ИЗВЕСТИЯ РАН. ТЕОРИЯ И СИСТЕМЫ УПРАВЛЕНИЯ, 2014, № 6, с. 80–93

КОМПЬЮТЕРНЫЕ МЕТОДЫ

УДК 004.031.43

АЛГОРИТМ РАСПРЕДЕЛЕНИЯ РЕСУРСОВ В ЦЕНТРАХ ОБРАБОТКИ ДАННЫХ С РАЗДЕЛЬНЫМИ ПЛАНИРОВЩИКАМИ ДЛЯ РАЗЛИЧНЫХ ТИПОВ РЕСУРСОВ

© 2014 г. П. М. Вдовин, В. А. Костенко

Москва, МГУ

Поступила в редакцию 03.04.14 г., после доработки 14.05.14 г.

Рассматривается алгоритм отображения запросов на физические ресурсы для центров обработки данных с раздельными планировщиками для различных типов ресурсов: вычислительных, сетевых и систем хранения данных. Алгоритм основан на сочетании жадных стратегий и стратегий ограниченного перебора. Алгоритм позволяет выбирать требуемый баланс между вычислительной сложностью и качеством получаемых отображений путем ограничения максимально допустимой глубины перебора. Приводятся теоретические и экспериментальные результаты исследования его свойств.

DOI: 10.7868/S000233881405014X

Введение. В [1, 2] разработана математическая модель центра обработки данных (ЦОД), которая позволяет описывать широкий класс архитектур ЦОД, и сформулирована математическая постановка задачи распределения ресурсов в режиме использования Infrastructure-as-a-Service (IaaS) [3]. В предложенной модели вычислительные ресурсы, системы хранения данных и сетевые ресурсы рассматриваются как планируемые типы ресурсов, и их планирование происходит согласованно в смысле соблюдения соглашений о качестве сервиса (service-level agreement – SLA) [4]. Это позволяет задавать SLA для всех типов ресурсов ЦОД и строить такие отображения запросов на физические ресурсы, для которых возможно гарантированное выполнение запрашиваемых SLA.

В [2] проведено сравнение различных подходов к распределению ресурсов в ЦОД. В данной работе рассматривается алгоритм отображения запросов на физические ресурсы для ЦОД с раздельными планировщиками для различных типов ресурсов и приводятся результаты исследования его свойств.

Наиболее близкими задачами с точки зрения гарантированного выполнения SLA являются задачи построения расписаний для систем реального времени [5–10]. В этих задачах в качестве критерия SLA задаются ограничения на допустимое время выполнения прикладных программ или процессов программы, которые нарушаться не должны. В рассматриваемой задаче в качестве критериев SLA устанавливаются требования к “объемам” выделяемых элементам запросов физических ресурсов. Это делает проблематичным использование алгоритмов построения расписаний для данной задачи.

1. Математическая постановка задачи распределения ресурсов ЦОД. *Модель физических ресурсов ЦОД* будем задавать графом $H = (P \cup M \cup K, L)$, где P – множество вычислительных узлов, M – множество хранилищ данных, K – множество коммутационных элементов сети обмена ЦОД, L – множество физических каналов передачи данных. На множествах P, M, K и L определены векторные функции, аргументами которых являются соответственно элементы множеств P, M, K или L. Значениями функций выступают характеристики соответствующего вычислительного узла, хранилища данных, коммутационного элемента или канала передачи данных:

$$(ph_1, ph_2, \ldots, ph_{n1}) = fph(p), \quad p \in P,$$

$$(mh_1, mh_2, \ldots, mh_{n2}) = fmh(m), \quad m \in M,$$

$$(kh_1, kh_2, \ldots, kh_{n3}) = fkh(k), \quad k \in K,$$

$$(lh_1, lh_2, \ldots, lh_{n4}) = flh(l), \quad l \in L.$$

В данной работе предполагается, что коммутационные элементы и каналы передачи данных имеют одинаковые характеристики. На практике наиболее часто используются следующие характеристики физических элементов:

1) для вычислительных узлов: ⟨число ядер⟩, ⟨частота⟩, ⟨тип ядра⟩, ⟨объем оперативной памяти⟩, ⟨объем дисковой памяти⟩;

2) для хранилищ данных: ⟨объем памяти⟩, ⟨тип хранилища данных⟩;

3) для коммутационных элементов: ⟨пропускная способность⟩, ⟨задержка⟩;

4) для каналов передачи данных: ⟨пропускная способность⟩, ⟨задержка⟩.

