed algorithms have been implemented in code; their performance has been estimated. We also investigate their impact on the effectiveness of the photogrammetric system that is currently under development and experimentally prove algorithms' efficiency.

Keywords: computer vision, photogrammetry, correspondence problem, parallel algorithms, maximum clique problem, epipolar geometry.

FOR CITATION

Tushev S.A., Sukhovilov B.M. Parallel Algorithms for Effective Correspondence Problem Solution in Computer Vision. *Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering.* 2017. vol. 6, no 2. pp. 49–68. DOI: 10.14529/cmse170204.

Introduction

Effective matching between images of a 3D object points (also known as point correspondence problem) is one of the key problems in computer vision [1]. There are several ways to solve the correspondence problem, such as feature detection, a method based on the epipolar geometry restraints, and the combination of these two methods. Feature detection is the most popular approach for solving the point correspondence problem. These methods analyze images and look for *features*, such as corners, ridges, contrast points etc. A numerical value, called *descriptor*, is calculated from the image data based on some vicinity of the feature. This approach allows us to find point correspondences almost instantaneously by matching the numerical values of descriptors, as local feature detectors are robust against most image transformations (caused by movement of the camera) given that lighting conditions remain the

* The paper is recommended for publication by the Program Committee of the International Scientific Conference "Parallel Computational Technologies (PCT) 2017".

same and the camera position does not change drastically. Another advantage is the fact that feature detection does not require any knowledge of the camera relative position.

Unfortunately, feature detection methods only work well in stable lighting conditions. If the light source moves, features might alter their appearance, structure of shadows, or disappear. This leads to changes in numerical values of the descriptors that make it impossible to find matches. Harris detector [2], SIFT [3] and SURF [4] are among the most popular and widely used feature detectors and descriptors.

It is also possible to solve correspondence problem using epipolar geometry [1, 5]. As it is shown in Section 1.1, if we choose an image point in the first image, the corresponding image point in the second image may only lie on the epipolar line [1]. Epipolar line can be easily calculated, provided that we know camera intrinsic parameters with the relative position and orientation of a pair of cameras. Thus, it is possible to narrow the search area to a single epipolar line (for a pair of images), or even to the crossing point of two epipolar lines (for a triplet of images). An approach based on epipolar geometry does not have any special requirements to lightning conditions of the image contents; however, it requires cameras' parameters and their relative position to be known. Fraser in [5] describes various approaches of finding point correspondence based on epipolar geometry in 2- and 3-dimensional space. He also states that computational difficulty grows rapidly when the number of images is increased.

A group of special (typically industrial) photogrammetric systems uses retroreflective targets (to identify the key points of the object being measured) and a flash to ensure that those targets are easily distinguishable in an industrial environment that often has insufficient lighting conditions. This directed light drastically changes the structure of shadows in the image, thus changing the values of feature descriptors. In this case, point correspondence problem can only be solved by means of epipolar geometry. Unfortunately, in practice the point may lie slightly out of the calculated epipolar line due to some degree of uncertainty in estimated relative camera position or due to camera's manufacturing imperfections. Thus in real-life applications it is better to search for a point in some δ-vicinity of the epipolar line. Moreover, as the number of images and targets increases, the search area may contain more other points or artefacts (such as glares or reflections mistakenly recognized by the system as retroreflective targets), so the correspondence problem becomes non-trivial. As we show in Sections 1.2 and 1.3, the exact solution of point correspondence problem belongs to clique problem in a multipartite graph and has exponential complexity.

We develop an industrial photogrammetric system that uses retroreflective targets [6, 7], so we have to use epipolar geometry to solve the point correspondence problem. Our system should work on a mid-high level laptop. Moreover, due to specifications, the whole process of the 3D reconstruction (from uploading images from the camera to obtaining accurate 3D coordinates of the targets) should take no more than 5-6 minutes. This requires us to develop some new effective point matching algorithms based on epipolar geometry, which would allow us to solve the problem in acceptable time. To achieve this, we use parallelization along with an iterative scheme that adjusts the accuracy for estimation of targets and cameras' positions. This allows using heuristic clique finding algorithms (particularly but not exclusively their parallel versions) that finally lead us to an exact solution of the point correspondence problem in acceptable time.

It is impossible to parallelize the whole problem of 3D reconstruction due to its specific nature. However, we may significantly increase overall efficiency of the process by performing

parallel data processing at several stages, thanks to the widespread parallel architecture (nowadays most of the mid-high level laptops contain multi-core CPUs, as well as discrete GPUs).

In this paper we will cover different aspects of the problem. We start with considering the theoretical solution of the point correspondence problem by the means of epipolar geometry (Sections 1.1-1.3) and estimating its complexity (Section 3.1). Section 1.4 describes our iterative scheme that allows us to use near-polynomial time clique finding algorithms that are described in section 1.5. Section 2 covers our software implementation of the algorithms. Test results are provided in Section 3. Our findings are summarized in the final section, "Conclusion and future work".

Related work. Point matching is one of the key tasks in computer vision. Most works rely on having photometrically distinct features that allow computing a descriptor from image data [8, 9]. SIFT [3] and SURF [4] are among the most popular and widely used methods. We can name BRISK [8] and FREAK [11] among the developments of the recent years. There are also works that consider hybrid approach, such as [12]

However, due to the usage of photometrically identical targets, we should rely solely on epipolar geometry. One of the most fundamental works that cover multiple aspects of computer vision, including epipolar geometry, is [1]. There are two primary approaches to finding point correspondences via epipolar geometry: so-called clustering method, which operates within 3D space, and plane-based methods. Various techniques that utilize the clustering method are considered in [13, 14]. Among the papers that describe the plane-based approach, we can name both classical works by Maas [15, 16], Zhang et al. [17], as well as recent papers, such as [18, 19]. Fraser in [20] describes "presently adopted approaches for close-range photogrammetric network orientation, along with three categories of processing for 3D point determination".

Modern photogrammetry systems, such as V-STARS [21] or Agisoft PhotoScan [22] are likely to use similar approaches to finding point correspondences. However, due to commercial considerations, very little information concerning the internal target detection mechanisms, point matching algorithms etc. is available. Among the few works that consider various aspects of photogrammetry systems we can name [5] as the closest one to our photogrammetric system.

In this paper, we propose new effective graph-based algorithms that utilize epipolar geometry for finding point correspondences across the whole set of images. This is, to the best of our knowledge, a new approach; other techniques that utilize epipolar geometry for point matching are based on 2, 3 or 4 images [15, 23].

An extension of 4-images approach to an arbitrary number of images has been reported by Dold and Maas in [16], although they state high computational complexity of their method that "grows exponentially with the number of images and would hardly be tolerable in an application with 18 images". However, our method, despite utilizing similar principles of epipolar geometry, is different: we use a graph as a mathematical model of point correspondence problem, so that each clique in the graph represent a potential matched point across all images in the set. Thus we can "automatically" get all point matches, by choosing the largest disjoint cliques from the graph. We also consider additional information such as clique weight to choose the best candidates for establishing point correspondence. While we encountered the same exponential computational complexity within our superclique approach, we developed another approach that allows to find exact point correspondences across the whole range of images in a reasonable time. It includes our new parallel graph-based algorithms along with an iterative

scheme. Thus it makes it possible to find an exact solution of the point correspondence problem for hundreds of images in a fast and efficient way.

Another approach that utilizes epipolar geometry, tree hierarchy, graph theory and clustering was reported in [24].

1. Theory

1.1. Epipolar geometry and point correspondence

Epipolar geometry describes the so-called stereoscopic pair – two cameras with known relative pose and orientation that observe the same three-dimensional object.

Suppose that point X is simultaneously observed by two cameras: the left camera with optical center O_L and the right camera with optical center O_R (Fig. 1). The real (three-dimensional) point X is projected on the left and right image planes as X_L and X_R correspondingly. The baseline, which connects cameras' centers O_L and O_R, intersects with image planes in points e_L and e_R, which are known as epipoles [1].

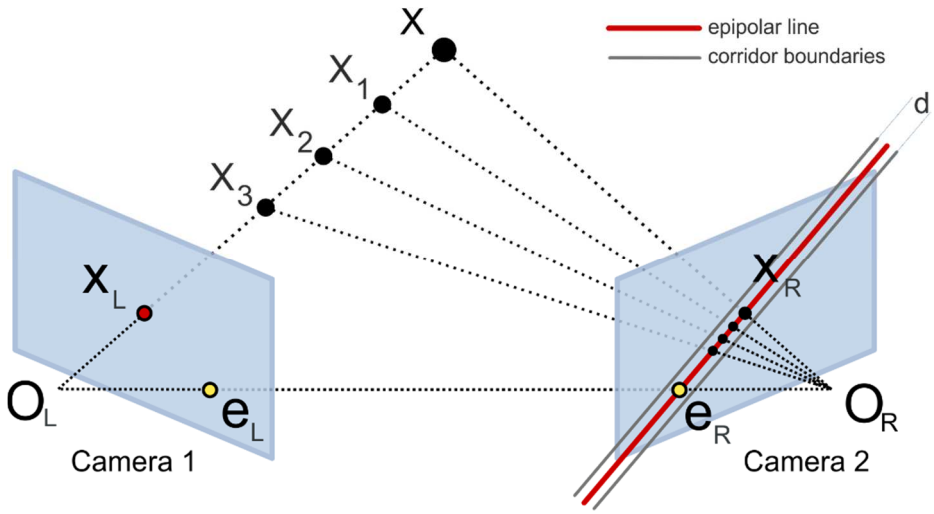


Fig. 1. Epipolar geometry and epipolar corridors

Apparently, each 3D point X has its own epipolar plane that goes through baseline and through X (thus through X_L as well). The intersection of the epipolar plane with the right image plane forms an epipolar line that goes through X_R.

Given a case where real 3D coordinates of X are unknown (for instance, processing a set of images when we have only two-dimensional coordinates of X_L), we cannot unambiguously locate X_R on the right image due to the loss of depth when projecting the point to an image plane. However, it is still possible to narrow the search area of the corresponding point X_R from the whole image to epipolar line.

An epipolar line (in the form of $ax + by + c = 0$) that corresponds to X_L in the right image can be described as (1), according to [25]:

$$e = F \cdot X_L, \tag{1}$$

where e is the column vector of coefficients a, b and c, X_L is the column of homogeneous point coordinates in the left image, F is a fundamental matrix calculated for the given pair of images by camera intrinsic parameters K and camera relative pose and orientation, which are represented by essential matrix E [1]:

$$F = K^{-1} \cdot E \cdot K, \tag{2}$$

Essential matrix E can be calculated as (3):

$$E = \begin{bmatrix} 0 & -tx(3) & tx(2) \\ tx(3) & 0 & -tx(1) \\ -tx(2) & tx(1) & 0 \end{bmatrix} \cdot R, \tag{3}$$

where tx is the normalized translation vector between the cameras' centers. Given the pair of translation vectors t_l and $\mathrm{t_r}$ which define location of the cameras' centers in some global coordinate system, we can calculate tx from (4) and (5):

$$tx = \frac{tx'}{\sqrt{\|tx'\|}}, \tag{4}$$

$$tx' = t_r^T - R \cdot t_l^T, \tag{5}$$

where R is the rotation matrix between the two cameras. It can be found by (6) from a pair of rotation matrices A_l и A_r in some global coordinate system:

$$R = A_r \cdot A_l^T. \tag{6}$$

It is evident that epipolar line on the right image can go through several different image points, thus making the point correspondence problem to be non-trivial.

In order to use the epipolar geometry, camera poses and orientations should be known before making calculations. In our photogrammetric system we use so-called *coded targets* that provide us with the global coordinate system for all images [26].

Unfortunately, real cameras are not as perfect as their mathematical models are. In the right picture, the point may lie slightly off the calculated epipolar line due to some degree of uncertainty in the estimated relative camera position or due to the camera's manufacturing imperfections. Thus in practice it is better to use the *epipolar corridor* of width d that is more likely to contain the point.

Switching from epipolar lines to epipolar corridors brings even more nearby candidate image points into the correspondence search area. Moreover, some of the images may contain extra artefacts inside of the corridor (such as glares or reflections mistakenly recognized by the system as retroreflective targets). Thus, point matching algorithm should be robust and insensitive to various disturbances and artefacts.

1.2. Epipolar geometry and point correspondence

We propose to use the multipartite graph G={V;E} as a mathematical model of the point correspondence problem. Graph G consists of a set of image points V (graph *vertices*) and a set of possible correspondences between points E (graph *edges*). A pair of points is considered adjacent if each point of the pair lies within the epipolar corridor calculated for the opposite point of the pair. Each single image forms a separate part of the graph (because vertices that represent points in the same image cannot be adjacent), so the whole graph becomes k-partite, where k is the number of images.

The algorithm for building the graph from epipolar data is given below. For each pair of images we calculate fundamental matrix F using (2) — (6). Then we process each pair of image points in the "*left*" and "*right*" images of the pair. For each point of the pair we calculate epipolar line on the "opposite" image (the one that the point does not belong to) and Euclidian distance to this line for the other point of the pair. When the mean pixel distance to the

epipolar lines (7) is not greater than the half of the allowed width of the epipolar corridor, two vertices in G become adjacent with the edge weight w, equal to (7):

$$w = \frac{1}{2}\left(\frac{|e_l \cdot p_r|}{\sqrt{e_l(1)^2+e_l(2)^2}} + \frac{|e_r \cdot p_l|}{\sqrt{e_r(1)^2+e_r(2)^2}}\right), \quad (7)$$

Here e_l, e_r are the column vectors of coefficients (1) for epipolar lines in the left and the right image correspondingly; p_l, p_r are homogeneous pixel coordinates of image points in the left and the right images (8):

$$p = \begin{bmatrix} x_{px} \\ y_{px} \\ 1 \end{bmatrix}, \quad (8)$$

This algorithm has a large number of independent calculations and possesses high cyclomatic complexity, which makes it a good candidate for parallelization. We describe our parallel implementation of this algorithm in Section 2.1.

1.3. Epipolar geometry and point correspondence

The exact, theoretical solution of the point correspondence problem is a superclique – a clique in a supergraph, a graph where each vertex represents a clique in graph G (a clique is a subset of graph where every two distinct vertices are adjacent).

At the first step, we should initially find all maximal cliques in G with the number of vertices greater than or equal to some pre-defined value t (which is used to filter out artefacts). Each clique $C_i = \{V_i\}$ represents a possible 3D candidate point; its vertices V_i represent possible images of this point in various pictures that satisfy epipolar restraints. Besides the set of vertices, each clique obtains its unique identifier.

At the next step, we build a supergraph – graph S with vertices C_i. A pair of cliques C_i and C_j are adjacent in S if they do not have any common vertices V_i (i.e. they are disjoint: $C_i \cap C_j \equiv \{V_i\} \cap \{V_j\} = \emptyset$)

Superclique is the maximum clique of S that consists of disjoint cliques $\{C_i\}$ and represents the final set of 3D points. At this step each Ci from superclique represents an actual 3D point identified across all the pictures in the set. The superclique is the exact solution of the graph-based point correspondence problem. This approach requires two steps.

We have introduced t before as a parameter that controls filtering various artefacts, such as glares, blinks etc. It is assumed that those disturbances are visible in no more than *t–1* pictures in the series. If a bright point that resembles retroreflective target is visible in *t* or more images, it can be considered as an additional 3D feature that can be used to improve accuracy of the computations. The value of t partially depends on shape of object being measured. In our work we typically use t = 4 ... 6.

Unfortunately, the usage of the superclique approach in practice is limited due to computational complexity. Nowadays both the maximum and the maximal clique problems are considered to be NP-complete [27]. While listing all maximal cliques in G is hard, finding maximum clique in S is much harder (as S may contain much more edges and vertices than G). We provide some experimental data and estimations in Section 3.1.

According to our research, at the current level of technology it is impossible to use the superclique approach for practical tasks, except for cases with low number of targets and cameras that are not of any practical industrial interest.

1.4. Exact solution of the problem using an iterative scheme

Our photogrammetric system must perform measurements in a specified amount of time. Our tests show that the superclique approach requires large amounts of time, which exceed the acceptable limits even for small-sized datasets.

In order to meet the time requirements we develop an iterative scheme for finding point correspondences that uses approximate superclique finding methods, triangulation of 3D points with filtering of outliers, bundle adjustment [28] of point and cameras' parameters with backprojection techniques that allow finding new point images in the pictures. This scheme makes it possible to use approximate clique finding algorithms of near-polynomial time complexity at every single iteration. Eventually, this scheme provides the exact solution of the point correspondence problem (and estimation of their 3D coordinates) in a reasonable amount of time.

1.5. Parallel local graph-based algorithm for finding correspondence points

We propose a new parallel algorithm for finding point correspondence. The key idea of this algorithm is building a number of small *local* graphs instead of processing one large *global* graph. This is the further development of our PLG family of algorithms, which are described in [29], with the consideration of gathered experience and criticism.

General idea. Most algorithms of this family process the whole set of point images in series. At each iteration, algorithm builds a small-sized local graph for the current image point, which becomes the *seeding* vertex, or seeding point (*SP*). All other image points that lie within the corresponding epipolar corridors in all other pictures are also added to the local graph as vertices, adjacent to the seeding point (*NP*). The corresponding edge weight is calculated by equation (7). In the next step, the algorithm analyses epipolar geometry of the newly added point images and adds more edges to the graph between existing vertices. Some members of the PLG family even expand the graph at this step by adding new vertices that are connected to *NP* (but not to *SP*). Finally, the algorithm finds the maximum clique in the graph. Vertices that form this maximum clique represent the same 3D point across all pictures [12].

An intuitive (geometrical) explanation of the working principles of the point matching algorithms that are based on local graphs can be formulated as the following:

1. An image point (*SP*) is picked in a certain way from the set of all image points across all pictures;
2. Epipolar lines in all other images are calculated for *SP*;
3. All other image points from other pictures that lie within some δ-vicinity of the epipolar lines are considered as potential corresponding points;
4. New epipolar line across all other images are calculated for each of the potential corresponding points;
5. In each image, we choose an image point that lies at the crossing point of the maximum number of epipolar lines (or in some δ-vicinity of the crossing point);
6. The set of chosen image points represents the same 3D point with the maximum degree of probability (which is proven by the experiments).

Our method conceptually resembles point matching between a triplet of images [1, 5], however, it takes more data to consider (we analyze all pictures that contain the necessary

scene fragment). Thus, potentially, it may achieve more accurate point matching than triplet-based methods. Our experiments have proven this method to be robust and accurate.

Different algorithms in PLG family pick the seeding point in a different way. At present, we consider the algorithm called PLG4 to be the most optimal. At every iteration, it picks the vertex with the highest degree in the global graph (i.e. the image point with the maximum number of candidate image points in the proximity of its epipolar lines), which has not been included to any cliques before, as the seeding vertex for the current local graph. The local graph in PLG4 must contain only vertices adjacent to *SP*. We consider this algorithm as more accurate rather than PLG1 or PLG2 described in [29], thus our parallel point correspondence algorithm will be based upon these principles.

Our parallel point correspondence algorithm. Most algorithms in PLG family work with the converging set of image points (because image points that have formed a clique are excluded from consideration). This feature makes data to be dependent from the previous step, thus making parallelization of this algorithm more difficult. Besides, the exclusion of image point from further analysis it could potentially lead to the loss of other cliques of the same size, but with lesser summary weight of edges, which are more preferable.

```
parallel_for_each (point ∈ 2DPoints)
  G_p := ∅
  for_each neighbour ∈ point.Targets
    G_p ← vertex(neighbour)
    G_p← edge(point ↔ neighbour)
  end_for_each
  for_each (neighbour ∈ point.Targets)
    for_each (link ∈neighbour.Targets)
      if (G_p ∋ link as vertex)
        G_p← edge(link ↔ neighbour)
      endif
    end_for_each
  end_for_each
  grouped_point := find_maximum_clique_of_min_weight(G_p)
  if (size(grouped_point) ≥ t )
    { cliques } ← grouped_point
  endif
end_parallel_for_each
cliques := sort_by_size (cliques)
foreach clique ∈ cliques
  foreach clique2 ∈cliques[clique...end]
    if (clique ⋂ clique2 ≠ ∅)
      cliques := cliques \ clique2
    endif
  end_for_each
end_for_each
```

Fig. 2. Parallel PLG algorithm pseudo-code

In our new algorithm, we decided not to exclude the image points that have formed cliques from further processing, so the input dataset does not converge with time. This not only simplifies parallelization of algorithms, but also allows creating a brand new approach based

on local graphs as well. A sequential implementation of this approach is also possible, although it is far less effective.

Therefore, our parallel algorithm for finding point correspondence consists of two stages. At the first stage, we build local graphs for *all* image points in a parallel. The algorithm also tries to find maximum clique in each local graph that it builds. Once the parallel stage is complete, we refine the resulting set of cliques to get rid of the clique intersections. Our new algorithm is provided in pseudo-code given in Fig. 2.

`find_maximum_clique_of_min_weight` procedure searches for the maximum clique in G_p, choosing the one with lesser summary edge weight in case of two cliques of the same size being compared. From practical point of view, clique with lesser summary edge weight has it vertices located closer to the corresponding epipolar lines, thus being more preferable over the clique with greater summary edge weight.

2. Implementation

We implement both epipolar-corridors-to-graph and the point correspondence algorithms in C++11. We use Microsoft Visual Studio 2015 C++ Compiler to build our code.

In the current version of our software implementation of algorithms, we use Microsoft C++ Concurrency Runtime\Parallel Patterns Library for the purpose of parallelization. Our choice is based on the requirements for photogrammetric software being developed: it must run under MS Windows on a mid-high level laptop. Current software only uses CPU cores for performing calculations. We consider tailoring our algorithms to GPU architecture as one of the directions for the future work.

Other modules of our photogrammetric system that are not described in this paper are implemented with either GNU Octave or C++.

2.1. Implementation of parallel algorithm for graph building from epipolar geometry

Initially, we had been implementing epipolar-corridors-to-graph algorithm with the use of OpenCV 3.1 Library [30] for matrix operations (because OpenCV is the inner standard of our photogrammetric system). However, profiling showed that OpenCV matrix performance is low, so we chose to use Eigen 3.2.8 library [31] for matrix operations which gives us extra speedup.

Our parallel algorithm is given in pseudo-code in Fig. 3.

We modify the initial algorithm described in Section 1.2 by splitting it into several steps. First, we build a batch of tasks (a pair of images + corresponding fundamental matrix) in a parallel. Then we process this batch (as well in a parallel) by calculating (7) for each pair of image points from current pair of images and writing down pairs that lie within the corridor. Finally, we build a graph (in a form of edge list, as required by design rules) based on the data from the second step.

These modifications make calculation of fundamental matrices to be independent from the processing of image point pairs. Besides that, it reduces cyclomatic complexity of algorithm from 4 nested cycles to 3 and gives us more potential for parallelization.

```
pairs<i,j> := all_possible_pairs(1…N_images)
parallel_for_each(pairs)
  F := equations (2) - (6)
  tasks ← <i, j, F>
end_parallel_for_each
parallel_for_each(tasks)
  for_each( p_l ∈ image_points(i) )
    e_r = F · p_l
    for_each( p_r ∈ image_points(j) )
      e_l = F · p_r
      if ( equation (7) ≤ ec_halfwidth )
        graph ← edge(p_l, p_r)
      endif
    end_for_each
  end_for_each
end_parallel_for_each
```

Fig. 3. Parallel algorithm for graph building from epipolar geometry

2.2. Details on implementation of parallel algorithm for point correspondence problem

We implement our newly proposed graph-based point correspondence algorithm as a part of clique finding module, which also implements other clique finding algorithms described in [29].

The pseudo-code of the new algorithm is given above in Fig. 2.

In order to find the maximum clique, we use modified Konc and Janěžič [32] Maximum Clique Algorithm, also known as MCQDyn. We use its reference implementation [33] as the basis of function `find_maximum_clique_of_min_weight` in Fig. 2. Our modifications of this algorithm are described in the next Section 2.3.

Our clique finding module also contains implementation of other sequential point correspondence algorithms described in [29]. We use those algorithms as competitors to our new parallel algorithm. Most of those algorithms (i.e. PLG4) also use MCQDyn to find the maximum clique. However, other sequential algorithms, such as CE, find *all* maximal cliques in a graph. In order to do this, we use Tomita's variation [34, 35] of Brohn–Kerbosch algorithm [36] with pivoting. Our code is based on Ozaki's implementation [37] that uses adjacency lists built upon `unordered_set` from Boost [38].

2.3. Implementation of the modified Konc-Janěžič algorithm for the maximum clique with minimal summary edge weight

Most maximum clique algorithms are only capable of finding clique with the maximum number of vertices. In tasks related to epipolar geometry it is also important to consider *the summary edge weight of a vertex set* (which is also called *clique weight*). Edge weight defines how far the vertex is from its corresponding epipolar line, thus clique weight shows how close this vertex set lies to its corresponding epipolar lines. Given two cliques of the same size, we should prefer the one that has lesser summary edge weight.

In the text below we name key differences of our implementation of Konc-Janěžič maximum clique algorithm (also known as MCQDyn) that find maximum clique with minimal weight from its reference implementation [33].

We introduce additional data structure for keeping edge weights. It is implemented as `unordered_map<pair<int,int>,double>`, which has search time complexity of O(1). For working with clique weight, it is also necessary to implement the corresponding `clique_weight` function.

Immediately after exiting the recursion call, the reference version of the algorithm checks whether the newly found candidate clique Q can unseat current maximum clique QMAX by the following criterion: $|Q| > |QMAX|$. Our modified criteria is given in Fig. 4. It also skips cliques with the size lower than *t (threshold)*, which is used for filtering out artefacts:

```
if |Q| ≥ threshold
  if |Q| > |QMAX| OR
      |Q| = |QMAX| AND clique_weight(|Q|)<clique_weight(|QMAX|)
         QMAX := Q
  endif
endif
```

Fig. 4. Modified QMAX unseat criterion

3. Experimental results

Test machine A is equipped with Intel Core i7-6700K processor running at 4.0 GHz (4 cores, 8 threads) and 48 GBs of RAM. We as well estimate integral algorithm efficiency (section 3.4) on machine B, which is equipped with Intel Core 2 Duo CPU running at 2.66 GHz (2 cores, 2 threads) and 4 GBs of RAM. Except where otherwise noted, we provide average running time for a series of n=10 launches with the same input data as "running time of algorithm". All datasets can be downloaded from `https://github.com/tushev/pplgx-sample-data`.

3.1. Estimation of complexity for superclique approach

In order to estimate time complexity for superclique approach, we use one of our test series with 74 pictures and 122 spatially dense retroreflective targets. We obtain graph G with 1 048 vertices and 19 542 edges with edge density of 3,56%. The corresponding *supergraph* S consists of 16 356 vertices и 77 140 880 edges with edge density of 57,67%. It took 74,4 seconds on test machine A just to build S's adjacency matrix in RAM (with the total program's memory consumption of 6,97 Gb). In comparison, the previous step that found 16 356 maximal cliques in G took only 0,138 seconds. We were unable to find superclique as we had to terminate superclique discovery step after more than 2 hours of computations.

Thus, at the moment, the superclique approach may only be of theoretical interest. It is unsuitable for most practical photogrammetric applications (except for the cases with small number of cameras and targets) due to high time complexity.

3.2. Experimental results for parallel graph building algorithms

Tab. 1 provides average running times for three variants of the epipolar-corridors-to-graph algorithm: classic sequential implementation that uses OpenCV library for matrix operations, parallel implementation with OpenCV and parallel implementation with Eigen library for matrix operations.

Table 1

Running time of graph building algorithms from epipolar geometry, s.

#	# of pictures	Total images of points	Avg. images of points per picture	parallel-cv	parallel-eigen	classic
1	23	2186	95	0,8658	0,3831	5,0344
2	30	1288	42	0,5966	0,3606	2,0730
3	89	896	10	0,4149	0,2776	0,3608

We use three different photographic sessions as input data. The data given in Tab. 1 suggest that our parallel OpenCV implementation runs faster than sequential implementation on dense sessions (like datasets 1&2), while the overheads slow it down on the small-sized sparse dataset 3. Nevertheless, the Eigen-based parallel implementation always runs faster because of better performance of matrix operations. All the tests are performed on machine A.

3.3. Experimental results for point correspondence algorithms for synthetic data

We develop a simulation model of our photogrammetric system to estimate efficiency of point correspondence algorithms. This simulator includes graph generator, which allows to form synthetical graphs with the specified number of "pictures" taken, number of targets and number of artefacts. By varying these parameters, we obtain graphs with different number of vertices, edges and different edge density.

Tab. 2 contains average runtime of point correspondence algorithms on different synthetic graphs on machine A.

Table 2

Running time of point correspondence algorithms, s.

Graph #	Vertices	Edges	Graph edge density	Images	PPLGx, s	PLG4, s	CE, s
1	2 188	71 024	2,97%	23	**0,207**	0,266	0,444
2	2 175	19 722	0,83%	23	**0,024**	0,842	0,162
3	2 188	69 284	2,90%	23	**0,191**	0,27	0,439
4	2 978	42 786	0,97%	20	**0,107**	0,212	0,338
5	11 112	375 414	0,61%	53	**2,936**	7,734	4,457
6	20 590	981 122	0,46%	71	**14,488**	15,507	16,461
7	32 739	1 609 460	0,30%	83	**30,332**	219,215	40,253

We denote the new parallel point correspondence algorithm as PPLGx, which is described together with sequential PLG4 algorithm in Section 1.5. A sequential CE point matching algorithm is described in [29].

The data in Tab. 2 suggest that our new algorithm is faster in all cases. However, this data does not reflect overall matching efficiency of proposed approaches and should only be used to compare the execution time between algorithms. To estimate overall integral matching efficiency we should test our algorithms as a part of the photogrammetric system.

3.4. Estimation of integral algorithm efficiency

As shown in Section 1.4, our photogrammetric system uses iterative scheme for finding point correspondence that refines data accuracy on each iteration. We have chosen the following variables as the key performance indicators: the total running time, the number of 3D points identified, the mean reprojection error, the total number of point images identified and the number of iterations, denoted as N.

We have picked two test datasets that represent two kinds of situations we may encounter while performing photogrammetric reconstruction. Tab. 3 contains experimental results for a typical photogrammetric session, which is representative for most measurement procedures in our practice.

Table 3

Integral efficiency of the photogrammetric system ("typical" session)

Algorithm	Avg. runtime (Machine A), s	Avg. runtime (Machine B), s	# 3D points	# 2D points	N	Mean reproj. error, px
PPLGx	41,08	135,55	161	1701	17	0,396974
PLG4	45,63	139,92	161	1701	17	0,396974
CE	45,18	141,51	161	1701	17	0,396974

Tab. 4 contains experimental results for our most computationally hard case as of the current date with high spatial density of retroreflective targets. Such situations are not common, but they contain a very large number of candidate points and therefore are the most challenging ones. They also may lead to varying results due to different principles lying beneath matching algorithms.

Table 4

Integral efficiency of the photogrammetric system ("extreme" session)

Algorithm	Avg. runtime (Machine A), s	Avg. runtime (Machine B), s	# 3D points	# 2D points	N	Mean reproj. error, px
PPLGx	187,14	502,59	253	2371	29	0,307259
PLG4	240,39	557,44	253	2379	45	0,305532
CE	269,73	597,10	251	2362	36	0,307569

We use parallel epipolar-geometry-to-graph algorithm with PPLGx and its sequential version with PLG4 and CE.

As we see from Tables 3 & 4, our new parallel algorithm (PPLGx) is the fastest in both cases. Also, in the "extreme" case, it converges the system much earlier than other point correspondence algorithms, and finds the same number of 253 3D points as PLG4. However, in this case, PLG4 identifies 0,33% more image points and gains slightly less reprojection error (around 0,0017 pixels).

Note on overall system accuracy. The overall measurement error of the photogrammetric system mostly depends on final bundle adjustment procedure and the accuracy of measurements of pixel coordinates of the circle targets in the images [26]. The main goal of the point correspondence algorithm is to identify as many targets in the images as possible. Thus, if any of the point correspondence algorithms manages to find more points than the other, the result measurement error of the photogrammetric system (represented in this

case by mean reprojection error) decreases (as shown in the data from Tab. 4). Otherwise, if all algorithms converge to the same result (as in Tab. 3), the final bundle adjustment procedure gains the same accuracy. In this case, the fastest algorithm becomes the most preferable one.

This allows us to create a kind of "switch" in our software, which detects the complexity of the given session and asks the user whether to perform fast but slightly less accurate calculations, or precise but 20-25% slower calculations. If the given session is identified as "typical", the system always chooses the fast parallel point correspondence algorithm.

Conclusion and future work

In this paper we describe how epipolar geometry may be used to find point correspondences when other methods such as feature detectors are not applicable. We introduce a multipartite graph as mathematical representation of the system and describe how to construct it from epipolar geometry. We also propose parallel implementation of graph building algorithm and estimate its efficiency over sequential implementation.

We propose a new parallel graph-based algorithm for finding point correspondence, which is based on the idea of local graphs of small size. Our tests prove that the new algorithm is faster than other sequential point matching algorithms, both on synthetic data and as a part of the photogrammetric system. Also, it is accurate and it produces the same results for all sparse cases as the other algorithms. However, for spatially dense cases, it may lead to slightly different results with negligible reprojection differences. We believe that fine-tuning of the other parts of the photogrammetric system may mitigate these differences and regard this as one of the directions for the future work.

We consider tailoring our algorithms to GPU architecture as the future work as well.

The work was supported by Act 211 Government of the Russian Federation, contract № 02.A03.21.0011.

References

1. Hartley R., Zisserman A. Multiple View Geometry in Computer Vision. 2nd ed. Cambridge University Press, 2004.
2. Baggio D. et al. Mastering OpenCV with Practical Computer Vision Projects. Packt Publishing, 2012.
3. Bay H., Tuytelaars T., van Gool L. SURF: Speeded Up Robust Features. *Computer Vision and Image Understanding*. 2008. vol. 110, no. 3, pp. 346–359. DOI: 10.1016/j.cviu.2007.09.014
4. Lowe D.G. Object Recognition from Local Scale-Invariant Features. *Proceedings of the Seventh IEEE International Conference on Computer Vision*. 1999. vol. 2, pp. 1150–1157. DOI: 10.1109/ICCV.1999.790410
5. Fraser C.S. Innovations in Automation for Vision Metrology Systems. *Photogrammetric Record*. 1997. vol. 15, no. 90, pp. 901–911.
6. Sukhovilov B. M., Grigorova E. A., Development of a Photogrammetric System for Measuring the Spatial Coordinates of Structural Elements of the Frame of a Low-Floor Tram. *Nauka JuUrGU. Materialy 67-j Nauchnoj Konferencii. Sekcii Jekonomiki, Upravlenija i Prava.* [Science of SUSU. Materials of the 67th Scientific Conference. Section

of Economics, Management and Law]. Chelyabinsk, Publishing of the South Ural State University, 2015, pp. 458–463. (in Russian)

7. Tushev S.A., Sukhovilov B.M.. Some Ways to Improve the Performance of Automatic Calibration of Digital Cameras. *Molodoj Issledovatel: Materialy 2-J Nauchnoj Vystavki-Konferentsii Nauchno-Tekhnicheskikh i Tvorcheckikh Rabot Studentov.* [Young Researcher: Materials of the 2nd Scientific Exhibition-Conference of Scientific, Technical and Creative Works of Students]. Chelyabinsk, Publishing of the South Ural State University, 2015, pp. 434–439. (in Russian)
8. Hartmann W., Havlena M., Schindler K. Recent Developments in Large-Scale Tie-Point Matching. *ISPRS Journal of Photogrammetry and Remote Sensing*. 2016. vol. 115, pp. 47–62. DOI: 10.1016/j.isprsjprs.2015.09.005
9. Remondino F., Spera M. G., Nocerino E., Menna F., Nex F. State of the Art in High Density Image Matching. *The Photogrammetric Record*. 2014. vol. 29(146), pp. 144–166. DOI: 10.1111/phor.12063
10. Leutenegger S., Chli M., Siegwart R. BRISK: Binary Robust invariant scalable keypoints. *Proceedings of the 2011 International Conference on Computer Vision (ICCV '11)*. IEEE Computer Society, Washington, DC, USA, 2011. pp. 2548–2555. DOI: 10.1109/ICCV.2011.6126542
11. Ortiz R. FREAK: Fast Retina Keypoint. *Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (CVPR '12)*. IEEE Computer Society, Washington, DC, USA, 2012. pp. 510–517. DOI: 10.1109/CVPR.2012.6247715
12. Fraser C. S., Cronk S. A Hybrid Measurement Approach for Close-Range Photogrammetry. *ISPRS Journal of Photogrammetry and Remote Sensing*. 2009, vol. 64(3), pp. 328–333. DOI: 10.1016/j.isprsjprs.2008.09.009
13. Leung C., Lovell B. C. 3D Reconstruction through Segmentation of Multi-View Image Sequences. *Proceedings of the 2003 APRS Workshop on Digital Image Computing*. 2003. pp. 87–92. Available at: http://espace.library.uq.edu.au/view/UQ:10960 (accessed: 29.11.2016)
14. Qiqiang F., Guangyun L. Matching of Artificial Target Points Based on Space Intersection. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences.* Beijing 2008, vol. *XXXVII,* part *B5*, pp. 111–114.
15. Maas, H.-G. Complexity analysis for the establishment of image correspondences of dense spatial target fields. *International Archives of Photogrammetry and Remote Sensing*. 1992. vol. 29, pp. 102–107.
16. Dold J., Maas, H.-G. An Application of Epipolar Line Intersection in a Hybrid Close-Range Photogrammetric System. *International Archives of Photogrammetry and Remote Sensing*. 1994. vol. 30(5), pp. 65–70.
17. Zhang Z., Deriche R., Faugeras O., Luong Q. T. A Robust Technique for Matching Two Uncalibrated Images through the Recovery of the Unknown Epipolar Geometry. *Artificial Intelligence*. 1995. vol. 78(1–2), pp. 87–119. DOI: 10.1016/0004-3702(95)00022-4
18. Arnfred J. T., Winkler S. A General Framework for Image Feature Matching Without Geometric Constraints. *Pattern Recognition Letters.* 2016. vol. 73, pp. 26–32. DOI: 10.1016/j.patrec.2015.12.017