Ресурсный запрос будем задавать графом $G = (W \cup S, E)$, где W – множество виртуальных машин, используемых приложениями, S – множество виртуальных хранилищ данных (storage-элементов), E – множество виртуальных каналов передачи данных между виртуальными машинами и storage-элементами запроса. На множествах W, S и E определены векторные функции, аргументы которых – соответствующие элементы множеств W, S или E. Значениями функций являются характеристики (требуемое качество сервиса (SLA)) соответствующей виртуальной машины, storage-элемента или виртуального канала:

$$(wg_1, wg_2, \ldots, wg_{n1}) = fwg(w), \quad w \in W,$$

$$(sg_1, sg_2, \ldots, sg_{n2}) = fsg(s), \quad s \in S,$$

$$(eg_1, eg_2, \ldots, eg_{n4}) = feg(e), \quad e \in E.$$

Характеристики SLA элемента запроса совпадают с характеристиками соответствующего ему физического ресурса.

Назначением ресурсного запроса будем называть отображение

$$A : G \to H = \{W \to P, S \to M, E \to \{K, L\}\}.$$

Выделим три типа отношений между характеристиками запросов и соответствующими характеристиками физических ресурсов. Обозначим через x_i i-ю характеристику запроса и через y_j – соответствующую ей характеристику физического ресурса j. Тогда эти отношения можно записать следующим образом.

1. Недопустимость перегрузки емкости физического ресурса:

$$\sum_{i \in R_j} x_i \le y_j,$$

здесь R_j – множество запросов, назначенных на выполнение на физическом ресурсе j, x_i – соответствующая характеристика назначенного запроса.

2. Соответствие типа запрашиваемого ресурса типу физического ресурса:

$$x_i = y_j.$$

3. Наличие требуемых характеристик у физического ресурса:

$$x_i \le y_j.$$

Отображение $A : G \to H = \{W \to P, S \to M, E \to \{K, L\}\}$ будем называть корректным, если для всех физических ресурсов и всех их характеристик выполняются отношения 1–3.

Остаточным графом доступных ресурсов называется граф H_{res}, для которого переопределены значения функций по характеристикам, которые должны удовлетворять отношению 1:

$$fph_{res}(p) = fph(p) - \sum_{w \in W_p} fwg(w),$$

$$fmh_{res}(m) = fmh(m) - \sum_{s \in S_m} fsg(s), \tag{1.1}$$

$$fkh_{res}(k) = fkh(k) - \sum_{e \in E_l} feg(e), \quad flh_{res}(l) = flh(l) - \sum_{e \in E_k} feg(e).$$

Здесь W_p – множество виртуальных машин, назначенных на выполнение на вычислительном узле p, E_l – множество виртуальных каналов, отображенных на физический канал l, E_k – множество

виртуальных каналов, проходящих через коммутационный элемент k, S_m – множество storage-элементов, размещенных в хранилище данных m.

В качестве *исходных данных* задачи назначения ресурсных запросов на физические ресурсы заданы:

1) множество запросов $Z = \{G_i\}$, поступивших планировщику;

2) остаточный граф доступных ресурсов $H_{res} = (P \cup M \cup K, L)$.

Требуется: из множества Z разместить на выполнение в ЦОД максимальное число запросов, таких, что отображение A является корректным.

В случае, когда невозможно назначить виртуальное хранилище данных, возможен вызов процедуры репликации, позволяющей дублировать данные на нескольких физических хранилищах данных. Данная проблема возникает, когда имеются виртуальные хранилища данных с высокой интенсивностью считывания и низкой интенсивностью записи. В этом случае для обеспечения консистентности данных не требуется канал с высокой пропускной способностью, и половина приложений может работать с виртуальным хранилищем данных, а другая – с его копией, которая располагается в другом физическом хранилище данных.

Репликацией называется отображение $R : H \to H$, дублирующее данные некоторого $m \in M$ и создающее виртуальный канал поддержки консистентности данных (m', l_1, k_1, …, k_{n-1}, l_n, m); $k_i \in K, l_i \in L, m' \in M$, m' – реплика хранилища m.

Если storage-элемент является репликацией некоторого виртуального хранилища данных s, то в граф запрашиваемых ресурсов G добавляется вершина s'. В граф G также добавляется виртуальный канал между вершинами s и s', пропускная способность которого определяется исходя из требования обеспечения консистентности репликации и базы данных.

На вход алгоритму назначения запросов на физические ресурсы подаются остаточный граф доступных ресурсов H_{res} и множество ресурсных запросов $\{G_i\}$. Множество $\{G_i\}$ формирует диспетчер задач. В него, кроме вновь поступивших запросов, могут входить и запросы, которые выполняются и для которых допустима миграция. Диспетчер задач также определяет время запуска планировщика.