19. Takimoto R.Y., De Castro Martins T., Takase F.K., De Sales Guerra Tsuzuki M. Epipolar Geometry Estimation, Metric Reconstruction and Error Analysis from Two Images. *IFAC Proceedings*. 2012. vol. 2012;14, pp. 1739–1744. DOI: 10.3182/20120523-3-RO-2023.00098
20. Fraser C. Advances in Close-Range Photogrammetry. *Photogrammetric Week 2015*. 2015. pp. 257–268.
21. V-STARS – Geodetic Systems, Inc. Available at: https://www.geodetic.com/v-stars/ (accessed: 29.01.2017).
22. Agisoft PhotoScan. Available at: http://www.agisoft.com/ (accessed: 29.01.2017).
23. Maas H.-G. Automatic DEM Generation by Multi-Image Feature Based Matching. *International Archives of Photogrammetry and Remote Sensing*. 1996. vol. 31, pp. 484–489.
24. Bhowmick B., Patra S., Chatterjee A., Madhav Govindu V., Banerjee S. Divide and Conquer: A Hierarchical Approach to Large-Scale Structure-from-Motion. *Computer Vision and Image Understanding*. 2017. vol. 157, pp. 190–205. DOI: 10.1016/j.cviu.2017.02.006
25. Forsyth D., Ponce J. Computer Vision: A Modern Approach. 2nd ed. Pearson, 2011.
26. Sukhovilov B.M., Sartasov E.M., Grigorova E. A. Improving the Accuracy of Determining the Position of the Code Marks in the Problems of Constructing Three-Dimensional Models of Object. *Procedings of the 2nd International Conference on Industrial Engineering, Applications and Manufacturing (ICIEAM)*, Chelyabinsk, Russia, 2016. IEEE Xplore Digital Library. pp. 1–4. DOI: 10.1109/ICIEAM.2016.7911682
27. Bomze I., Budinich M., Pardalos P., Pelillo M. The Maximum Clique Problem. *Handbook of Combinatorial Optimization*, 1999. pp. 1–74.
28. Triggs B., McLauchlan P., Hartley R., Fitzgibbon A.. Bundle Adjustment — A Modern Synthesis. *Vision Algorithms: Theory & Practice.* 2000. Vol 1883. pp. 298–372. DOI: 10.1007/3-540-44480-7_21
29. Tushev S.A., Sukhovilov B.M.. Effective Graph-Based Point Matching Algorithms. *Procedings of the 2nd International Conference on Industrial Engineering, Applications and Manufacturing (ICIEAM)*, Chelyabinsk, Russia, 2016. IEEE Xplore Digital Library. pp. 1–5. DOI: 10.1109/ICIEAM.2016.7911628
30. OpenCV (Open Source Computer Vision Library): Available at: http://opencv.org/ (accessed: 29.11.2016).
31. Eigen: C++ Template Library For Linear Algebra: Available at: http://eigen.tuxfamily.org/ (accessed: 29.11.2016).
32. Konc J., Janežič D. An Improved Branch and Bound Algorithm for the Maximum Clique Problem. *MATCH Communications in Mathematical and in Computer Chemistry*. 2007. vol. 58, pp. 569–590
33. Konc J., Janezic D. New C++ Source Code of the MaxCliqueDyn Algorithm: Available at: http://insilab.org/maxclique/ (accessed: 29.11.2016).
34. Tomita E., Tanaka A., Takahashi H. The Worst-Case Time Complexity for Generating All Maximal Cliques and Computational Experiments. *Theoretical Computer Science*. 2006. vol. 363, no. 1, pp. 28–42. DOI: 10.1016/j.tcs.2006.06.015
35. Conte A. Review of the Bron-Kerbosch Algorithm and Variations. Available at: http://www.dcs.gla.ac.uk/~pat/jchoco/clique/enumeration/report.pdf (accessed: 29.11.2016)

36. Bron C., Kerbosch J. Algorithm 457: Finding All Cliques of an Undirected Graph. *Communications of the ACM*, 1973. vol. 16, no. 9, pp. 575–577. DOI: 10.1145/362342.362367
37. Ozaki K. smly:find_cliques – a Pivoting Version of Bron-Kerbosch Algorithm. Available at: https://gist.github.com/smly/1516622 (accessed: 29.11.2016)
38. Boost C++ libraries.: Available at: http://www.boost.org/ (accessed: 29.11.2016).

УДК 004.92, 004.021 **DOI: 10.14529/cmse170204**

ПАРАЛЛЕЛЬНЫЕ АЛГОРИТМЫ ДЛЯ ЭФФЕКТИВНОГО ПОИСКА СООТВЕТСТВУЮЩИХ ТОЧЕК В ЗАДАЧАХ КОМПЬЮТЕРНОГО ЗРЕНИЯ

© 2017 г. С.А. Тушев, Б.М. Суховилов
Южно-Уральский государственный университет
(454080 Челябинск, пр. им. В.И. Ленина, д. 76),
E-mail: science@tushev.org, sukhovilovbm@susu.ru
Поступила в редакцию: 01.05.2017

В настоящей статье предложены параллельные алгоритмы для поиска соответствующих точек в задачах компьютерного зрения. Разрабатываемая коллективом авторов фотограмметрическая система основана на использовании искусственных световозвращающих мишеней, идентичных по фотометрическим параметрам. В связи с этим традиционные методы поиска соответствий на основе вычисления дескрипторов (SIFT, SURF, и др.) неприменимы; фотограмметрическая система использует методы, основанные на эпиполярной геометрии. В настоящей статье предложены эффективные алгоритмы поиска соответствий между точками по всей совокупности снимков (в отличие от классических методов, использующих 2-4 снимка), основанные на графах. Приведено точное двухшаговое решение задачи через суперклику графа потенциальных соответствий; показана невозможность практического нахождения суперклики в реальных задачах в связи с вычислительной сложностью. Предложена эффективная параллельная реализация алгоритма формирования графа на основе эпиполярных ограничений, а также быстродействующий параллельный эвристический алгоритм поиска клик в данном графе. Применение итерационной схемы с обратным проецированием точек, отсевом выбросов и уравниванием координат точек и положений камер через метод связок позволяет в итоге получать точное решение задачи с использованием эвристического алгоритма поиска клик на каждой итерации. Предложенная архитектура системы дает значительный выигрыш во времени. Разработаны программные реализации описанных алгоритмов. Выполнена сравнительная оценка эффективности и производительности предложенных алгоритмов применительно к разрабатываемой фотограмметрической системе, экспериментально подтверждена эффективность предлагаемых решений.

Ключевые слова: компьютерное зрение, фотограмметрия, поиск соответствующих точек, параллельные алгоритмы, нахождение максимальной клики, эпиполярная геометрия

ОБРАЗЕЦ ЦИТИРОВАНИЯ

Tushev S.A., Sukhovilov B.M. Parallel Algorithms for Effective Correspondence Problem Solution in Computer Vision // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2017. Т. 6, № 2. С. 49–68. DOI: 10.14529/cmse170204.

Литература

1. Hartley R., Zisserman A. Multiple View Geometry in Computer Vision. 2nd ed. Cambridge University Press, 2004.
2. Baggio D. et al. Mastering OpenCV with Practical Computer Vision Projects. Packt Publishing, 2012.

3. Bay H., Tuytelaars T., van Gool L. SURF: Speeded Up Robust Features // Computer Vision and Image Understanding. 2008. Vol. 110, No. 3, P. 346–359. DOI: 10.1016/j.cviu.2007.09.014
4. Lowe D.G. Object recognition from local scale-invariant features // Proceedings of the Seventh IEEE International Conference on Computer Vision. 1999. Vol. 2, P. 1150–1157. DOI: 10.1109/ICCV.1999.790410
5. Fraser C.S. Innovations in automation for vision metrology systems // Photogrammetric Record. 1997. Vol. 15, No. 90, P. 901–911.
6. Суховилов Б. М., Григорова Е. А. Разработка фотограмметрической системы измерения пространственных координат элементов конструкций каркаса низкопольного трамвая // Наука ЮУрГУ [Электронный ресурс]. Материалы 67-й научной конференции. Секции экономики, управления и права. Челябинск: Издательский центр ЮУрГУ, 2015. С. 458–463.
7. Тушев С.А., Суховилов Б.М. Некоторые способы повышения производительности автоматической калибровки цифровых камер// Молодой исследователь: материалы 2-й научной выставки-конференции научно-технических и творческих работ студентов. Челябинск: Издательский центр ЮУрГУ, 2015. С. 434–439.
8. Hartmann W., Havlena M., Schindler K. Recent developments in large-scale tie-point matching // ISPRS Journal of Photogrammetry and Remote Sensing. International Society for Photogrammetry and Remote Sensing, Inc. (ISPRS), 2016. Vol. 115. P. 47–62.. DOI: 10.1016/j.isprsjprs.2015.09.005
9. Remondino F. et al. State of the art in high density image matching // The Photogrammetric Record. 2014. Vol. 29, No 146. P. 144–166. DOI: 10.1111/phor.12063
10. Leutenegger S., Chli M., Siegwart R.Y. BRISK : Binary Robust Invariant Scalable Keypoints. DOI: 10.1109/ICCV.2011.6126542
11. Ortiz R., Alahi A., Vandergheynst P. FREAK: Fast retina keypoint // Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition. 2012. P. 510–517. DOI: 10.1109/CVPR.2012.6247715
12. Fraser C.S., Cronk S. A hybrid measurement approach for close-range photogrammetry // ISPRS Journal of Photogrammetry and Remote Sensing. Elsevier B.V., 2009. Vol. 64, No 3. P. 328–333.
13. Leung C., Lovell B.C. 3D Reconstruction through Segmentation of Multi-View Image Sequences // Proceedings of the 2003 APRS Workshop on Digital Image Computing. 2003. P. 87–92. URL: http://espace.library.uq.edu.au/view/UQ:10960 (дата обращения: 29.11.2016)
14. Qiqiang F., Guangyun L. Matching of artificial target points based on space intersection // The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences. 2008. Vol. XXXVII. P. 111–114.
15. Maas H.-G. Complexity analysis for the establishment of image correspondences of dense spatial target fields // International Archives of Photogrammetry and Remote Sensing. 1992. Vol. 29. P. 102–107.
16. Dold J., Maas, H.-G. An Application of Epipolar Line Intersection in a Hybrid Close-Range Photogrammetric System // International Archives of Photogrammetry and Remote Sensing. 1994. Vol. 30(5), P. 65–70.

17. Zhang Z. et al. A robust technique for matching two uncalibrated images through the recovery of the unknown epipolar geometry // Artificial Intelligence. 1995. Vol. 78, No 1–2. P. 87–119. DOI: 10.1016/0004-3702(95)00022-4
18. Arnfred J.T., Winkler S. A general framework for image feature matching without geometric constraints // Pattern Recognition Letters. Elsevier B.V., 2016. Vol. 73. P. 26–32. DOI: 10.1016/j.patrec.2015.12.017
19. Takimoto R.Y. et al. Epipolar geometry estimation, metric reconstruction and error analysis from two images // IFAC Proceedings Volumes (IFAC-PapersOnline). 2012. Vol. 14, No 1. P. 1739–1744. DOI: 10.3182/20120523-3-RO-2023.00098
20. Fraser C. Advances in Close-Range Photogrammetry // Photogrammetric Week 2015. 2015. P. 257–268.
21. V-STARS – Geodetic Systems, Inc. URL: https://www.geodetic.com/v-stars/ (дата обращения: 29.01.2017).
22. Agisoft PhotoScan. URL: http://www.agisoft.com/ (дата обращения:: 29.01.2017).
23. Maas H.-G. Automatic DEM generation by multi-image feature based matching // International Archives of Photogrammetry and Remote Sensing. 1996. Vol. 31. P. 484–489.
24. Bhowmick B. et al. Divide and conquer: A hierarchical approach to large-scale structure-from-motion // Computer Vision and Image Understanding. Elsevier Inc., 2017. Vol. 157. P. 190–205. DOI: 10.1016/j.cviu.2017.02.006
25. Forsyth D., Ponce J. Computer Vision: A Modern Approach. 2nd ed. Pearson, 2011.
26. Суховилов Б.М., Сартасов Е.М., Григорова Е.А. Повышение точности определения положения кодовых марок в задачах построения трехмерных моделей объектов // Пром-инжиниринг: труды II международной научно-технической конференции. Челябинск: Издательский центр ЮУрГУ, 2016. С. 363–366.
27. Bomze I., Budinich M., Pardalos P., Pelillo M. The Maximum Clique Problem // Handbook of Combinatorial Optimization, 1999. P. 1–74.
28. Triggs B., McLauchlan P., Hartley R., Fitzgibbon A.. Bundle Adjustment — A Modern Synthesis // Vision Algorithms: Theory & Practice. 2000. Vol. 1883. P. 298–372. DOI: 10.1007/3-540-44480-7_21
29. Тушев С.А., Суховилов Б.М. Эффективные алгоритмы поиска соответствий точек на снимках на основе графов // Пром-инжиниринг: труды II международной научно-технической конференции. Челябинск: Издательский центр ЮУрГУ, 2016. С. 464–468.
30. OpenCV (Open Source Computer Vision Library): URL: http://opencv.org/ (дата обращения: 29.11.2016).
31. Eigen: C++ template library for linear algebra: URL: http://eigen.tuxfamily.org/ (дата обращения: 29.11.2016).
32. Konc J., Janežič D. An improved branch and bound algorithm for the maximum clique problem // MATCH Communications in Mathematical and in Computer Chemistry. 2007, P. 569–590
33. Konc J., Janezic D. New C++ source code of the MaxCliqueDyn algorithm: URL: http://insilab.org/maxclique/ (дата обращения: 29.11.2016).

34. Tomita E., Tanaka A., Takahashi H. The worst-case time complexity for generating all maximal cliques and computational experiments // Theoretical Computer Science. 2006, Vol. 363, No. 1, P. 28–42. DOI: 10.1016/j.tcs.2006.06.015
35. Conte, A. Review of the Bron-Kerbosch algorithm and variations. URL: http://www.dcs.gla.ac.uk/pat/jchoco/clique/enumeration/report.pdf (дата обращения: 29.11.2016)
36. C. Bron, J. Kerbosch. Algorithm 457: finding all cliques of an undirected graph // Communications of the ACM. Sep 1973, Vol. 16, No. 9, P. 575–577. DOI: 10.1145/362342.362367
37. Ozaki K. smly:find_cliques – a pivoting version of Bron-Kerbosch algorithm. URL: https://gist.github.com/smly/1516622 (дата обращения: 29.11.2016)
38. Boost C++ libraries.: URL: http://www.boost.org/ (дата обращения: 29.11.2016).

Тушев Семен Александрович, аспирант, высшая школа экономики и управления, «Южно-Уральский государственный университет (национальный исследовательский университет)» (Челябинск, Российская Федерация)

Суховилов Борис Максимович, д.т.н, с.н.с., высшая школа экономики и управления, «Южно-Уральский государственный университет (национальный исследовательский университет)» (Челябинск, Российская Федерация)

УДК 004.853 DOI: 10.14529/cmse170205

КОМПЛЕКС ПРОГРАММ АВТОМАТИЧЕСКОГО ПОСТРОЕНИЯ СЕМАНТИЧЕСКОЙ СЕТИ СЛОВ

© 2017 г. Д.А. Усталов[1,2], А.В. Созыкин[1,2]

[1] *Институт математики и механики им. Н.Н.Красовского Уральского отделения Российской академии наук (620990 Екатеринбург, ул. Софьи Ковалевской, д. 16),*

[2] *Уральский федеральный университет имени первого Президента России Б.Н. Ельцина (620002 Екатеринбург, ул. Мира, д. 19)*

E-mail: dau@imm.uran.ru

Поступила в редакцию: 01.05.2017

Семантическая сеть слов — это ориентированный граф, вершины которого — лексические значения слов, а ребра — отношения между ними. В статье представлен комплекс программ SWN, предназначенный для построения семантической сети слов в автоматическом режиме путем структурирования неразмеченных словарей синонимов и словарей родо-видовых отношений с использованием векторных представлений слов, полученных на основе обработки корпуса неструктурированных текстов на естественном языке. Комплекс программ включает в себя реализацию методов обнаружения групп синонимов и построения отношений между отдельными значениями слов, основанных на обучении без учителя, а также модуля расширения отношений, основанного на обучении с учителем. Приведена модель предметной области с использованием формализма VOWL. Архитектура комплекса программ представлена в формализме UML и включает модуль обнаружения понятий, модуль построения семантических отношений между значениями слов, модуль расширения семантических отношений, модуль преобразования результатов работы в форматы Семантической паутины, и модуль построения оценочного набора данных при помощи краудсорсинга. Представленный комплекс программ является программным обеспечением с открытым исходным кодом и доступнен для интеграции в различные системы интеллектуального анализа данных.

Ключевые слова: семантическая сеть, лексическая семантика, программная инженерия, свободное программное обеспечение, Семантическая паутина, VOWL, UML.

ОБРАЗЕЦ ЦИТИРОВАНИЯ

Усталов Д.А., Созыкин А.В. Комплекс программ автоматического построения семантической сети слов // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2017. Т. 6, № 2. С. 69–83. DOI: 10.14529/cmse170205.

Введение

Семантическая сеть — это ориентированный граф, вершины которого — понятия, а ребра — отношения между ними [1]. Такой способ представления знаний широко применяется в области искусственного интеллекта и обработки естественного языка, что подтверждается использованием таких семантических ресурсов, как WordNet для английского языка и РуТез для русского языка [2]. К сожалению, язык изменяется быстрее, чем обновляются подобные ресурсы: возникают новые слова или новые значения существующих слов. Это приводит к тому, что все больше внимания уделяется созданию методов автоматического построения семантических сетей на основе обработки и выравнивания доступных структурированных и неструктурированных языковых ресурсов. Задача автоматического построения онтологии (англ. *ontology learning*) предполагает как интеграцию готовых словарей, так и извлечение информации из неразмеченных

корпусов текстов и использование машинного перевода для обеспечения полноты данных [3]. Известным примером высококачественной семантической сети, построенной автоматическим образом и доступной более чем для двухсот различных естественных языков, является BabelNet [4].

В последние годы особенно заметна тенденция по разработке методов обучения без учителя для автоматического формирования понятий и связей между ними, см. обзоры в [1, 3–5]. Это вызвано двумя причинами: 1) популярностью и полнотой неструктурированных ресурсов, таких как Википедия и Викисловарь, построенных при помощи краудсорсинга, 2) существенным снижением стоимости высокопроизводительных вычислительных ресурсов, что позволяет разрабатывать методы машинного обучения с использованием больших объемов данных. В данной работе представлен комплекс программ, формирующий семантическую сеть путем связывания отдельных лексических значений слов в виде специальной структуры данных — семантической сети слов [6–8]. В отличие от традиционных подходов к связыванию отдельных понятий [2], данный подход не требует высококачественного словаря понятий для их связывания друг с другом [4].

Статья организована следующим образом. В разделе 1 содержится описание предметной области и использованного подхода к построению семантической сети слов — разновидности семантической сети. В разделе 2 представлена архитектура комплекса программ SWN (сокр. англ. *semantic word network* — семантическая сеть слов) и функциональные особенности входящих в него программ. В заключении обсуждаются полученные результаты и рассматриваются направления дальнейших исследований.

1. Автоматическое построение семантической сети слов

Под семантической сетью слов мы будем понимать такую семантическую сеть, вершинами которой являются не понятия, т.е. множества синонимов [2], также известные как *синсеты*, а отдельные лексические значения слов, составляющие эти понятия.

Определение 1. Семантическая сеть слов — это семантическая сеть, вершины которой — лексические значения слов, а ребра — отношения между ними.

При обозначении слова и идентификатора его отдельного значения используется нотация, подобная принятой в BabelNet [4], но без указания части речи в нижнем регистре: запись $лук^1$ и $лук^2$ обозначает два различных значения одной и той же лексемы «лук». Например, слово «лук» имеет не меньше двух значений: $лук^1$ — метательное оружие, $лук^2$ — растение, и т. д. В целях избежания неоднозначности, в тексте статьи сноски не используются. Основное внимание в данной работе посвящено построению семантической сети на основе лексических значений слов, причем рассматривается единственный класс семантических онтошений — родо-видовые отношения, т. е. отношение между менее общим словом (гипонимом) и более общим словом (гиперонимом) [2]. Например, упорядоченная пара слов (*котенок*, *млекопитающее*) является корректной родо-видовой парой слов.

Общая схема метода построения семантической слов представлена на рис. 1. Данные методы ориентированы на использование таких ресурсов, как словарь синонимов и неразмеченная коллекция документов (корпус текстов). Источниками данных для такого метода являются материалы Викисловаря как словаря отношений между словами и неструктурированные тексты электронной библиотеки lib.rus.ec как корпуса текстов, содержащего 13 млрд словоупотреблений. На первом этапе производится выделение

значений слов и объединение их в группы близких слов при помощи кластеризации графа синонимов [6], после чего осуществляется связывание — формирование семантических отношений между лексическими значениями слов [7]. Кроме того, на этапе связывания производится расширение материалов существующих словарей [8], но эта операция не является обязательной.

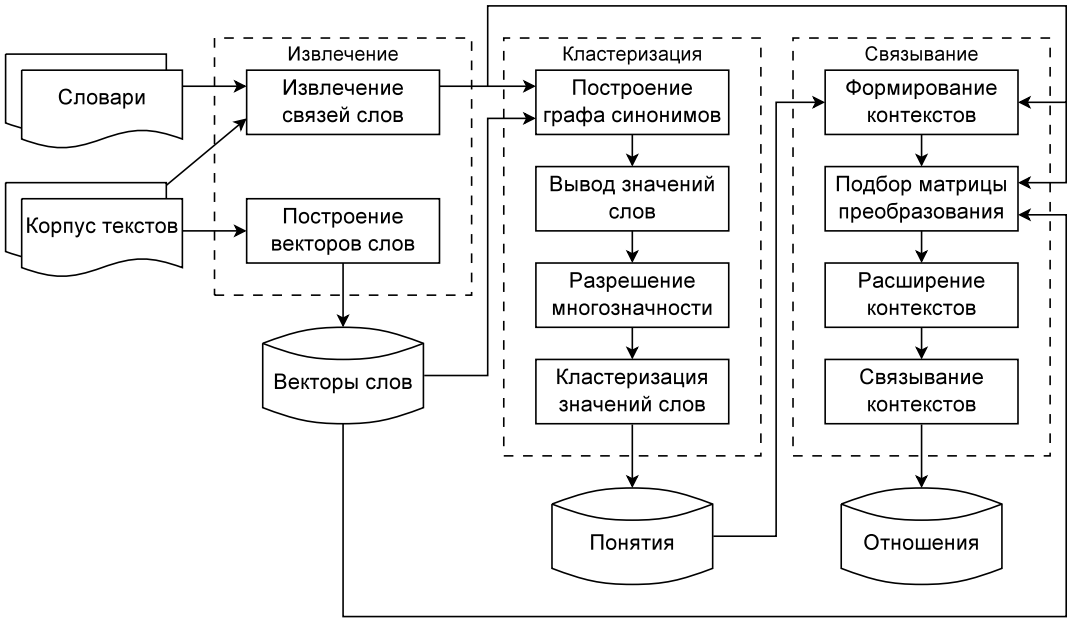


Рис. 1. Общая схема метода построения семантической сети слов

Для обеспечения возможности широкого применения семантической сети слов, используется запись в формализме RDF (англ. *Resource Description Framework*). RDF использует представление данных в виде троек «субъект–предикат–объект» [9]. На рис. 2 изображена диаграмма классов полученной семантической сети слов с точки зрения формализма VOWL [10], использующая модели SKOS [11] и Lemon [12] для записи лексико-семантической информации.

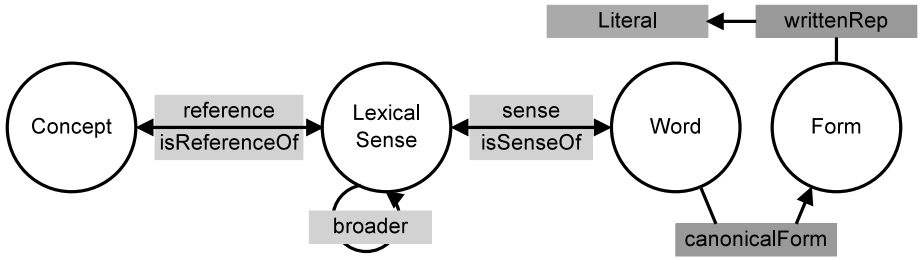


Рис. 2. VOWL-диаграмма информационной модели предметной области

2. Комплекс программ построения семантической сети слов

UML-диаграмма пакетов комплекса программ SWN представлена на рис. 3. Модуль обнаружения понятий Watset (от англ. *what* — «что?» и *set* — множество) реализует метод обнаружения понятий [6]. Модуль связывания Watlink (от англ. *what* — «что?» и *link* — связь) реализует метод связывания значений слов [7]. Модуль подбора матрицы линейного преобразования Hyperstar (от англ. *hyper* — «гипер» и *star* — «любой») реализует метод подбора матрицы линейного преобразования, используемый для расширения словарей [8].

Модуль экспорта данных (SWNRDF) реализует преобразование семантической сети слов в стандартный формат RDF [13] при помощи программы *Converter*.

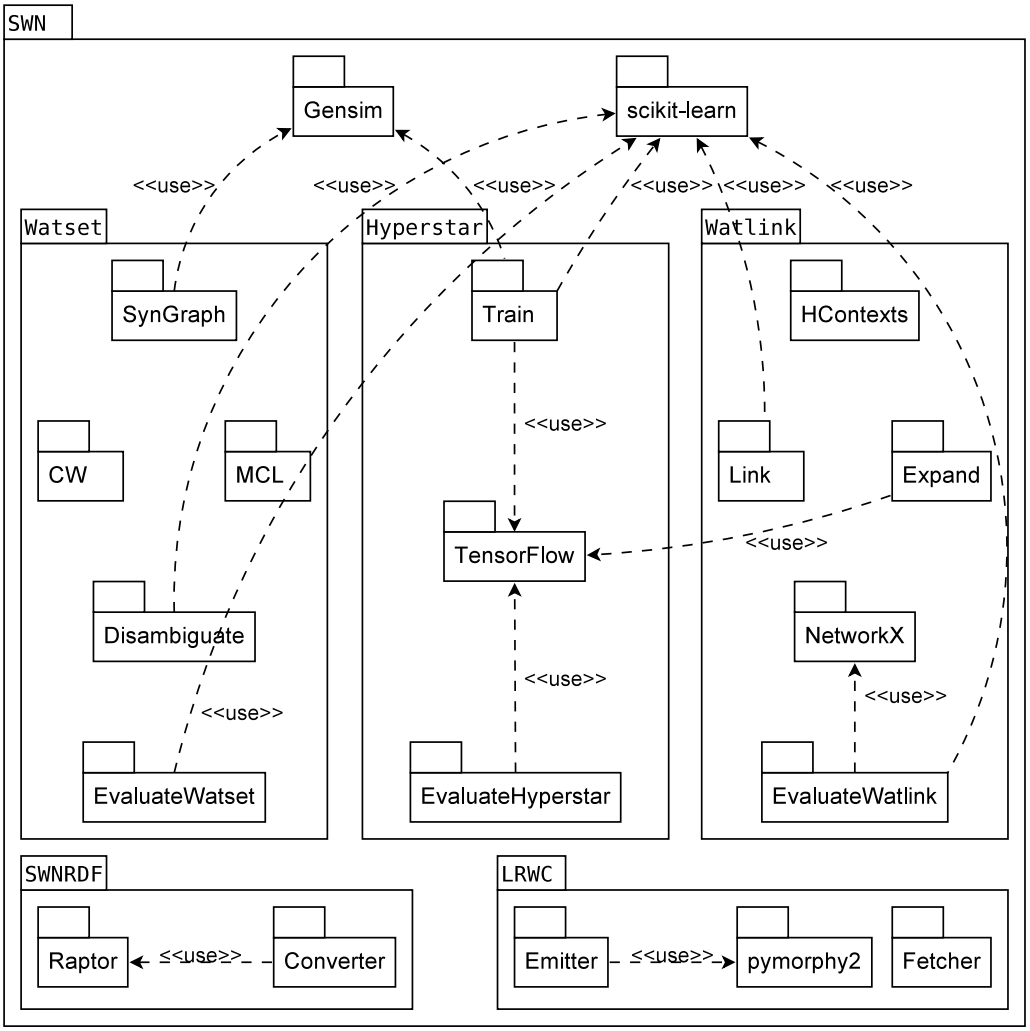


Рис. 3. UML-диаграмма пакетов комплекса программ

При реализации комплекса программ SWN использованы языки программирования Python, AWK, Java и Bash. Применяются внешние библиотеки, в том числе библиотека алгоритмов машинного обучения, подготовки и обработки данных scikit-learn [14], реализация алгоритма кластеризации Chinese Whispers [15] (CW), реализация марковского алгоритма кластеризации [16] (MCL), библиотека тематического моделирования и работы с векторами слов Gensim [17], библиотека методов оптимизации TensorFlow [18], библиотека работы с графами NetworkX [19], а также средства обработки RDF-троек Raptor [20] и морфологический анализатор pymorphy2 [21].

2.1. Модуль Watset

Модуль Watset реализует одноименный метод обнаружения понятий на основе графа синонимов [6]. На рис. 4 представлена UML-диаграмма активности обнаружения понятий, состоящая из трех шагов [6]: подготовка данных (программа *SynGraph*), обнаружение понятий (программы *CW*, *MCL* и *Disambiguate*), тестирование (программа *EvaluateWatset*).

Сначала производится загрузка исходных словарей и извлечение из них множества пар синонимов. При необходимости, вычисляется значение семантической близости

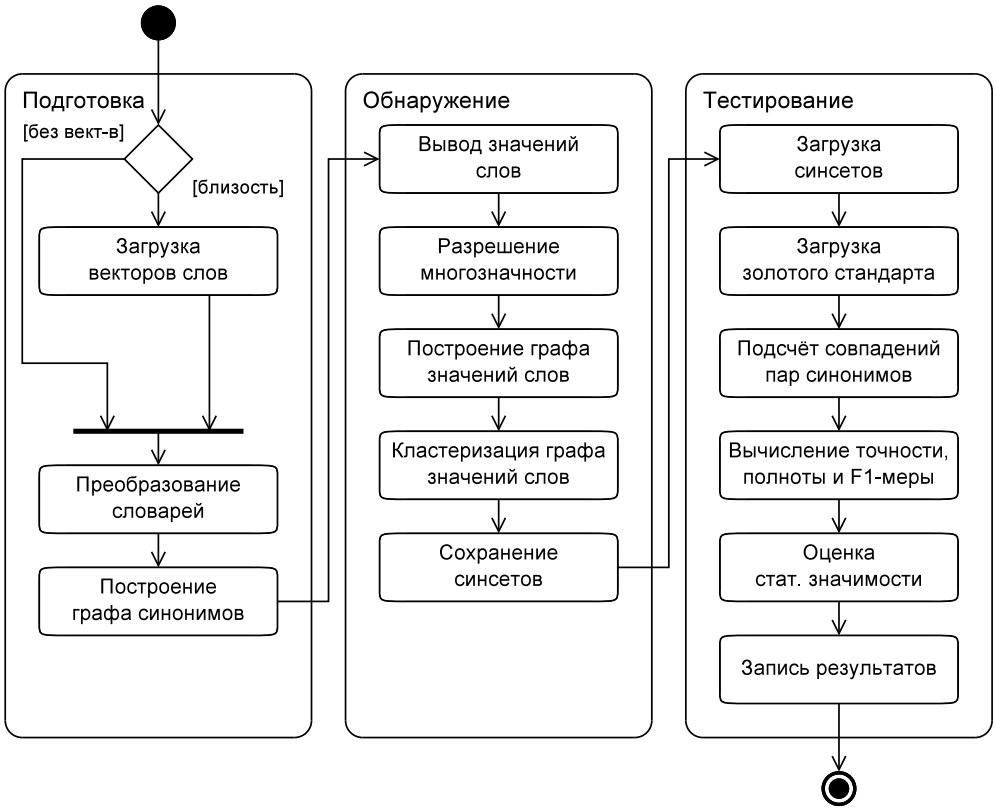


Рис. 4. UML-диаграмма активности обнаружения понятий

между парами синонимов на основе косинусной меры близости между векторами слов. Эти сведения используются при построении графа синонимов. При отсутствии сведений о семантической близости слов предусмотрено два альтернативных варианта: использование единичных весов для каждого ребра графа синонимов или подсчет количества появлений пары синонимов в исходных словарях. На этапе вывода значений слов допускается использование двух различных алгоритмов кластеризации эго-сетей: Chinese Whispers [15] или MCL [16]. На этапе разрешения многозначности производится разрешение многозначности в контекстах, причем в целях повышения производительности используется традиционный прием параллелизма по данным: каждое слово обрабатывается независимо в отдельном процессе. Определение номера значения слова в контексте производится путем максимизации косинусной меры близости [6]:

$$\hat{u} = \underset{u' \in \operatorname{senses}(u)}{\arg\max} \cos(\operatorname{ctx}(s), \operatorname{ctx}(u')), \tag{1}$$

где s — значение некоторого слова с известным номером значения, u — слово с неизвестным номером значения, $\operatorname{senses}(u)$ — множество значений слова u, $\operatorname{ctx}(\cdot)$ — контекст, т. е. множество синонимов слова в указанном значении. В результате разрешения многозначности формируется граф значений слов, кластеризация которого для получения синсетов производится методом Chinese Whispers или MCL. Синсеты получают уникальные номера и записываются в текстовый файл. Это необходимо как для использования данных в других задачах, так и для оценки качества. При оценке качества загружаются полученные синсеты и синсеты золотого стандарта. Затем каждый синсет из n слов преобразуется во множество из $\frac{n(n-1)}{2}$ пар синонимов и производится подсчет совпадений пар синонимов в полученном ресурсе и золотом стандарте. Вычисляются значения

информационно-поисковых критериев точности, полноты и F_1-меры [22] и оценивается статистическая значимость значения каждого критерия [23]. После выполнения всех указанных процедур осуществляется запись результатов оценки в текстовый файл.

2.2. Модуль Hyperstar

Модуль Hyperstar осуществляет подбор матрицы линейного преобразования векторных представлений гипонимов в векторные представления гиперонимов на основе модифицированного подхода, первоначально предложенного в [24]. На рис. 5 представлена UML-диаграмма активности подбора матрицы линейного преобразования, состоящая из трех условных шагов: подготовка данных и обучение модели (программа *Train*), тестирование (программа *EvaluateHyperstar*). Исходными данными для подбора матрицы являются векторы слов и примеры родо-видовых отношений между словами, полученные из словарей. В процессе используются только те пары слов, для которых имеются векторы. Это вызвано тем, что векторы слов строятся на основании большого корпуса текстов с различными подходами к предварительной обработке, например, фильтрации низкочастотных слов. Полученные пары векторов слов разбиваются на три различные выборки в соотношении: 60 % данных составляют обучающую выборку для подбора параметров, 20 % данных составляют валидационной выборки для подбора метапараметров, и оставшиеся 20 % составляют тестовую выборку для оценки качества модели.