Выходом алгоритма назначения запросов на физические ресурсы является множество назначений ресурсных запросов на физические ресурсы $\{A_i : G_i \to H, i = \overline{1,n}\}$ *и множество репликаций* $\{R_i\}, i = 0,1,\ldots$

2. Алгоритм назначения запросов на физические ресурсы ЦОД. Алгоритм назначения запросов на физические ресурсы с раздельными планировщиками состоит из трех шагов.

Ш а г 1. Назначение виртуальных машин на вычислительные узлы.

Ш а г 2. Назначение storage-элементов на физические хранилища данных.

Ш а г 3. Для полученных отображений виртуальных машин и storage-элементов на физические ресурсы построение отображения виртуальных каналов запросов на физические каналы передачи данных и коммутационные элементы.

2.1. Назначение виртуальных машин и storage-элементов. В теории расписаний задачи назначения виртуальных машин и storage-элементов на физические ресурсы известны как задачи упаковки предметов в контейнеры [11]. Предлагаемый алгоритм основан на сочетании жадных стратегий и стратегий ограниченного перебора. Алгоритм осуществляет назначение элементов запросов по жадной схеме и в случае невозможности назначения очередного элемента вызывает процедуру ограниченного перебора. Для данной процедуры задается параметр, ограничивающий глубину перебора. Это позволяет выбирать требуемый баланс между вычислительной сложностью и точностью алгоритма.

Общая схема процедуры назначения элементов запроса (виртуальных машин или storage-элементов) выглядит следующим образом.

Ш а г 1. Из множества ресурсных запросов $\{G_i\}$ выбрать очередной запрос G_i в соответствии с жадным критерием K_G.

Ш а г 2. Выбрать очередной элемент e (виртуальную машину $e \in W, W \in G_i$ или storage-элемент $e \in S, S \in G_i$) в соответствии с жадным критерием K_e.

Ш а г 3. Сформировать множество физических ресурсов Ph ($Ph \subseteq P$ или $Ph \subseteq M$ соответственно), на которые может быть назначен выбранный элемент e, т.е. для которого выполнены отношения корректности отображения в случае назначения запроса e на физический ресурс.

Ш а г 3.1. Если множество Ph пусто, то вызвать процедуру ограниченного перебора. Если процедура возвращает неуспех, то запрос G_i не может быть назначен: удалить ранее назначенные

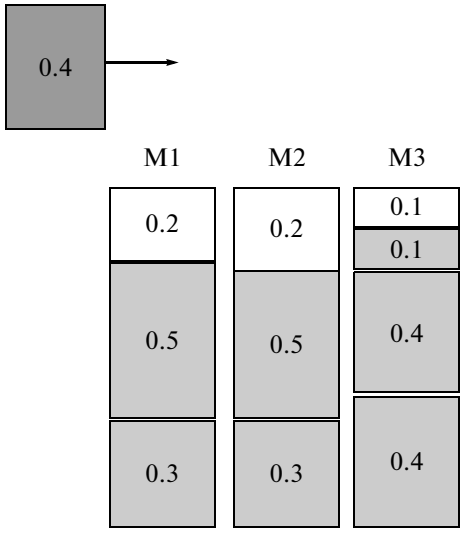

Рис. 1. Пример выполнения процедуры ограниченного перебора

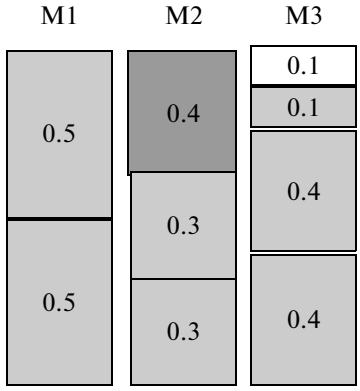

Рис. 2. Переназначение элементов из рис. 1 для назначения элемента

элементы запроса G_i и переопределить значения характеристик физических ресурсов в соответствии с функциями (1.1). Если множество $\{G_i\}$ не пусто, перейти на шаг 1; в противном случае завершить работу алгоритма.

Ш а г 3.2. Если множество Ph не пусто, то выбрать физический элемент $ph \in Ph$ для назначения в соответствии с жадным критерием K_{ph} и назначить элемент запроса e на физический элемент ph, и переопределить значения его характеристик в соответствии с функциями (1.1). Перейти на шаг 2, если есть нерассмотренные элементы запроса. Иначе перейти на шаг 1, если множество $\{G_i\}$ не пусто; в противном случае завершить работу алгоритма.