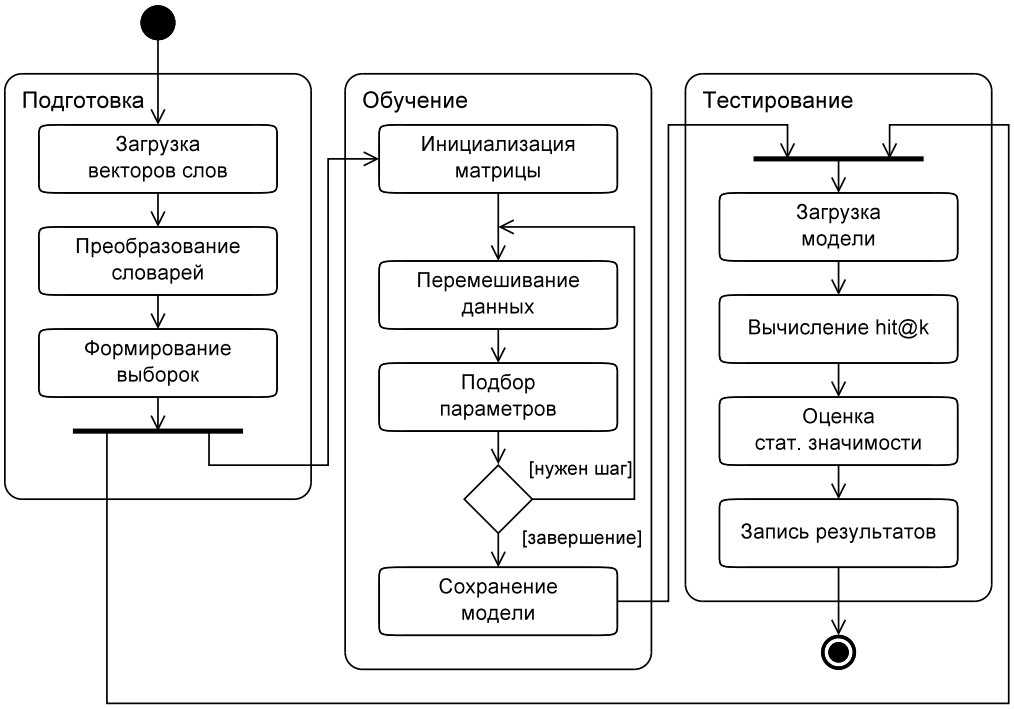


Рис. 5. UML-диаграмма активности подбора матрицы линейного преобразования

В начале процесса обучения все элементы матрицы генерируются как независимые между собой случайные величины, имеющие стандартное нормальное распределение с параметрами $\mu = 0$ и $\sigma = 0{,}1$; допущения о свойствах матрицы не используются [24]. На каждом шаге обучения производится перемешивание данных и выполняется подбор значений элементов матрицы с целью минимизации следующей функции потерь с применением регуляризации на основе выборки отрицательных примеров [8] (при записи

используется нотация вектора-строки, т. е. вектор $\vec{v}$ является матрицей размера $1 \times |\vec{v}|$):

$$\Phi^* = \arg\min_{\Phi} \frac{1}{|\mathcal{P}|} \sum_{(\vec{x},\vec{y})\in\mathcal{P}} \|\vec{x}\Phi - \vec{y}\|^2 + \lambda R,$$
$$R_h = \frac{1}{|\mathcal{P}|} \sum_{(\vec{x},\_)\in\mathcal{P}} (\vec{x}\Phi\Phi \cdot \vec{x})^2, \qquad R_s = \frac{1}{|\mathcal{N}|} \sum_{(\vec{x},\vec{z})\in\mathcal{N}} (\vec{x}\Phi\Phi \cdot \vec{z})^2, \tag{2}$$

где Φ — матрица линейного преобразования, $\vec{x}$ — вектор гипонима, $\vec{y}$ — вектор гиперонима, $\vec{z}$ — вектор синонима $\vec{x}$, $\mathcal{P}$ — обучающая выборка с положительными примерами, $\mathcal{N}$ — обучающая выборка с отрицательными примерами, λ — параметр важности члена регуляризации $R \in \{R_h, R_s\}$, R_h — регуляризатор, накладывающий ограничения на близость преобразования $\vec{x}\Phi\Phi$ к исходному гипониму $\vec{x}$, R_s — регуляризатор, накладывающий ограничения на близость преобразования $\vec{x}\Phi\Phi$ к синониму $\vec{z}$ исходного гипонима $\vec{x}$. Процесс обучения завершается по достижении указанного при запуске количества шагов; двоичное представление полученной матрицы записывается в файл. Оценка качества предполагает загрузку полученных матриц и вычисление значения критерия hit@k по валидационной выборке для подбора параметров или по тестовой выборке для оценки качества работы метода. Оценивается статистическая значимость значения данного критерия [8]. После выполнения всех указанных процедур осуществляется запись результатов оценки в текстовый файл.

2.3. Модуль Watlink

Модуль Watlink реализует одноименный метод построения однозначных семантических отношений [7] и осуществляет построение семантической сети слов на основе родо-видовых пар слов и ранее полученных синсетов. На рис. 6 представлена UML-диаграмма активности построения отношений, состоящая из трех условных шагов: подготовка данных (программа *HContexts*), связывание (программы *Link* и, опционально, *Expand*), тестирование (программа *EvaluateWatlink*).

Исходными данными для построения отношений являются синсеты и сведения о требуемых семантических отношениях. Извлечение семантических отношений производится из словарей, содержащих перечисленные родо-видовые пары в текстовом виде: $\mathrm{hctx}(S) = \{h : w \in \mathrm{words}(S), (w,h) \in \mathcal{R}\}$, где $\mathrm{hctx}(S)$ — иерархический контекст синсета S, $\mathrm{words}(S)$ — слова, входящие в синсет S, $\mathcal{R}$ — множество родо-видовых отношений. На основе этих сведений формируются иерархические контексты каждого синсета. При необходимости загружаются векторы слов, матрица линейного преобразования, и осуществляется расширение иерархических контекстов с использованием ранее полученной матрицы линейного преобразования [7, 8]:

$$\mathrm{hctx}'(S) = \cup_{\substack{w\in\mathrm{words}(S),\\(w,h)\in\mathcal{R}}} \{w\} \times \mathrm{NN}_n(\vec{h}) \cup \mathrm{hctx}(S), \tag{3}$$

где $\mathrm{NN}_n(\vec{h})$ — операция поиска n ближайших соседей векторного представления слова h.

Разрешение многозначности в иерархических контекстах реализовано с использованием трех различных подходов к взвешиванию каждого гиперонима в иерархическом контексте: tf, idf и tf–idf [22], причем под «термином» понимается гипероним, а под «документом» понимается синсет. В целях повышения производительности, используется традиционный прием параллелизма по данным: каждый синсет обрабатывается независимо в отдельном

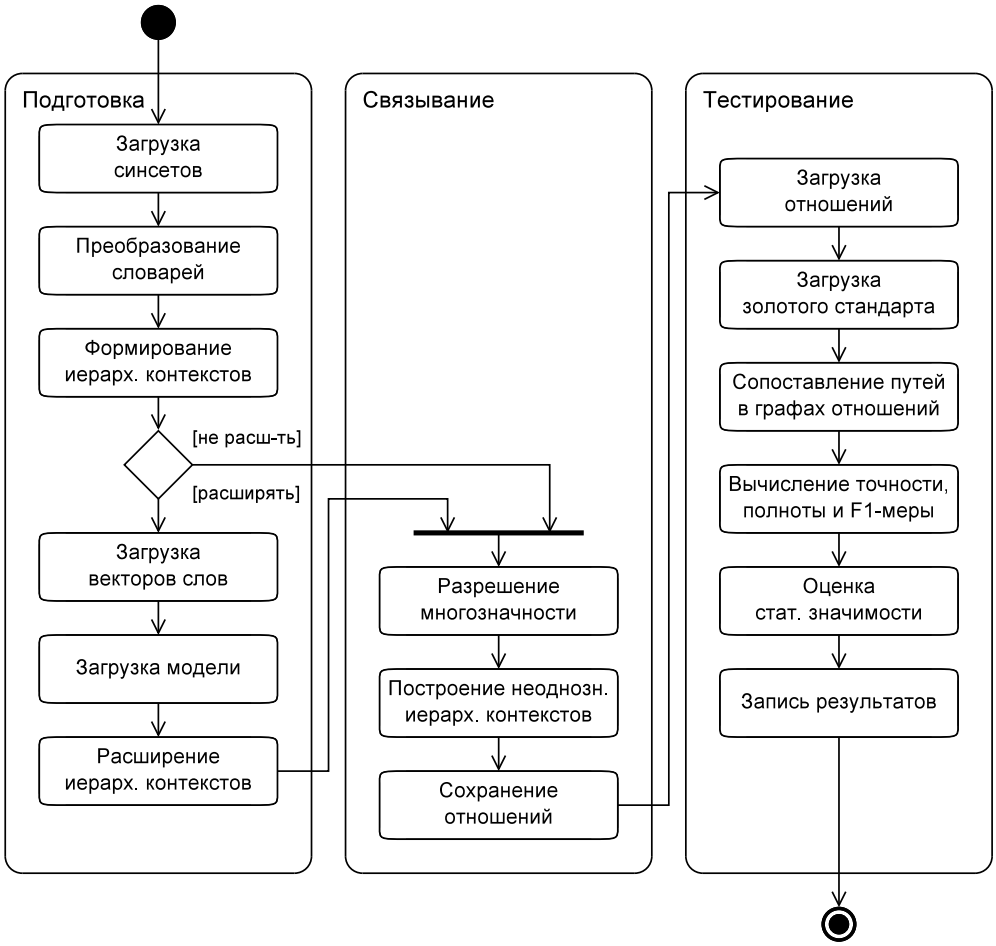


Рис. 6. UML-диаграмма активности построения отношений

процессе. Для каждого слова h в иерархическом контексте $\mathrm{hctx}(S)$ синсета S определяется номер значения путем максимизации косинусной меры близости [7]:

$$\hat{h} = \underset{\substack{S' \in \mathcal{S}, S \neq S', h' \in S', \\ \mathrm{words}(\{h'\}) = \{h\}}}{\arg\max} \cos(\mathrm{hctx}(S), S'). \tag{4}$$

При построении иерархических контекстов со снятой неоднозначностью используется только n гиперонимов, получившие максимальный вес по итогам выполнения этапа разрешения многозначности. Семантическая сеть слов сохраняется в текстовый файл. При оценке качества загружаются полученные отношения и отношения между словами золотого стандарта в виде ориентированных графов. Затем, для каждого отношения определяется существование пути от гипонима к гипериниму в графе золотого стандарта. Вычисляются значения информационно-поисковых критериев точности, полноты и F_1-меры [22] и оценивается статистическая значимость значения каждого критерия [23]. После выполнения всех указанных процедур осуществляется запись результатов оценки в текстовый файл.

2.4. Модуль LRWC

На рис. 7 представлена UML-диаграмма активности построения оценочного набора данных при помощи краудсорсинга на основе выполнения микрозадач, именуемая LRWC (сокр. англ. *Lexical Relations from the Wisdom of the Crowd* — «мудрость толпы о семантических отношениях»). Активность состоит из двух условных шагов [7]: подготовка

заданий и их размещение на платформе краудсорсинга (программа *Emitter*) и получение суждений участников процесса краудсорсинга о корректности семантических отношений (программа *Fetcher*).

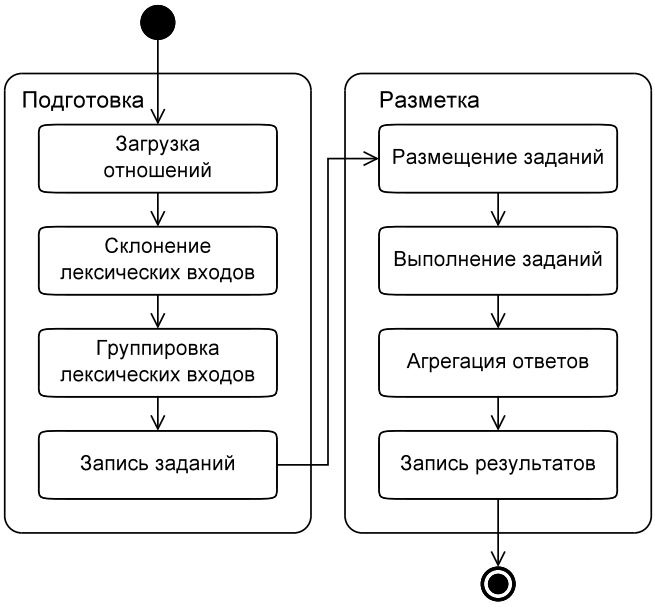


Рис. 7. UML-диаграмма активности построения оценочного набора данных при помощи краудсорсинга на основе выполнения микрозадач

Оценочный набор данных составляется для семантических отношений между значениями слов, т. е. для множества упорядоченных пар вида (*котенок*[1], *млекопитающее*[1]). От участника процесса краудсорсинга требуется предоставить положительный или отрицательный ответ на вопрос вида «Правда ли *котенок* — это разновидность *млекопитающего*?». Для этого вышестоящее слово приводится в форму родительного падежа при помоши морфологического анализатора. Каждая пара слов оценивается независимо несколькими разными участниками, после чего производится агрегация ответов всех участников на все пары слов и сохранение оценочного набора данных в файл.

Заключение

В работе представлен комплекс программ SWN, осуществляющий построение семантической сети слов и ее запись в форматах Семантической паутины на основе информационной модели (рис. 2). Описана архитектура комплекса программ, включающего в себя программы обнаружения понятий, построения и расширения семантических отношений, построения оценочного набора данных. Программы написаны с использованием параллелизма по данным, что позволяет использовать вычислительные узлы с большим количеством доступных ядер центрального процессора для ускорения вычислений. Кроме того, при реализации методов использованы высокоэффективные внешние библиотеки, такие как scikit-learn [14] и TensorFlow [18]. В настоящее время все программы функционируют в режиме командной строки. Связывание программ, написанных на разных языках программирования, осуществляется путем перенаправления потоков стандартного ввода и вывода в сценариях командного процессора Bash и утилиты make. Входные данные представлены в виде текстовых файлов, поля в которых разделены

знаком табуляции. Все промежуточные и итоговые результаты, кроме матрицы линейного преобразования, также представлены в текстовом виде. Кроме того, итоговый результат записывается в формате N-Triples [20].

Эксперименты по построению семантической сети слов для русского языка при помощи данного комплекса программ подтверждают его эффективность по сравнению с аналогичными решениями [6–8]. Важной особенностью методов, лежащих в основе представленного комплекса программ, является независимость от высококачественного исходного семантического ресурса для построения и связывания понятий. Исходный код разработанных программ доступен на GitHub [25–27] вместе с инструкциями по использованию. Среди возможных направлений развития комплекса программ SWN отмечается его интеграция в системы интеллектуального анализа данных, разработка графического интерфейса пользователя, а также использование аппаратных ускорителей вычислений для увеличения производительности программ кластеризации графа и связывания значений слов. В настоящее время комплекс программ предназначен для работы только под операционной системой Linux; поддержка других операционных систем, в т. ч. Windows, не предусмотрена.

Исследование выполнено при финансовой поддержке РФФИ в рамках научного проекта № 16-37-00354 мол_а и при финансовой поддержке РГНФ в рамках научного проекта № 16-04-12019 «Интеграция тезаурусов RussNet и YARN». Авторы благодарят компанию Microsoft Research за предоставленные вычислительные ресурсы в облачной среде Microsoft Azure в рамках программы Azure for Research.

Литература

1. Gonçalo Oliveira H., Gomes P. ECO and Onto.PT: a Flexible Approach for Creating a Portuguese Wordnet Automatically. Language Resources and Evaluation. 2014. Vol. 48, No. 2. P. 373–393. DOI: 10.1007/s10579-013-9249-9.

2. Лукашевич Н.В. Тезаурусы в задачах информационного поиска. М.: Изд-во МГУ, 2011. 512 с.

3. Wong W. et al. Ontology Learning from Text: A Look Back and into the Future. ACM Computing Surveys. 2012. Vol. 44, No. 4. P. 20:1–20:36. DOI: 10.1145/2333112.2333115.

4. Navigli R., Ponzetto S.P. BabelNet: The Automatic Construction, Evaluation and Application of a Wide-Coverage Multilingual Semantic Network. Artificial Intelligence. Vol. 193. P. 217–250. DOI: 10.1016/j.artint.2012.07.001.

5. Camancho Collados J., Pilehvar M.T., Navigli R. Nasari: Integrating Explicit Knowledge and Corpus Statistics for a Multilingual Representation of Concepts and Entities. Artificial Intelligence. Vol. 240. P. 36–64. DOI: 10.1016/j.artint.2016.07.005.

6. Усталов Д.А. Обнаружение понятий в графе синонимов // Вычислительные технологии. 2017. Т. 22, Специальный выпуск 1. С. 99–112. URL: `http://depot.nlpub.ru/ustalov.jct2017.pdf` (дата обращения: 25.04.2017).

7. Усталов Д.А. Построение семантической сети слов путем расширения иерархических контекстов // Компьютерная лингвистика и интеллектуальные технологии: По материалам ежегодной Международной конференции «Диалог» (Москва, 31 мая — 3

июня 2017 г.). М.: Изд-во РГГУ, 2017. В печати. URL: http://depot.nlpub.ru/ustalov.dialog2017.pdf (дата обращения: 06.05.2017).

8. Ustalov D.A., Arefyev N.V., Biemann C., Panchenko A.I. // Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers. Association for Computational Linguistics, 2017, P. 543–550. URL: https://aclweb.org/anthology/E/E17/E17-2087.pdf (дата обращения: 10.04.2017).

9. Berners-Lee T., Hendler J., Lassila O. The Semantic Web. Scientific American. 2001. Vol. 284, No. 5. P. 28–37. URL: https://www.scientificamerican.com/article/the-semantic-web/ (дата обращения: 10.03.2017).

10. Lohmann S. et al. Visualizing Ontologies with VOWL. Semantic Web. 2016. Vol. 7, No. 4. P. 399–419. DOI: 10.3233/SW-150200.

11. van Assem M. et al. A Method to Convert Thesauri to SKOS // 3rd European Semantic Web Conference, ESWC 2006 Budva, Montenegro, June 11–14, 2006 Proceedings. Springer Berlin Heidelberg, 2006. P. 95–109. DOI: 10.1007/11762256_10.

12. McCrae J., Spohr D., Cimiano P. Linking Lexical Resources and Ontologies on the Semantic Web with Lemon // The Semantic Web: Research and Applications: 8th Extended Semantic Web Conference, ESWC 2011, Heraklion, Crete, Greece, May 29–June 2, 2011, Proceedings, Part I. Springer Berlin Heidelberg, 2011. P. 245–259. DOI: 10.1007/978-3-642-21034-1_17.

13. Усталов Д.А. Тезаурусы русского языка в виде открытых связанных данных // Компьютерная лингвистика и интеллектуальные технологии: По материалам ежегодной Международной конференции «Диалог» (Москва, 27 — 30 мая 2015 г.). М.: Изд-во РГГУ, 2015. С. 616–625. URL: http://www.dialog-21.ru/digests/dialog2015/materials/pdf/UstalovDA.pdf (дата обращения: 21.02.2017).

14. Pedregosa F. et al. Scikit-Learn: Machine Learning in Python // Journal of Machine Learning Research. 2011. Vol. 12. P. 2825–2830. URL: http://www.jmlr.org/papers/v12/pedregosa11a.html (дата обращения: 07.03.2017).

15. Biemann C. Chinese Whispers: An Efficient Graph Clustering Algorithm and Its Application to Natural Language Processing Problems // Proceedings of the First Workshop on Graph Based Methods for Natural Language Processing. Association for Computational Linguistics, 2006. P. 73–80. URL: http://dl.acm.org/citation.cfm?id=1654774 (дата обращения: 15.03.2017).

16. van Dongen S. Graph Clustering by Flow Simulation. Ph.D. Thesis. University of Utrecht, 2000. URL: https://dspace.library.uu.nl/handle/1874/848 (дата обращения: 27.03.2017).

17. Řehuřek R., Sojka P. Software Framework for Topic Modelling with Large Corpora // New Challenges for NLP Frameworks Programme: A workshop at LREC 2010. European Language Resources Association, 2010. P. 51–55. URL: https://radimrehurek.com/gensim/lrec2010_final.pdf (дата обращения: 03.04.2017).

18. Abadi M. et al. TensorFlow: A System for Large-Scale Machine Learning // 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16). USENIX Association, 2016. P. 265–283. URL: https://www.usenix.org/conference/osdi16/technical-sessions/presentation/abadi (дата обращения: 10.04.2017).

19. Hagberg A.A., Schult D.A., Swart P.J. Exploring Network Structure, Dynamics, and Function using NetworkX // Proceedings of the 7th Python in Science Conference. 2008. P. 11–15. URL: http://conference.scipy.org/proceedings/scipy2008/paper_2/ (дата обращения: 05.12.2016).

20. Beckett D. The Design and Implementation of the Redland RDF Application Framework. Computer Networks. 2002. Vol. 39, No. 5. P. 577–588. DOI: 10.1016/S1389-1286(02)00221-9.

21. Korobov M. Morphological Analyzer and Generator for Russian and Ukrainian Languages. Analysis of Images, Social Networks and Texts: 4th International Conference, AIST 2015, Yekaterinburg, Russia, April 9–11, 2015, Revised Selected Papers. Springer International Publishing, 2015. P. 320–332. DOI: 10.1007/978-3-319-26123-2_31.

22. Manning C.D., Raghavan P., Schütze H. Introduction to Information Retrieval. Cambridge University Press, 2008. 506 p.

23. Riedl M., Biemann C. Unsupervised Compound Splitting With Distributional Semantics Rivals Supervised Methods // Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. Association for Computational Linguistics, 2016. P. 617–622. URL: https://aclweb.org/anthology/N/N16/N16-1075.pdf (дата обращения: 16.02.2017).

24. Fu R. et al. Learning Semantic Hierarchies via Word Embeddings. Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). Association for Computational Linguistics, 2014. P. 1199–1209. URL: https://aclweb.org/anthology/P/P14/P14-1113.pdf (дата обращения: 26.04.2016).

25. dustalov/watset: Concept Discovery from Synonymy Graphs. URL: https://github.com/dustalov/watset (дата обращения: 10.04.2017).

26. dustalov/watlink: Concept Linking. URL: https://github.com/dustalov/watlink (дата обращения: 10.04.2017).

27. dustalov/projlearn: Learning Word Subsumption Projections. URL: https://github.com/dustalov/projlearn (дата обращения: 10.04.2017).

Усталов Дмитрий Алексеевич, м.н.с., Институт математики и механики им. Н.Н. Красовского УрО РАН; ассистент, кафедра высокопроизводительных компьютерных технологий, Уральский федеральный университет (Екатеринбург, Российская Федерация)

Созыкин Андрей Владимирович, к.т.н., Институт математики и механики им. Н.Н. Красовского УрО РАН; зав. каф., кафедра высокопроизводительных компьютерных технологий, Уральский федеральный университет (Екатеринбург, Российская Федерация)

DOI: 10.14529/cmse170205

A SOFTWARE SYSTEM FOR AUTOMATIC CONSTRUCTION OF A SEMANTIC WORD NETWORK

© 2017 D.A. Ustalov[1,2], A.V. Sozykin[1,2]

[1]*Krasovskii Institute of Mathematics and Mechanics (ul. Sofii Kovalevskoy 16, Yekaterinburg, 620990 Russia),*

[2]*Ural Federal University (ul. Mira 19, Yekaterinburg, 620002 Russia)*

E-mail: dau@imm.uran.ru

Received: 01.05.2017

A semantic word network is a network that represents the semantic relations between individual words or their lexical senses. In this paper, we present a software system for automatic construction of a semantic word network. The system, called SWN, is designed for the construction for such a semantic word network and includes the implementation of unsupervised concept discovery and semantic relation establishing methods as well as the implementation of a supervised relation expansion method. The methods use widely available language resources, such as semantic relation dictionaries and background text corpora. The domain model has been presented using the VOWL notation. The system architecture is represented using the UML notation and is composed of the concept discovery module, semantic relation construction module, the Semantic Web export module, and the evaluation dataset construction module based on microtask-based crowdsourcing. The present software system is open source and is available for integration into third-party data mining systems.

Keywords: semantic network, lexical semantics, software engineering, free software, Semantic Web, VOWL, UML.

FOR CITATION

Ustalov D.A., Sozykin A.V. A Software System for Automatic Construction of a Semantic Word Network. *Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering.* 2017. vol. 6, no. 2. pp. 69–83. (in Russian) DOI: 10.14529/cmse170205.

References

1. Gonçalo Oliveira H., Gomes P. ECO and Onto.PT: a Flexible Approach for Creating a Portuguese Wordnet Automatically. Language Resources and Evaluation. 2014. vol. 48, no. 2. pp. 373–393. DOI: 10.1007/s10579-013-9249-9.

2. Loukachevitch N.V. *Tezaurusy v zadachakh informatsionnogo poiska* [Thesauri in Information Retrieval Tasks]. Moscow, MSU Publishing, 2011. 512 p.

3. Wong W. et al. Ontology Learning from Text: A Look Back and into the Future. ACM Computing Surveys. 2012. vol. 44, no. 4. pp. 20:1–20:36. DOI: 10.1145/2333112.2333115.

4. Navigli R., Ponzetto S.P. BabelNet: The Automatic Construction, Evaluation and Application of a Wide-Coverage Multilingual Semantic Network. Artificial Intelligence. vol. 193. pp. 217–250. DOI: 10.1016/j.artint.2012.07.001.

5. Camancho Collados J., Pilehvar M.T., Navigli R. Nasari: Integrating Explicit Knowledge and Corpus Statistics for a Multilingual Representation of Concepts and Entities. Artificial Intelligence. vol. 240. pp. 36–64. DOI: 10.1016/j.artint.2016.07.005.

6. Ustalov D.A. Concept Discovery from Synonymy Graphs. *Vychislitel'nye tekhnologii* [Computational Technologies]. 2017. vol. 22, Special Issue 1. pp. 99–112. Available at: `http://depot.nlpub.ru/ustalov.jct2017.pdf` (accessed: 25.04.2017).

7. Ustalov D.A. Expanding Hierarchical Contexts for Constructing a Semantic Word Network. *Komp'yuternaya lingvistika i intellektual'nye tekhnologii: Po materialam ezhegodnoi Mezhdunarodnoi konferentsii «Dialog» (Moskva, 31 maya — 3 iyunya 2017 g.)* [Computational Linguistics and Intellectual Technologies: papers from the Annual conference "Dialogue" (Moscow, May 31–June 3, 2017)]. Moscow, RSUH, 2017. In press. Available at: `http://depot.nlpub.ru/ustalov.dialog2017.pdf` (accessed: 06.05.2017).

8. Ustalov D.A., Arefyev N.V., Biemann C., Panchenko A.I. Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers. Association for Computational Linguistics, 2017, pp. 543–550. Available at: `https://aclweb.org/anthology/E/E17/E17-2087.pdf` (accessed: 10.04.2017).

9. Berners-Lee T., Hendler J., Lassila O. The Semantic Web. Scientific American. 2001. vol. 284, no. 5. pp. 28–37. Available at: `https://www.scientificamerican.com/article/the-semantic-web/` (accessed: 10.03.2017).

10. Lohmann S. et al. Visualizing Ontologies with VOWL. Semantic Web. 2016. vol. 7, no. 4. pp. 399–419. DOI: 10.3233/SW-150200.

11. van Assem M. et al. A Method to Convert Thesauri to SKOS. 3rd European Semantic Web Conference, ESWC 2006 Budva, Montenegro, June 11–14, 2006 Proceedings. Springer Berlin Heidelberg, 2006. pp. 95–109. DOI: 10.1007/11762256_10.

12. McCrae J., Spohr D., Cimiano P. Linking Lexical Resources and Ontologies on the Semantic Web with Lemon. The Semantic Web: Research and Applications: 8th Extended Semantic Web Conference, ESWC 2011, Heraklion, Crete, Greece, May 29–June 2, 2011, Proceedings, Part I. Springer Berlin Heidelberg, 2011. pp. 245–259. DOI: 10.1007/978-3-642-21034-1_17.

13. Ustalov D.A. Russian Thesauri as Linked Open Data. *Komp'yuternaya lingvistika i intellektual'nye tekhnologii: Po materialam ezhegodnoi Mezhdunarodnoi konferentsii «Dialog» (Moskva, 27 — 30 maya 2015 g.)* [Computational Linguistics and Intellectual Technologies: papers from the Annual conference "Dialogue" (Moscow, May 27–30, 2015)]. Moscow, RSUH, 2015, pp. 616–625. Available at: `http://www.dialog-21.ru/digests/dialog2015/materials/pdf/UstalovDA.pdf` (accessed: 21.02.2017).

14. Pedregosa F. et al. Scikit-Learn: Machine Learning in Python. Journal of Machine Learning Research. 2011. vol. 12. pp. 2825–2830. Available at: `http://www.jmlr.org/papers/v12/pedregosa11a.html` (accessed: 07.03.2017).

15. Biemann C. Chinese Whispers: An Efficient Graph Clustering Algorithm and Its Application to Natural Language Processing Problems. Proceedings of the First Workshop on Graph Based Methods for Natural Language Processing. Association for Computational Linguistics, 2006. pp. 73–80. Available at: `http://dl.acm.org/citation.cfm?id=1654774` (accessed: 15.03.2017).

16. van Dongen S. Graph Clustering by Flow Simulation. Ph.D. Thesis. University of Utrecht, 2000. Available at: `https://dspace.library.uu.nl/handle/1874/848` (accessed: 27.03.2017).

17. Řehuřek R., Sojka P. Software Framework for Topic Modelling with Large Corpora. New Challenges for NLP Frameworks Programme: A workshop at LREC 2010. European Language Resources Association, 2010. pp. 51–55. Available at: `https://radimrehurek.com/gensim/lrec2010_final.pdf` (accessed: 03.04.2017).

18. Abadi M. et al. TensorFlow: A System for Large-Scale Machine Learning. 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16). USENIX Association, 2016. pp. 265–283. Available at: https://www.usenix.org/conference/osdi16/technical-sessions/presentation/abadi (accessed: 10.04.2017).

19. Hagberg A.A., Schult D.A., Swart P.J. Exploring Network Structure, Dynamics, and Function using NetworkX. Proceedings of the 7th Python in Science Conference. 2008. pp. 11–15. Available at: http://conference.scipy.org/proceedings/scipy2008/paper_2/ (accessed: 05.12.2016).

20. Beckett D. The Design and Implementation of the Redland RDF Application Framework. Computer Networks. 2002. vol. 39, no. 5. pp. 577–588. DOI: 10.1016/S1389-1286(02)00221-9.

21. Korobov M. Morphological Analyzer and Generator for Russian and Ukrainian Languages. Analysis of Images, Social Networks and Texts: 4th International Conference, AIST 2015, Yekaterinburg, Russia, April 9–11, 2015, Revised Selected Papers. Springer International Publishing, 2015. pp. 320–332. DOI: 10.1007/978-3-319-26123-2_31.

22. Manning C.D., Raghavan P., Schütze H. Introduction to Information Retrieval. Cambridge University Press, 2008. 506 p.

23. Riedl M., Biemann C. Unsupervised Compound Splitting With Distributional Semantics Rivals Supervised Methods. Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. Association for Computational Linguistics, 2016. pp. 617–622. Available at: https://aclweb.org/anthology/N/N16/N16-1075.pdf (accessed: 16.02.2017).

24. Fu R. et al. Learning Semantic Hierarchies via Word Embeddings. Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). Association for Computational Linguistics, 2014. pp. 1199–1209. Available at: https://aclweb.org/anthology/P/P14/P14-1113.pdf (accessed: 26.04.2016).

25. Ustalov D.A. dustalov/watset: Concept Discovery from Synonymy Graphs. Available at: https://github.com/dustalov/watset (accessed: 10.04.2017).

26. Ustalov D.A. dustalov/watlink: Concept Linking. Available at: https://github.com/dustalov/watlink (accessed: 10.04.2017).

27. Ustalov D.A. dustalov/projlearn: Learning Word Subsumption Projections. Available at: https://github.com/dustalov/projlearn (accessed: 10.04.2017).

DOI: 10.14529/cmse170206

DYNAMIC ROUTING ALGORITHMS AND METHODS FOR CONTROLLING TRAFFIC FLOWS OF CLOUD APPLICATIONS AND SERVICES*

© 2017 г. I.P. Bolodurina, D.I. Parfenov
Orenburg State University
(460018 Orenburg, Pobedy avenue, 13),
E-mail: prmat@mail.osu.ru, fdot_ it@mail.osu.ru
Received: 01.05.2017

Nowadays, we see a steady growth in the use of cloud computing in modern business. This enables to reduce the cost of IT infrastructure owning and operation; however, there are some issues related to the management of data processing centers. One of these issues is the effective use of companies' computing and network resources. The goal of optimization is to manage the traffic in cloud applications and services within data centers. Taking into account the multitier architecture of modern data centers, we need to pay a special attention to this task. The advantage of modern infrastructure virtualization is the possibility to use software-defined networks and software-defined data storages. However, the existing optimization of algorithmic solutions does not take into account the specific features of the network traffic formation with multiple application types.

The task of optimizing traffic distribution for cloud applications and services can be solved by using software-defined infrastructure of virtual data centers. We have developed a simulation model for the traffic in software-defined networks segments of data centers involved in processing user requests to cloud application and services within a network environment. Our model enables to implement the traffic management algorithm of cloud applications and optimize the access to storage systems through the effective use of data transmission channels.

During the experimental studies, we have found that the use of our algorithm enables to decrease the response time of cloud applications and services and, therefore, increase the productivity of user requests processing and reduce the number of refusals.

Keywords: software-defined network, data centers, cloud computing, IT infrastructure.

FOR CITATION

Bolodurina I.P., Parfenov D.I. Dynamic Routing Algorithms and Methods for Controlling traffic Flows of Cloud Applications and Services. *Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering.* 2017. vol. 6, no. 2. pp. 84–98. DOI: 10.14529/cmse170206.

Introduction

Nowadays, we see a steady growth in the use of cloud computing in modern business. This enables to reduce the cost of IT infrastructure owning and operation due to more efficient use of companies' computing and network resources. At present, the solutions for virtual infrastructure are dynamically developing. Thus, the container technology has been lately used for placing cloud applications and services within virtual data centers. Container technologies are mostly based on Docker. Besides, modern data centers rather use virtual infrastructures instead of physical infrastructures especially based on software-defined components: networks, data storages [2, 3], etc. This changes the mechanisms launch and placement management of applications and services.

* The paper is recommended for publication by the Program Committee of the International Scientific Conference "Parallel Computational Technologies (PCT) 2017".

The most significant changes are found in the SaaS and PaaS service scheduling [4]. Conventional methods of scheduling and distributing resources using the maximum responses to user requests in the High Performance Computing (HPC) systems do not prove to be efficient [1, 5]. This is because the response time in the HPC is less significant than in cloud systems.

Thus, it is important to develop effective scheduling and resource distributing methods for cloud systems that optimize the response time in user requests. To assess the methods, we have developed a simulation model for the traffic in software-defined network segments of data centers involved in the processing of user requests to cloud application and service within a network environment.

The paper is organized as follows. Section 1 is devoted to describing structural model of virtual data centers. Section 2 is devoted to describing simulation model software-defined infrastructure of a virtual data center has several heterogeneous applications and services. Section 3 provides describes an optimization algorithm of adaptive routing and balancing of the application and services flows in a heterogeneous cloud platform, which is located in the data processing center. Section 4 provides experimental results of our investigation. Section 5 is devoted to describing traditional approaches to route traffic based on load-balancing and solution proposed by world scientists for solving this problem. Conclusion section includes summary of our investigation as well as future work overview.

1. A structural model of virtual data centers

To understand the operation principles of a cloud application, we need to define its place in the infrastructure of a virtual data center. A virtual data center is a dynamic object, changing in time t, its state can be formalized as a directed graph of the following form:

$$VirtDC = \big(Node(t), Connect(t)\big), \tag{1}$$

where vertices $Node(t) = \{Node_1, \cdots, Node_n\}$ — are active elements of the virtual data center infrastructure (computing nodes, storage systems and others); directed edges $Connect(t) = \{Connect_1, \cdots, Connect_v\}$ — are active user connections to the cloud applications.

The specific feature of cloud applications is the approach, where users have access to them and to their services; however, they do not know anything about their actual location. In most cases, users only know the address of the aggregation node and the application name. The cloud system automatically selects the optimal virtual machine for the request, on which it is to be processed.

Before we talk about the ways of placing cloud applications in a virtual data center, we need to determine their structure, basic parameters, and key characteristics of their operation affecting the efficiency of their use. For this purpose, we have developed a generalized model of a cloud application.

The generalized model of a cloud application is a multilayer structure, described in a form of graphs to characterize the connections of individual elements. The model can be represented in the form of three basic layers, detailing the connections of the specific objects of virtual cloud infrastructure: applications, related services and provided resources.

The cloud application is a weighted directed acyclic graph of data dependencies:

$$CloudAppl = (G, V), \tag{2}$$

Its vertices G are tasks that get information from the sources and process it in accordance with the user requests; its directed edges V between corresponding vertices are a link between

tasks in a schedule plan. The schedule plan is defined as a procedure which is prepared to follow the user's request ($SchemeTask$).

Each vertex $g \in G$ is characterized by the following tuple:

$$g = (Res, NAppl, Utime, SchemeTask), \quad (3)$$

where Res are the resource requirements; $NAppl$ is the number of application instances; $Utime$ is the estimated time for of the users' request execution; $SchemeTask$ is a communication scheme of data transmission between sources and computing nodes.

Each directed edge $v \in V$ connects the application with the required data source. It is characterized by the following tuple:

$$v = (u, v, Tdata, Mdata, Fdata, Vdata, Qdata), \quad (4)$$

where u and v are linked vertices; $Tdata$ is the type of transmitted data; $Mdata$ is the access method to the data source; $Fdata$ is the physical type of the accessed object (a file in the storage system, a local file, distributed database, data services and so on); $Vdata$ is the traffic volume estimated by the accessed data (in Mb), $Qdata$ is the requirements for the QoS (quality of services).

The model is original because it enables to calculate the consolidated assessment of its work with data sources for each application. It allows predicting the performance of the whole cloud system.

As mentioned earlier, a cloud service is one of the key slices in the generalized model of a cloud application. The cloud service is an autonomous data source for the application, for which it acts as a consolidated data handler. Generally, the cloud service is highly specialized and designed to perform a limited set of functions. The advantage of connecting a cloud application to the service is the isolated data processing, in contrast to direct access to the raw data, when a cloud application does not use a service. The usage of services reduces the execution time for user requests. The cloud service is described as a directed graph of data dependencies. The difference lies in the fact that from the user's viewpoint, the cloud service is a closed system.

The cloud service can be formalized as a tuple:

$$CloudServ = (ArgIP, NameServ, FormatIN, FormatOUT), \quad (5)$$

where $AgrIP$ is the address of aggregation computing node; $NameServ$ is the service name; $FormatIN$ is the format of input data; $FormatOUT$ is the format of output data.

The aggregator of a service selects the optimal virtual machine; it is executed on this machine. In addition, all its applications are distributed between predefined virtual machines or physical servers. Their new instances are scaled dynamically depending on the number of incoming requests from cloud applications, users or other services.

To describe the placement of cloud applications and services in the data center infrastructure, we have also implemented the model of a cloud resource. A cloud resource is an object of a data center, which describes the behavior and characteristics of the individual infrastructure elements, depending on its current state and parameters. The objects of data center are disk arrays including detached storage devices, virtual machines, software-defined storages, various databases (SQL/NoSQL) and others. In addition, each cloud service or application imposes requirements on the number of computing cores, RAM and disk sizes, and the presence of special libraries on physical or virtual nodes used to launch their executing environments.