2.1.1. П р о ц е д у р а о г р а н и ч е н н о г о п е р е б о р а. Процедура ограниченного перебора вызывается в случае, когда очередной элемент запроса e ($e \in W$ или $e \in S$) не может быть назначен ни на один физический ресурс ph ($ph \in P$ или $ph \in M$ соответственно). Принцип работы процедуры ограниченного перебора показан на рис. 1, 2. На рис. 1 изображена ситуация, в которой виртуальный элемент с емкостью 0.4 не может быть назначен ни на один из физических элементов, несмотря на то, что суммарной свободной емкости достаточно для его назначения. При переназначении элементов таким образом, как изображено на рис. 2, фрагментация устраняется и виртуальный элемент может быть назначен.

Ограниченный перебор ограничивается заданным числом m, которое указывает максимальное количество физических элементов, среди которых можно производить переназначение (т.е.

при $m = 2$ назначенные элементы могут сниматься и переназначаться одновременно не более чем с двух физических элементов).

Общая схема процедуры ограниченного перебора, таким образом, следующая.

Ш а г 1. Для заданного m сформировать все множества $A_m \subseteq A$ из не более чем m физических ресурсов и перебирать в соответствии с уменьшением суммарного значения, вычисленного согласно жадному критерию K_G.

Ш а г 1.1. Если для некоторого множества A_m суммарная остаточная емкость ресурсов достаточна для назначения текущего элемента, то выполнить шаги 1.1.1–1.1.3. Иначе перейти на шаг 2.

Ш а г 1.1.1. Произвести снятие элементов, которые назначены на физические ресурсы из множества A_m.

Ш а г 1.1.2. Произвести попытку переназначения одним из указанных способов:

а) назначить сначала элемент e, согласно жадным стратегиям, на один из физических ресурсов множества A_m, после чего назначить все виртуальные элементы, снятые с физических элементов множества A_m, используя жадные стратегии;

б) сформировать множество, состоящее из элемента e и из всех виртуальных элементов, снятых с физических элементов множества A_m; произвести назначение сформированного множества согласно жадным стратегиям (см. шаги 2 и 3 общей схемы назначения элементов запроса без процедуры ограниченного перебора);

в) сформировать множество, состоящее из элемента e и из всех виртуальных элементов, снятых с физических элементов множества A_m; произвести полный перебор возможных назначений элементов из сформированного множества.

Ш а г 1.1.3. В случае неуспешного переназначения вернуть исходные назначения элементов.

Ш а г 2. Если после перебора всех указанных множеств элемент e не может быть назначен, вернуть неуспех.

2.1.2. Ж а д н ы е к р и т е р и и. Качество работы алгоритма существенно зависит от используемых жадных критериев. Критерии задаются для шага 1 общей схемы процедуры назначения элементов запроса, когда выбирается запрос для назначения (K_G), на шаге 2 – когда выбирается элемент запроса (K_e), и на шаге 3.2 – когда выбирается физический ресурс, на который назначается элемент (K_{ph}).

Предложенные критерии основаны на функции стоимости $r(e)$ назначения элемента e. В работе предлагается вычислять стоимость назначения элемента двумя способами:

1) динамический выбор самой загруженной в момент назначения характеристики ресурса и вычисление стоимости по выбранной характеристике;

2) вычисление взвешенной суммы характеристик ресурса согласно их загруженности.

Пусть элемент характеризуется вектором значений требуемых характеристик ресурсов ($r_{e,1}$, $r_{e,2}$, ... $r_{e,n}$), для которых недопустима перегрузка[1]. При динамическом выборе перед каждым назначением определяется самая загруженная характеристика:

$$max\_resource = \arg\max_i \left(\frac{\sum\limits_{G_a} \sum\limits_{e \in E,\, E \in G_a} r_{e,i}}{\sum\limits_{ph \in Ph} r_{ph,i}} \right).$$

В данной формуле суммирование в числителе производится по всем назначенным запросам G_a. Загрузка ресурса, таким образом, вычисляется как частное между суммарной емкостью назначенных элементов и общей емкостью физических элементов по данной характеристике.

Функция стоимости элемента при динамическом назначении рассчитывается как емкость самого загруженной характеристики: $r(e) = r_{e,max\_resource}$.