Each cloud resource can be formalized as follows:

$$CloudRes = (TRes, Param, State, Core, Rmem, Hmem, Lib), \quad (6)$$

where *TRes* is the type of resource; *Param* is the set of parameters; *State* is the state of resource; *Core* is the number of computing cores; *Rmem* is the size of RAM; *Hmem* is the size of a disk; *Lib* are for the libraries requirements.

The distinctive feature of the model suggested implies analyzing cloud resources from the user viewpoint and from the viewpoint of a software-defined infrastructure of the virtual data center. The model is innovative, since it simultaneously describes the application data placements and the state of the virtual environment, taking into account the network topology.

We have developed the model of the software-defined storage, which details the resource model of the virtual data center. It is represented in the form of a directed multigraph; its vertices are the virtual data center elements, which are responsible for application data placement (e.g. virtual disk arrays, DBMS and so on):

$$Stg_{ki} = \left(MaxV_{ki}, P_{ki}^{stg}, Vol_{ki}(t), \bar{R}_{ki}(t), \bar{W}_{ki}(t), S_{ki}^{stg}(t)\right), \quad (7)$$

where $MaxV_{ki} \in N$ is the maximum storage capacity in Mb; $P_{ki}^{stg} = \left\{P_{ki}^{stg}{}_{j}\right\}$ is the set of network ports; $Vol_{ki}(t) \in N \cup \{0\}$ is the available storage capacity in Mb; $\bar{R}_{ki}(t)$ and $\bar{W}_{ki}(t)$ are read and write speeds; $S_{ki}^{stg}(t) \in \{\text{“}online\text{”}, \text{“}offline\text{”}\}$ is state of software-defined storage.

The data storage system for applications is like a layer cake. It uses the principles of self-organization of resources. The basis of self-organization of data storages is an adaptive model of dynamic reconfiguration when resources are changing. The model allows optimizing the organizational structure of the cloud platform based on algorithms for searching optimal control nodes and allocating control groups. Our control model assumes two control levels for nodes and resources.

When a software-defined storage is created on each virtual computing node, the software module for exchanging state data between devices is executed. This exchange is carried out within a group of nodes by a single storage method. The least loaded node in the group is selected as the control node. This approach reduces the risk of the control node degradation.

If the control node is failed, the remaining group of virtual machines has all the information about each other, which allows choosing a new control node automatically. Each control node also carries out cooperation with control nodes from other groups to maintain up-to-date information on the state of the entire system.

Thus, the system of software-defined storages is constructed as a hierarchy that includes three basic levels: the level of local access, the level of the controlled group, and the level of data exchange within the whole system. In our model, the description of cloud applications consists of task descriptions and data source descriptions specifying directions and methods of data transfer as well as required resources.

The model has been used for the development of our data assignment algorithm for software-defined storages, and for our scheduling algorithm for cloud services and applications within the cloud platform.

2. Research methods

Generally, the software-defined infrastructure of a virtual data center has several heterogeneous applications and services. We can assume that the network of a virtual data

center encompasses at least three types of application traffic: web-applications, case-applications available on DaaS or SaaS models, and video services. To generate user requests in the simulation model, we apply weight coefficients k_1, k_2, k_3 for each traffic type. Each coefficient allows us to classify requests into types and affects the following set of parameters: running time, routes, priority in the process queue, request intensity, and the distribution law for each type of traffic.

Presented as a multi-channel queuing system, the simulation model of the software-defined infrastructure of the virtual data center includes a user request source (I), a queue (Qs) and a scheduler (S) who manages application hosting and its launch (App). Besides, it contains computing cluster (Srv) and systems of data center storage (Stg). The queuing system is represented in Fig. 1.

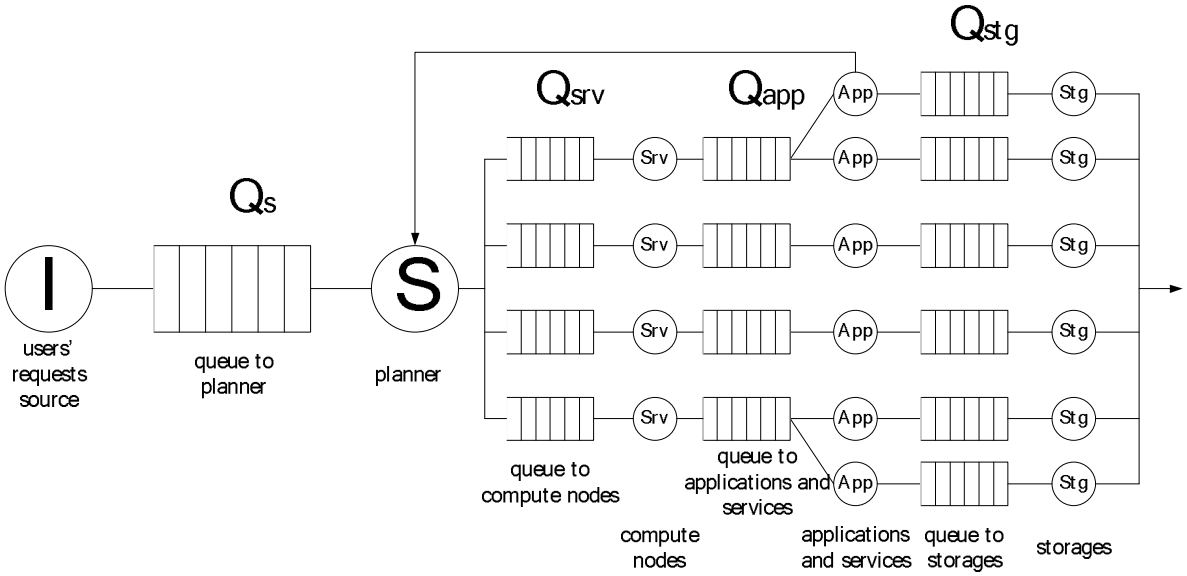


Fig. 1. A queuing system scheme of the software-defined infrastructure of the virtual data center

A queuing system model is stochastic. For its operation, it is necessary to make a user request flow to cloud applications and services with account for the distribution laws and request intensities for each type of cloud applications and services.

To optimize application distribution in the cloud environment of a virtual data center, it is necessary to determine the traffic distribution laws for each application type and distribute the traffic into access objects (virtual servers, containers, and storage systems). For this purpose, it is necessary to set a certain route and make the control law for it within the time interval $T = [t_1, t_2]$.

The dynamic of traffic in cloud applications and services of the software-defined infrastructure of a virtual data processing center can be described by the following discrete system:

$$x_{i,j}(t + \Delta t) = x_{i,j}(t) - \sum_{k=1}^{K}\sum_{l=1}^{N} s_{i,j}(t) u_{i,l}^{j,k}(t) + \sum_{m=1}^{N} s_{m,i}(t) u_{m,l}^{j}(t) + y_{i,j}(t) \qquad (8)$$

where N is the number of virtual nodes within the network; K is the number of application types within the network; $s_{i,j}(t)$ is the capacity of the channels between i-th computing node and j-th storage system $(i \neq j)$; $y_{i,j}(t) = \lambda_{i,j}(t)\Delta t$ is the traffic volume (the number of user

requests) at the moment t on the virtual node i-th, intended for transferring to the storage system j-th; $\lambda_{i,j}(t)$ is the intensity of incoming load, which is defined as the total intensity of the user request flow connecting to the virtual node i-th and using the storage system j-th; $u_{i,l}^{j,k}(t)$ is a part of the channel transmission capacity in a certain segment of the software-defined network (i,l) at the moment t for the user request flow to the application of type k, working with the data storage system j-th.

To exclude the possibility of overloading the objects of the virtual data center due to the limited queue buffers on compute nodes, as well as to use data transmission channel capacity efficiently, a number of restrictions are introduced for network parameters of cloud applications in software-defined networks.

The restrictions related to channel capacity limits can be written as follows:

$$0 \le u_{i,l}^{j}(t) \le u_{i,l}^{j(\max)} \le 1; \sum_{l=1}^{N} u_{i,l}^{j}(t) \le \varepsilon_{i,l}^{j,k} \le 1 \quad (9)$$

where $u_{i,l}^{j(\max)}$ is the limit of channel capacity available for the computing node i-th in the software-defined network segment l-th for traffic transfer to the storage j-th; $\varepsilon_{i,l}^{j,k}$ is the part of the channel capacity for the compute node i-th in the software-defined network segment l-th for the transmission of user requests to the application of type k to the storage system j-th.

The use of a software-defined network within a virtual data center allows controlling the queues, which introduces extra restrictions:

$$0 \le x_{i,j}(t) \le x_{i,j}^{(\max)}; \sum_{l=1}^{N} x_{i,j}(t) \le x_{i}^{(\max)} \quad (10)$$

where $x_{i,j}^{(\max)}$ is the maximum queue length allowed on the i-th compute node for processing the incoming traffic to storage system j-th; $x_{i}^{(\max)}$ is a maximum volume of the buffer allowed on the compute node i-th.

Let us consider the system performance as a criterion of optimality, gained through a fixed period $T = [t_1, t_2]$, which is formalized within the model as an objective function in the following form:

$$\sum_{t=0}^{t-1} \sum_{k=1}^{K} \sum_{i=1}^{N} \sum_{j=1}^{N} s_{i,j}(t) u_{i,l}^{j,k}(t) \to \max \quad (11)$$

To solve the optimization task, we use an iterative method that allows us to explore the dynamics of the system at the interval $T = [t_1, t_2]$ and control channel capacity for a certain type of applications in a software-defined network.

3. Suggested algorithm

Based on the constructed models, we have developed an optimization algorithm of adaptive routing and balancing of the application and services flows in a heterogeneous cloud platform, which is located in the data processing center. The algorithm aims to ensure the efficient management of the application and services' flows under dynamic changes in the load on the communication channels used to deploy data center software-defined networks by reducing the design complexity for optimal route schemes.

Generalized algorithm is as follows:

1. Divide all channels in network in two subsets, ST and SR.

2. Generate optimal routes for data flow of the particular class of applications.

3. Determine the points of entry in two subset, ST and SR.

4. Search for all the alternative routes at minimum cost.

5. Calculate alternative routes for dynamic load change in the communication channel.

6. Form the full list of alternative paths in network.

7. Define whether there were load changes.

If "Yes" then **GO TO** Step **8**, else **GO TO** Step **7**.

8. Define whether you need to change the route for the current data flow.

If "Yes" then **GO TO** Step **9**, else **GO TO** Step **13**.

9. Calculate metrics connection.

10. Define list of other network objects placed higher in the hierarchy, which performance has decreased.

If network objects found then **GO TO** Step **11**, else **GO TO** Step **12**.

11. Define the new minimum length route for every network object, whose performance has been decreased.

12. Design the new optimal route tree.

13. Transfer the current data flows to new routes. Reshape the list of alternative routes.

GO TO Step **7**.

Fig. 2. Generalized plan of optimization algorithm of adaptive routing and balancing of the application and services flows in a heterogeneous cloud platform

Let us describe the algorithm in more detail. At the first step algorithm splits the software-defined network multiple channels into a subset of channels, which are included in the tree of routes passing through the ST set segments, a subset of alternative channels passing through the SR set segments.

At the second step algorithm generates optimal routes in the software-defined network for data flow of the particular class of applications in a software-defined network cloud platform based on the communication channels' weight for each of the subsets.

At the step 3 algorithm determines the point of entry into the tree of optimal routes and in the variety of alternative channels for each of the software-defined network channels. At the next step (Step 4) algorithm searches for all the alternative routes at minimum cost, taking into account the channel's weight in the previously constructed tree of optimal routes of the software-defined network for each network object, which is a leaf of the tree. Bind these lists to a network object incident for the reporting channel, located in the hierarchy below.

At the step 5 if a network object is not a leaf of the tree, then calculate alternative routes in the software-defined network for this object, taking into account the weight of the channel in the previously constructed tree of the optimum routes and select the best value of the

alternative route. This procedure is performed to generate a list of alternative routes in case of a dynamic load change in the communication channel.

At the step 6 algorithm builds the full list of alternative paths in a software-defined network of the data center for each communication hub. At the step 7 algorithm defines whether there were load changes for some connections by analyzing the received protocol information. If they were, move to Step 8, otherwise – to Step 7. At the step 8 algorithm defines whether you need to change the route for the current data flow using the list of alternative routes. If so, move to Step 9, otherwise – to Step 13.

At the step 9 algorithm for a network object with decreased potential and changed metrics connection in the alternative routes list, defines the minimum length route and put the connection, which led to the network object potential decrease into the optimal route tree and the changed path from the optimal paths tree into a set of alternative SDN routes. When required, you should review the calculations of channel's function weight.

After the transfer to the alternative software-defined network route, at the step 10 algorithm should define if the potential of other data center network objects placed higher in the hierarchy has decreased. If so, move to Step 11, otherwise – to Step 12.

At the step 11 algorithm should define the new minimum length route for every network object, whose potential has been decreased. If the new minimal length route for every object of the network includes a route from the alternative route list, you need to put this route into the software-defined network optimal routes tree and put the route from the optimal routes tree into the set of alternative routes.

At the step 12 Algorithm designs the new optimal route tree in a software-defined network of data center. At the last step algorithm transfers the current data flows of applications and services to affordable data center routes, reconsider the tree entry points; and as for the variety of alternative routes, reshape the list of alternative routes for each network entity that has changed. Then go to step 7.

Application of the proposed algorithm for optimizing the adaptive routing of data flow balancing has allowed us to reduce the complexity of calculating the optimum route to the value $O(k N)$, where k is the number of completed transitions to alternative routes, and N is the number of objects in the software-defined network of data center. Thus, the algorithm is designed to speed up the search and selection of optimal routes for application and service data flows arranged in a heterogeneous cloud platform under dynamic load changes on the communication channels.

4. Experimental results

To assess the efficiency of the developed algorithm for optimizing the adaptive routing of applications and services and balancing data flow in a heterogeneous data center cloud platform, we have conducted a pilot study. We have chosen the Openstack cloud system as a basic platform. For comparison, we have applied algorithms used in the OpenFlow version 1.4 for route control of the software-defined network of the data center in the experiment. We have created a data center prototype with basic nodes and software modules for the developed algorithms that redistribute data and applications flows for the pilot study. To verify the developed algorithm of optimal routing and traffic balancing under conditions of dynamic changes of channels in the software-defined network of the data center, we have deployed several experimental networks consisting of 25, 50, 100, 200, 300, and 400 objects. All generated

requests have been reproduced consequently at two pilot sites: with the traditional routing technology (platform 1, NW) and with the technology of software-defined networks (platform 2, SDN). This restriction was caused by the need to compare the results with traditional network infrastructure incapable of dynamic reconfiguration. Two tests were carried out on platform 2. In the first case, we have used the model OpenFlow version 1.4 routing algorithms, in the second case (NEW SDN), we have applied the developed routing optimization algorithm. Experiment time was one hour, which corresponds to the most prolonged period of peak demand recorded in a real traffic network of heterogeneous cloud platform. We have chosen the response time of applications and services that work in a cloud platform as a basic metrics to assess the efficiency of the proposed solutions. The results of the experiment are provided in Fig. 2.

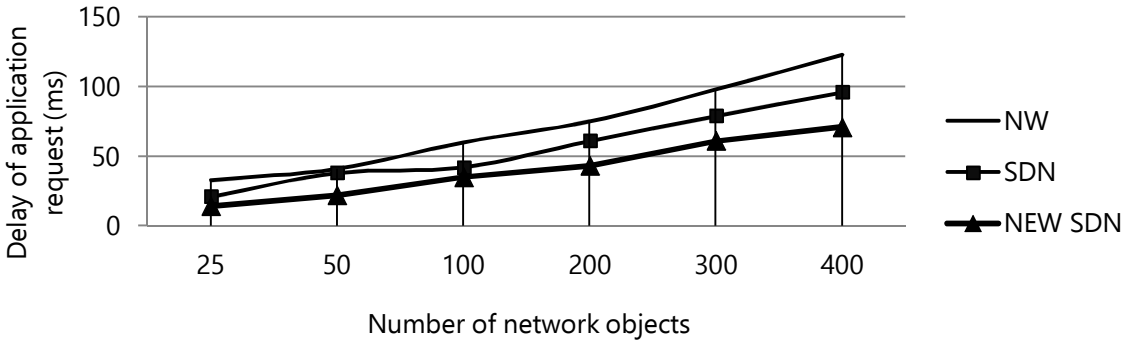


Fig. 3. A schedule of dependence of the response time of applications and services in a heterogeneous cloud platform from the quantities of network objects in the data center

As shown in our research, the algorithm for optimizing the adaptive routing of data flow balancing, based on collected information about alternative routes, has enabled to reduce the response time for applications and services of a heterogeneous cloud platform with a dynamically changing load on channels by 40% compared to traditional networks, and by 25% compared to the model algorithms of the Protocol version 1.4 of OpenFlow. Thus, the algorithm is efficient in designing optimal routes and traffic balancing in SDN data center in case of dynamic changes of load on communication channels.

5. Discussion

Traditional approaches to route traffic based on load-balancing are reactive. They use simple classical algorithms for distributed computing tasks First Fit or Best Fit. Such algorithms as [5–8] First Come First Served Scan, Most Processors First Served Scan, and Shortest Job First Scan are popular too. Their main disadvantage is poor utilization of a computer system due to a large number of windows in the task launch schedule and problem with "hanging up" when their service is postponed indefinitely due to tasks of higher priority. The solution proposed by D. Lifka from Argonne National Laboratory is usually applied as an alternative method of load distribution between nodes. It is based on the aggressive variant of Backfill algorithm [5, 6, 8] and has two conflicting goals – a more efficient use of computing resources by filling the empty windows schedule and prevention of problems with "hanging up" due to redundancy mechanism. D. Feytelson and A. Weil offered a conservative variant of Backfill algorithm [6]. Further, various modifications have been created by B. Lawson and E.

Smyrni [7], D. Perkovic and P. Keleher [9], S. Srinivasan [8]. The main drawback of these algorithms is the time lag during calculation, which is not acceptable for critical services at the time of failure.

In addition to the traditional reactive fault-tolerant technology, such as replication and redundancy to ensure reliability of networked storage cloud platforms, a group of scientists from Nankai University (Ying Lee, Lee Mingze Ganges Schang, Hiaoguang Liu, Zhongschei Lee, and Huiyun Tang) proposed an approach based on the Markov model, which provides secure storage of data without excessive redundancy [10]. However, a significant drawback of this model is the lack of classification and analysis of the types and sources of data to be placed in their consumption. Nevertheless, the model demonstrates a proactive approach that gives certain advantages to achieve the desired resiliency of cloud storage.

Reliability and availability of applications and services play an important role in the assessment of its cloud platform performance. A major shortcoming of existing software reliability solutions in the data center infrastructure is the use of traditional data flow routing methods. In this work, we offer to use the software-defined network technology to adjust the network to the current load of the applications and services that are hosted in a cloud platform before they start using pre-computed and installation routes of transmission (in case of known oriented acyclic graph task dependencies and communication schemes). The principles of a software-defined network first emerged in research laboratories at Stanford and Berkeley [11], and are currently being developed by the Open Network Foundation consortium, GENI project, the European project OFELIA [13] and the Russian University Consortium for the Development of Software-Defined Network Technology with Orenburg State University as its member.

Centralized decision on the organization of data center heterogeneous infrastructure proposed in the papers has some drawbacks including reliability support, cost of obtaining a complete and current network conditions, low scalability [15, 16]. We assume that the development of a fully decentralized solution is the best option [17]; however, in this case, there is a problem of interaction between the controllers of autonomous systems. We are going to address this issue within a framework of our research using SDX technology, which will be extended to exchange not only information about the network, but also distributed sections and condition of cloud services and applications.

The algorithms for routing data flows in a software-defined network in case of track selection published in scientific sources do not take into account the need to ensure the QoS parameters for the previously installed and routed data flows [18, 19]. We are going to do it within a framework of the developed methods of adaptive network communications routing.

The existing QoS algorithms to provide a software-defined network are also quite inefficient. The paper [17] describes an approach to dynamic routing of multimedia flows transmission that provide a guaranteed maximum delay via the LARAC algorithm (Lagrangian Relaxation Based Aggregated) [19]. However, the authors consider only the cases of single delays on each network connection and do not take into account the minimum guaranteed bandwidth. A similar approach is described in the paper [18]; the authors pose and solve the optimization problem for the transfer of multimedia traffic without losses on alternative routes, leaving the shortcuts for common data.

The researchers from Stanford have offered an algorithm for adaptive control of QoS Shortest Span First, which enables to calculate the optimal priorities for each flow

mathematically, to minimize crosstalk influence of flows on delay, to manage priorities dynamically depending on the current situation, and to lay the flow of data transmission through specific port queues [20].

We are going to formulate optimization problems for laying routes with QoS constraints and load balancing within a framework of adaptive routing methods of network communications cloud services and applications developed in this research. In their solution, we may use heuristics similar to the Shortest Span First algorithm. Besides, we will account for the distributed nature of a cloud platform.

The analysis of scientific sources on the topic of the study has shown that:

a) so far, there are no effective algorithmic solutions for planning virtual machines, cloud services, application-oriented accounting topology of the computer system, and communication tasks schemes;
b) the existing solutions for managing distributed scientific computing on multi-cloud platforms plan computing tasks without subsequent adjustment of network to their communication schemes and use traditional routing methods;
c) the existing methods of data flow routing can be enhanced by taking into account the QoS requirements and distributed nature of a heterogeneous cloud platform.

This demonstrates the novelty of the solutions offered by the project.

Thus, the development of new methods and algorithms to improve the efficiency of cloud computing with the use of heterogeneous cloud platforms is a crucial task.

Conclusion

This research has enabled to solve the issue of optimizing the distribution of cloud applications and services data flows in the virtual data center. The developed models have enabled to implement the adaptive algorithm of data flow routing for the effective use of data transmission channels located in the data center. The results show that the algorithm for optimizing the adaptive routing of data flow balancing, based on collected information about alternative routes, has enabled to reduce the response time for applications and services of a heterogeneous cloud platform with a dynamically changing load on channels by 40% compared to traditional networks, and by 25% compared to the model algorithms of the Protocol version 1.4 of OpenFlow. It became possible due to the identification of traffic routes at the initial stage by using the data of the application placement in the network of the virtual data center.

The experimental studies have found that the application of the developed algorithm allows improving the efficiency of user requests processing and reducing the number of failures.

We are going to assess our approach with a larger number of flows of applications, since it will allow us to assess the accuracy of the proposed solution. For this task we prepare software package which can simulate of the routing algorithms enables to identify applications and services in the virtual data center and create the optimal routes for data transmission.

The research has been supported by the Russian Foundation of Basic Research (grants 16-37-60086 mol_a_dk, 16-07-01004 a), and the President of the Russian Federation within the grant for state support of young Russian scientists (MK-1624.2017.9).

References

1. Bolodurina I., Parfenov D. Development and research of models of organization storages based on the software-defined infrastructure. *39th International Conference on Telecommunications and Signal Processing*, 2016, pp. 1-6. DOI: 10.1109/TSP.2016.7760818
2. Parfenov D., Bolodurina I., Shukhman A. Approach to the effective controlling cloud computing resources in data centers for providing multimedia services. *11th International Siberian Conference on Control and Communications*, 2015, pp. 1-6. DOI: 10.1109/SIBCON.2015.7147170
3. Singh D., Singh J., Chhabra A. High availability of clouds: failover strategies for cloud computing using integrated checkpointing algorithms. *International Conference on Communication Systems and Network Technologies*, 2012, pp. 698-703. DOI: 10.1109/CSNT.2012.155
4. Aida K., Kasahara H., Narita S. Job Scheduling Scheme for Pure Space Sharing among Rigid Jobs. *Lecture Notes in Computer Science*, 1998, Vol. 1459. pp. 98-121. DOI: 10.1007/bfb0053983
5. Garey M., Graham R. (1975) Bounds for multiprocessor scheduling with resource constraints. *SIAM Journal on Computing*, 1975, Vol. 4, Issue 2, pp. 187-200. DOI: 10.1137/0204015
6. Arndt O., Freisleben1 B., Kielmann T., Thilo F. A comparative study of online scheduling algorithms for networks of workstations. *Cluster Computing*, 2000, Vol. 4, Issue 2, pp. 95-112. DOI: 10.1023/A:1019024019093
7. Feitelson D., Weil A. Utilization and predictability in scheduling the IBM SP2 with backfilling. *Parallel Processing Symposium*, 1998, pp. 542-546. DOI: 10.1109/IPPS.1998.669970
8. Lawson B., Smirni E. Multiple-queue Backfilling Scheduling with Priorities and Reservations for Parallel Systems. *Lecture Notes in Computer Science*, 2002, Vol. 2537, pp. 40-47. DOI: 10.1007/3-540-36180-4_5
9. Perkovic D., Keleher P. Randomization, Speculation, and Adaptation in Batch Schedulers. *Supercomputing ACM/IEEE Conference*, 2000, pp. 7-18. DOI: 10.1109/SC.2000.10041
10. Srinivasan S., Kettimuthu R. Selective Reservation Strategies for Backfill Job Scheduling. *Lecture Notes in Computer Science*, 2002, Vol. 2357, pp. 55-71. DOI: 10.1007/3-540-36180-4_4
11. Jing L., Mingze L., Gang W., Xiaoguang L., Zhongwei L., Huijun T. Global reliability evaluation for cloud storage systems with proactive fault tolerance. *Lecture Notes in Computer Science*, 2015, Vol. 9531, pp. 189-203. DOI: 10.1007/978-3-319-27140-8_14
12. Rahme J., Xu H. Reliability-based software rejuvenation scheduling for cloud-based systems. *27th International Conference on Software Engineering and Knowledge Engineering*, 2015, pp. 1-6. DOI: 10.18293/seke2015-233
13. OFELIA: OpenFlow in Europe. Available at: http://www.fp7-ofelia.eu (accessed: 27.05.2016)
14. Kang J., Bannazadeh H., Rahimi H. Software-defined infrastructure and the future central office. *IEEE International Conference on Communications Workshops*, 2013, pp. 225-229. DOI: 10.1109/ICCW.2013.6649233

15. Mambretti J., Chen J., Yeh F. Software-Defined Network Exchanges (SDXs) and Infrastructure (SDI): Emerging innovations in SDN and SDI interdomain multi-layer services and capabilities. *First International Science and Technology Conference (Modern Networking Technologies)*, 2014, pp. 1-6. DOI: 10.1109/MoNeTeC.2014.6995590
16. Lin T., Kang J., Bannazadeh H. Enabling SDN Applications on Software-Defined Infrastructure. *Network Operations and Management Symposium*, 2014, pp. 1-7. DOI: 10.1109/NOMS.2014.6838226
17. Ibanez G., Naous J., Rojas E., Rivera D., Schuymer T. A Small Data Center Network of ARP-Path Bridges made of Openflow Switches. *36th IEEE Conference on Local Computer Networks*, 2011, pp. 15-23.
18. Shimonishi H., Ochiai H., Enomoto E., Iwata A. Building Hierarchical Switch Network Using OpenFlow. *International Conference on Intelligent Networking and Collaborative Systems*, 2009, pp. 391-394. DOI: 10.1109/INCOS.2009.66
19. Egilmez H. OpenQoS: An OpenFlow controller design for multimedia delivery with end-to-end quality of service over software-defined networks. *Signal & Information Processing Association Annual Summit and Conference (APSIPA ASC)*, 2012, pp. 1-6.
20. Kim W., Sharma P., Lee J., Banerjee S., Tourrilhes J., Lee S., Yalagandula P. Automated and Scalable QoS Control for Network Convergence. *Internet network management conference on Research on enterprise networking*, 2010, pp. 1-1.

УДК 004.75, 004.042 **DOI: 10.14529/cmse170206**

АЛГОРИТМЫ И МЕТОДЫ ДИНАМИЧЕСКОЙ МАРШРУТИЗАЦИИ ДЛЯ УПРАВЛЕНИЯ ПОТОКАМИ ТРАФИКА ОБЛАЧНЫХ ПРИЛОЖЕНИЙ И СЕРВИСОВ

© 2017 И.П. Болодурина[1], Д.И. Парфёнов[1]
[1]*Оренбургский государственный университет*
(460018 Оренбург, пр. Победы, д. 13)
E-mail: prmat@mail.osu.ru, fdot_it@mail.osu.ru
Поступила в редакцию: 01.05.2017

В настоящее время доля использования технологии облачных вычислений в современных бизнес процессах компаний неуклонно растет. Несмотря на то, что это позволяет снижать стоимость владения и эксплуатации ИТ инфраструктуры, существует ряд проблем, связанных с управлением центрами обработки данных. Одной из таких проблем является эффективность использования имеющихся в распоряжении компаний вычислительных и сетевых ресурсов. Одним из направлений оптимизации является процесс управления трафиком облачных приложений и сервисов в центрах обработки данных (ЦОД). Учитывая многозвенную архитектуру современных ЦОД, такая задача весьма нетривиальна. Преимуществом современной инфраструктуры виртуализации является возможность использования программно-конфигурируемых сетей и программно-управляемых хранилищ данных. Однако существующие алгоритмические решения при оптимизации не учитывают ряд особенностей формирования трафика в сети с несколькими классами приложений.

В рамках проведенного исследования решена задача оптимизации распределения трафика облачных приложений и сервисов для программно-управляемой инфраструктуры виртуального ЦОД. Предложена имитационная модель, позволяющая описать трафик в программно-конфигурируемых сегментах сети ЦОД, участвующих в обработки запросов пользователей к приложениям и сервисам расположенных сетевой среде, включающей в себя гетерогенную облачную платформу и программно-конфигурируемые хранилища данных.

Разработанная модель позволила реализовать алгоритм управления трафиком облачных приложений и оптимизировать доступ системе хранения, за счет эффективного использование канала для передачи данных.

В ходе экспериментальных исследований установлено что, применение разработанного алгоритма позволяет сократить время отклика облачных приложений и сервисов, и как следствие повысить производительность обработки запросов пользователей и снизить количество отказов.

Ключевые слова: программно-конфигурируемая сеть, центры обработки данных, облачные вычисления, ИТ-инфраструктура.

ОБРАЗЕЦ ЦИТИРОВАНИЯ

Bolodurina I.P., Parfenov D.I. Dynamic Routing Algorithms and Methods for Controlling traffic Flows of Cloud Applications and Services // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2017. Т. 6, № 2. С. 84–98. DOI: 10.14529/cmse170206.

Литература

1. Bolodurina I., Parfenov D. Development and research of models of organization storages based on the software-defined infrastructure // 39th International Conference on Telecommunications and Signal Processing, 2016, P. 1-6. DOI: 10.1109/TSP.2016.7760818
2. Parfenov D., Bolodurina I., Shukhman A. Approach to the effective controlling cloud computing resources in data centers for providing multimedia services // 11th International Siberian Conference on Control and Communications, 2015, P. 1-6. DOI: 10.1109/SIBCON.2015.7147170
3. Singh D., Singh J., Chhabra A. High availability of clouds: failover strategies for cloud computing using integrated checkpointing algorithms // International Conference on Communication Systems and Network Technologies, 2012, P. 698-703. DOI: 10.1109/CSNT.2012.155
4. Aida K., Kasahara H., Narita S. Job Scheduling Scheme for Pure Space Sharing among Rigid Jobs // Lecture Notes in Computer Science, 1998, Vol. 1459. P. 98-121. DOI: 10.1007/bfb0053983
5. Garey M., Graham R. (1975) Bounds for multiprocessor scheduling with resource constraints // SIAM Journal on Computing, 1975, Vol. 4, Issue 2, P. 187-200. DOI: 10.1137/0204015
6. Arndt O., Freisleben1 B., Kielmann T., Thilo F. A comparative study of online scheduling algorithms for networks of workstations // Cluster Computing, 2000, Vol. 4, Issue 2, P. 95-112. DOI: 10.1023/A:1019024019093
7. Feitelson D., Weil A. Utilization and predictability in scheduling the IBM SP2 with backfilling // Parallel Processing Symposium, 1998, P. 542-546. DOI: 10.1109/IPPS.1998.669970
8. Lawson B., Smirni E. Multiple-queue Backfilling Scheduling with Priorities and Reservations for Parallel Systems // Lecture Notes in Computer Science, 2002, Vol. 2537, P. 40-47. DOI: 10.1007/3-540-36180-4_5
9. Perkovic D., Keleher P. Randomization, Speculation, and Adaptation in Batch Schedulers // Supercomputing ACM/IEEE Conference, 2000, P. 7-18. DOI: 10.1109/SC.2000.10041
10. Srinivasan S., Kettimuthu R. Selective Reservation Strategies for Backfill Job Scheduling // Lecture Notes in Computer Science, 2002, Vol. 2357, P. 55-71. DOI: 10.1007/3-540-36180-4_4

11. Jing L., Mingze L., Gang W., Xiaoguang L., Zhongwei L., Huijun T. Global reliability evaluation for cloud storage systems with proactive fault tolerance // Lecture Notes in Computer Science, 2015, Vol. 9531, P. 189-203. DOI: 10.1007/978-3-319-27140-8_14
12. Rahme J., Xu H. Reliability-based software rejuvenation scheduling for cloud-based systems // 27th International Conference on Software Engineering and Knowledge Engineering, 2015, P. 1-6. DOI: 10.18293/seke2015-233
13. OFELIA: OpenFlow in Europe. URL: http://www.fp7-ofelia.eu (дата обращения: 27.05.2016)
14. Kang J., Bannazadeh H., Rahimi H. Software-defined infrastructure and the future central office // IEEE International Conference on Communications Workshops, 2013, P. 225-229. DOI: 10.1109/ICCW.2013.6649233
15. Mambretti J., Chen J., Yeh F. Software-Defined Network Exchanges (SDXs) and Infrastructure (SDI): Emerging innovations in SDN and SDI interdomain multi-layer services and capabilities // First International Science and Technology Conference (Modern Networking Technologies), 2014, P. 1-6. DOI: 10.1109/MoNeTeC.2014.6995590
16. Lin T., Kang J., Bannazadeh H. Enabling SDN Applications on Software-Defined Infrastructure // Network Operations and Management Symposium, 2014, P. 1-7. DOI: 10.1109/NOMS.2014.6838226
17. Ibanez G., Naous J., Rojas E., Rivera D., Schuymer T. A Small Data Center Network of ARP-Path Bridges made of Openflow Switches // 36th IEEE Conference on Local Computer Networks, 2011, P. 15-23.
18. Shimonishi H., Ochiai H., Enomoto E., Iwata A. Building Hierarchical Switch Network Using OpenFlow // International Conference on Intelligent Networking and Collaborative Systems, 2009, P. 391-394. DOI: 10.1109/INCOS.2009.66
19. Egilmez H. OpenQoS: An OpenFlow controller design for multimedia delivery with end-to-end quality of service over software-defined networks // Signal & Information Processing Association Annual Summit and Conference (APSIPA ASC), 2012, P. 1-6.
20. Kim W., Sharma P., Lee J., Banerjee S., Tourrilhes J., Lee S., Yalagandula P. Automated and Scalable QoS Control for Network Convergence // Internet network management conference on Research on enterprise networking, 2010, P. 1-1.

Болодурина Ирина Павловна, д.т.н., профессор, зав. каф., кафедра прикладной математики, Оренбургский государственный университет (Оренбург, Российская Федерация)

Парфёнов Денис Игоревич, к.т.н., доцент, кафедра прикладной математики, Оренбургский государственный университет (Оренбург, Российская Федерация)

СВЕДЕНИЯ ОБ ИЗДАНИИ

Научный журнал «Вестник ЮУрГУ. Серия «Вычислительная математика и информатика» основан в 2012 году.

Свидетельство о регистрации ПИ ФС77-57377 выдано 24 марта 2014 г. Федеральной службой по надзору в сфере связи, информационных технологий и массовых коммуникаций.

Журнал включен в Реферативный журнал и Базы данных ВИНИТИ; индексируется в библиографической базе данных РИНЦ. Журнал размещен в открытом доступе на Всероссийском математическом портале MathNet. Сведения о журнале ежегодно публикуются в международной справочной системе по периодическим и продолжающимся изданиям «Ulrich's Periodicals Directory».

Решением Президиума Высшей аттестационной комиссии Министерства образования и науки Российской Федерации журнал включен в «Перечень рецензируемых научных изданий, в которых должны быть опубликованы основные научные результаты на соискание ученой степени кандидата наук, на соискание ученой степени доктора наук» по следующим отраслям и группам специальностей: 05.13.00 – информатика, вычислительная техника и управление; 01.01.00 – математика; 25.00.00 – науки о Земле (№421).

Подписной индекс научного журнала «Вестник ЮУрГУ», серия «Вычислительная математика и информатика»: 10244, каталог «Пресса России». Периодичность выхода — 4 выпуска в год (февраль, май, август и ноябрь).

ПРАВИЛА ДЛЯ АВТОРОВ

1. Правила подготовки рукописей и пример оформления статей можно загрузить с сайта серии `http://vestnikvmi.susu.ru`. **Статьи, оформленные без соблюдения правил, к рассмотрению не принимаются.**
2. Адрес редакции научного журнала «Вестник ЮУрГУ», серия «Вычислительная математика и информатика»:

   Россия 454080, г. Челябинск, пр. им. В.И. Ленина, 76, ЮУрГУ, кафедра СП,

   ответственному секретарю Цымблеру М.Л.
3. Адрес электронной почты редакции: `vestnikvmi@susu.ru`
4. **Плата с авторов за публикацию рукописей не взимается, и гонорары авторам не выплачиваются.**