Для определения взвешенной стоимости элемента для каждого ресурса вычисляется дефицит:

[1]В данном случае рассматриваются только характеристики, для которых проверяется первое отношение между характеристиками запросов и соответствующими характеристиками физических ресурсов, т.е. суммарная емкость виртуальных элементов, назначенных на ресурс, не должна превосходить емкость физического ресурса; фильтрация элементов по двум другим отношениям уже произведена к моменту использования критериев.

$$d(i) = \frac{\sum\limits_{G} \sum\limits_{e \in E,\, E \in G} r_{e,i}}{\sum\limits_{ph \in Ph} r_{ph,i}}.$$

Данная величина определяется как частное, в котором числителем является общая требуемая емкость по данной характеристике для всех запросов G, знаменатель – общее значение имеющихся физических ресурсов. Чем больше дефицит, тем больше требуется ресурсов данного типа.

Функция стоимости вычисляется следующим образом[2]:

$$r(e) = \sum_{i=1..n} d(i)\, r_{e,i}$$

и равна взвешенной сумме требуемых ресурсов с учетом их дефицита.

Выделяется два варианта критериев: компактное назначение и сбалансированное назначение.

Компактное назначение используется, когда физические ресурсы сети не являются критическим ресурсом, т.е. в случаях, когда отношение стоимостей требуемых сетевых ресурсов к имеющимся сетевым ресурсам меньше заданного порога. В случае компактного назначения предлагаются следующие критерии:

1) K_G: выбирается запрос G с наибольшей суммарной требуемой стоимостью элементов;

2) K_e: среди элементов запроса G выбирается тот элемент e, который имеет наибольшую требуемую стоимость $r_v(e)$;

3) K_{ph}: среди множества физических элементов $ph \in Ph$, которые имеют достаточно ресурсов для назначения выбранного элемента e, выбирается тот, который имеет наименьшую стоимость (вычисляется аналогично стоимости виртуального элемента).

Описанные жадные критерии позволяют производить назначение по стратегии BFD (best-fit decreased, [11]). В случае одномерной упаковки данная стратегия достигает асимптотической точности назначения, равной 11/9. Это означает, что в наихудшем случае данный алгоритм назначает элементы в количество физических ресурсов, в 11/9 раз большее, чем оптимальный алгоритм.

Сбалансированное назначение предлагается использовать в случае, когда сетевые ресурсы являются критическими, т.е. когда отношение стоимости требуемых сетевых ресурсов к стоимости имеющихся сетевых ресурсов превосходит некоторый порог.

При сбалансированном назначении первые два критерия (K_G и K_e) совпадают с компактным назначением. Для критерия K_{ph} предлагается среди множества элементов $ph \in Ph$, имеющих достаточно емкости для назначения выбранного элемента e, выбрать тот, который имеет *наибольшую* стоимость. Данный подход распределяет виртуальные ресурсы среди всех имеющихся физических элементов, что предоставляет больше физических ресурсов сети при назначении виртуальных каналов.

2.2. Назначение виртуальных каналов на физические ресурсы сети. Назначение виртуальных каналов производится после того, как получено отображение виртуальных машин и storage-элементов, и основано на решении задачи нахождения k кратчайших путей в графе [12]. В случае неудачи назначения некоторого виртуального канала $e \in E$ вызывается процедура ограниченного перебора, которая заключается в переборе множеств ограниченной мощности из уже назначенных виртуальных каналов с целью их переназначения таким образом, чтобы данный виртуальный канал мог быть назначен. Общая схема процедуры ограниченного перебора виртуальных каналов схожа со схемой ограниченного перебора виртуальных машин и storage-элементов. В случае неудачи вызывается процедура репликации, которая создает реплику для storage-элемента с целью устранения перегрузки входящих в физическое хранилище данных каналов обмена. При создании реплики необходимо также проложить канал для поддержания консистентности между storage-элементом и его репликой.

Следует отметить, что процедура репликации в данном алгоритме используется с целью оптимизации назначения виртуальных каналов, т.е. для создания дополнительных возможных путей при назначении запросов.

[2] В данной формуле значение емкости ресурсов считается нормированным.

2.2.1. Общая схема назначения виртуальных каналов. Общая схема алгоритма назначения виртуальных каналов следующая.

Шаг 1. Выбрать запрос $G \in G_A$ для назначения виртуальных каналов, согласно заданному критерию K_G, где G_A – множество запросов, для которых удалось произвести назначение виртуальных машин и storage-элементов.

Шаг 2. Выбрать виртуальный канал $e \in E$ запроса G, согласно заданному критерию K_e.

Шаг 3. Произвести назначение виртуального канала e с помощью алгоритма нахождения k кратчайших путей и переопределить значения характеристик физических ресурсов в соответствии с функциями (1.1).

Шаг 4. В случае неудачи назначения произвести процедуру ограниченного перебора.

Шаг 5. В случае неудачи процедуры ограниченного перебора выполнить процедуру репликации.

Шаг 6. В случае неудачи запрос G не может быть назначен, снять все назначенные виртуальные каналы, виртуальные машины и storage-элементы запроса G и переопределить значения характеристик физических ресурсов в соответствии с функциями (1.1).

2.2.2. Жадные критерии. На первых двух шагах предлагается выбирать критерии, аналогичные выбранным при назначении виртуальных машин и storage-элементов. Таким образом, предлагается выбирать запрос с наибольшей суммарной стоимостью виртуальных каналов (K_G), после чего последовательно выбирать виртуальный канал с наибольшей стоимостью (K_e). Стоимость определяется аналогично стоимости для вычислительных ресурсов/виртуальных машин и хранилищ данных/storage-элементов. Обычно для каналов передачи данных и коммутаторов используется только одна характеристика: пропускная способность. Стоимость в данном случае вычисляется как значение емкости данной характеристики.

2.2.3. Назначение одного виртуального канала. Назначение одного виртуального канала производится с использованием алгоритма поиска k кратчайших путей (например, алгоритма Йена [12]). Общая схема алгоритма следующая.

Шаг 1. Для заданного k произвести поиск не более чем k кратчайших путей $p \in P$ минимальной длины (под длиной понимается количество коммутаторов $k \in K$, входящих в путь $p \in P$). Производится поиск только тех путей, каждый элемент которых имеет достаточно емкости для назначения данного виртуального канала. Если ни одного пути не найдено, то возвращается неудача.

Шаг 2. Среди найденных путей выбрать путь $p \in P$ с наибольшей остаточной суммарной стоимостью среди каналов передачи данных и коммутаторов.

Описанная процедура позволяет производить поиск с использованием двух критериев: минимальная длина и максимальная остаточная стоимость. Минимальная длина является первичным признаком, так как топологии ЦОД (таких, как fattree [2]) имеют древовидную структуру (с добавлением избыточных ребер), что позволяет выбирать из множества путей одинаковой длины для каждой пары элементов.

2.2.4. Процедура ограниченного перебора. Процедура ограниченного перебора заключается в выполнении следующих шагов.

Шаг 1. Для заданного m перебирать все множества $A_e \subseteq A$ из не более чем m назначенных виртуальных каналов в порядке убывания суммарной стоимости виртуальных каналов. Для каждого множества A_e выполнить шаги 1.1–1.3.

Шаг 1.1. Произвести снятие выбранных виртуальных каналов $a \in A_e$.

Шаг 1.2. Произвести попытку назначения виртуального канала e с помощью алгоритма k кратчайших путей; если назначение не удается, то вернуть назначение снятых виртуальных каналов $a \in A_e$ и перейти к следующему множеству $A_e \subseteq A$.

Шаг 1.3. Произвести попытку назначения всех снятых виртуальных каналов $a \in A_e$; в случае успешного назначения вернуть успех, иначе снять виртуальный канал e и вернуть исходные назначения.

Шаг 2. Если после перебора всех указанных множеств $A_e \subseteq A$ виртуальный канал e не может быть назначен, вернуть неуспех.

2.2.5. Процедура репликации. Процедура репликации возможна в случае, если одним из элементов, которые соединяет виртуальный канал $e = (w, s)$, является storage-элемент $s \in S$, для которого доступна репликация. Процедура заключается в поиске хранилища данных $m \in M$,

которое имеет достаточно ресурсов для назначения реплики (реплика требует столько же ресурсов, сколько и данный storage-элемент s), и суммарная длина путей всех виртуальных каналов $e = (w \in W, s)$ (включающих данный storage-элемент s) к данной реплике m минимальна. Кроме того, необходимо провести канал l для поддержания консистентности между репликой и исходным storage-элементом (требуемая пропускная способность канала l определяется типом storage-элемента s). Если проложить канал l для поддержания консистентности не удается, то рассматривается другое хранилище s данных в порядке убывания суммарной длины путей виртуальных каналов.

При успешном выполнении процедуры репликации последующее назначение виртуальных каналов, включающих данный storage-элемент s, можно производить на данную реплику m. Если не удается найти хранилище данных $m \in M$ с достаточным количеством ресурсов либо не удается провести канал поддержания консистентности данных l, то процедура возвращает неуспех.

3. Вычислительная сложность алгоритма. Пусть N – число вычислительных узлов, S – число хранилищ данных, W – число коммутаторов в сети передачи данных. Пусть RV – общее число виртуальных машин всех ресурсных запросов, RS – общее число storage-элементов, RVL – общее число виртуальных каналов. Обозначим через K_{nodes} коэффициент перебора для операции ограниченного перебора виртуальных машин и storage-элементов, K_{links} – коэффициент перебора в операции назначения виртуальных каналов, K_{path} – количество исследуемых кратчайших путей в процедуре нахождения k-кратчайших путей.

1. Назначение виртуальных машин на вычислительные узлы, на данном этапе выполняются следующие действия.

1.1. Сортировка виртуальных машин согласно выбранному критерию; сложность данного шага $O(RV \log(RV))$.

1.2. Для каждой из виртуальных машин выбор вычислительного узла, на который производится назначение (если назначение возможно); выполняется в наихудшем случае за $O(RV\ N)$ операций.

1.3. Процедура ограниченного перебора. Количество перебираемых сочетаний вычисляется как $C_N^{K_{nodes}}$ (число сочетаний из N по K_{nodes}). На каждом сочетании происходит попытка переназначения, сложность которого зависит от выбранного способа ограниченного перебора:

а) при использовании жадного алгоритма в процедуре ограниченного перебора сложность переназначения составляет $O(RV(\log(RV) + N))$;

б) при использовании алгоритма полного перебора в процедуре ограниченного перебора сложность в наихудшем случае достигает $O(RV!\,N)$.

Сложность в наихудшем случае на данном этапе составляет $O(RV(\log(RV) + N) + O(C_{RV}^{K_{nodes}} RV(\log(RV) + N))$ при использовании жадного алгоритма в процедуре ограниченного перебора и $O(RV(\log(RV) + N) + O(C_{RV}^{K_{nodes}} RV!\,N))$ – в процедуре ограниченного перебора алгоритма с полным перебором всех возможных отображений.

2. Назначение storage-элементов на хранилища данных. Данный этап выполняется аналогично предыдущему и имеет такую же сложность, при этом вместо количества виртуальных машин в оценках используется количество storage-элементов.

3. Назначение виртуальных каналов. Данный этап заключается в следующей последовательности шагов.

3.1 Сортировка виртуальных каналов по выбранному критерию. Сложность данного шага $O(RVL \log(RVL))$.

3.2. Попытка назначения данного виртуального канала с помощью процедуры построения k-кратчайших путей. Данная процедура состоит в использовании алгоритма Дейкстры не более $K_{path}(W + 2)$ раз и содержит не более $O(K_{path}(W + 2)^2)$ операций (так как виртуальные машины и storage-элементы уже назначены, количество вершин в графе поиска равно $W + 2$). Данный шаг, таким образом, оценивается сложностью $O(RVL\, K_{path}(W + 2)^3)$.

3.3. Процедура ограниченного перебора глубины K_{links} для виртуальных каналов. Данная операция оценивается аналогично назначению виртуальных машин (отличие состоит в использовании процедуры k-кратчайших путей при переназначении). Сложность данного шага, таким об-

Таблица 1. Изменение сложности алгоритма в зависимости от глубины перебора K_{nodes} при использовании жадного алгоритма в процедуре ограниченного перебора

Глубина перебора K_{nodes}	Коэффициенты увеличения сложности алгоритма в зависимости от	
	N	S
1	N	S
2	$N(N-1)/2$	$S(S-1)/2$
3	$N(N-1)(N-2)/6$	$S(S-1)(S-2)/6$
...	...	...
K_{nodes}	$C_N^{K_{nodes}}$	$C_S^{K_{nodes}}$

Таблица 2. Изменение сложности алгоритма в зависимости от глубины перебора K_{nodes} при использовании полного перебора в процедуре ограниченного перебора

Глубина перебора K_{nodes}	Коэффициенты увеличения сложности алгоритма в зависимости от	
	N и RV	S и RS
1	$(RV-1)!N$	$(RS-1)!S$
2	$(RV-1)!N(N-1)/2$	$(RS-1)!S(S-1)/2$
3	$(RV-1)!N(N-1)(N-2)/2$	$(RS-1)!S(S-1)(S-2)/6$
...	...	...
K_{nodes}	$(RV-1)!C_N^{K_{nodes}}$	$(RS-1)!C_S^{K_{nodes}}$

разом, равна $O(C_{RVL}^{K_{links}}(K_{links}+1)K_{path}(W+2)^3)$. Здесь $C_{RVL}^{K_{links}}$ – количество перебираемых сочетаний, $K_{links}+1$ – количество переназначаемых виртуальных каналов.

3.4. Процедура репликации. В наихудшем случае процедура репликации производится для каждого из storage-элементов. Сложность поиска хранилища данных для репликации равна $O(S(N+W+1)^2)$ и заключается в поиске путей минимальной длины к каждому хранилищу данных для графа из $(N+W+1)$ элемента. Сложность поиска пути для канала поддержания консистентности равна $O((W+2)^2)$. Общая сложность, таким образом, составляет $O(RVL(S(N+W+1)^2+(W+2)^2))$.

3.1. Зависимость сложности алгоритма от глубины ограниченного перебора. При исследовании сложности алгоритма с нулевой глубиной ограниченного перебора (при значениях $K_{nodes}=0$ и $K_{links}=0$) можно прийти к следующим выводам:

1) сложность алгоритма имеет кубическую зависимость от W, квадратичную зависимость от N и линейную зависимость от S (при отключении операции репликации алгоритм также линейно зависит от N);

2) при увеличении значений RV, RS и RVL сложность алгоритма возрастает как $O(RV\log RV)$, $O(RS\log RS)$ и $O(RVL\log RVL)$ соответственно;

3) алгоритм линейно зависит от параметра K_{path}.

Таблица 3. Изменение сложности алгоритма в зависимости от глубины перебора K_{links}

Глубина перебора K_{links}	Коэффициент увеличения сложности алгоритма в зависимости от RVL
1	RVL
2	$RVL(RVL-1)/2$
3	$RVL(RVL-1)(RVL-2)/6$
...	...
N	$C_{RVL}^{K_{links}}$

Таблица 4. Результаты исследований метода вычисления стоимости при полной загрузке

Способ вычисления стоимости	Точность: среднее количество назначенных запросов	Средняя загрузка элементов[5], %	Среднее время работы алгоритмов, с
Динамическая с выделением критической характеристики	95.63	96.3	63
Взвешенная сумма с учетом дефицита ресурсов	93.77	97.7	34

Примечание. Средняя загрузка вычисляется как средее арифметическое из загрузок по каждой из характеристик. Данная величина используется для сравнения, насколько плотно производится упаковка.

Таблица 5. Результаты исследований метода вычисления стоимости при неравномерной загрузке

Способ вычисления стоимости	Точность: среднее количество назначенных запросов	Средняя загрузка элементов	Максимальная загрузка по характеристикам	Минимальная загрузка по характеристикам	Среднее время работы алгоритмов, с
		%			
Динамическая с выделением критической характеристики	96.93	56.6	98.6	10	30
Взвешенная сумма с учетом дефицита ресурсов	97.58	56.8	98.9	10	21

При увеличении значений глубины перебора K_{nodes} и K_{links} сложность алгоритма возрастает как $C_N^{K_{nodes}}$ ($C_S^{K_{nodes}}$) и $C_{RVL}^{K_{links}}$ соответственно. Следует заметить, что при увеличении значения K_{nodes} увеличивается коэффициент в слагаемых $O(C_N^{K_{nodes}} RV(\log(RV) + N))$ и $O(C_S^{K_{nodes}} RS(\log(RS) + S))$, которые возрастают как $RV \log RV$ ($RS \log RS$ соответственно) в зависимости от RV (RS). Изменение сложности операции перебора в зависимости от значений K_{nodes} при использовании жадного алгоритма в процедуре ограниченного перебора показано в табл. 1. В случае полного перебора в процедуре ограниченного перебора сложность увеличивается, кроме того, в $(RV - 1)!$ ($(RS - 1)!$ соответственно) раз. Изменение сложности для переназначения с полным перебором показано в табл. 2.

В то же время при увеличении значения K_{links} меняется коэффициент в слагаемом $O(C_{RVL}^{K_{links}}(K_{links} + 1)K_{path}(W + 2)^3)$, который кубически зависит от W. Изменение сложности операции ограниченного перебора в зависимости от значений K_{links} приведено в табл. 3.

4. Экспериментальное исследование свойств алгоритма. Экспериментальные исследования проводились для выявления следующих свойств алгоритма.

1. Зависимость качества получаемых отображений и вычислительной сложности алгоритма от выбранного способа расчета стоимости элемента: динамическое вычисление с выбором критической характеристики или взвешенная сумма по всем критериям (см. разд. 2.1.2).

2. Зависимость качества получаемых отображений и вычислительной сложности алгоритма от используемого алгоритма в процедуре ограниченного перебора: жадный алгоритм (с назначением сначала текущего виртуального элемента или с его назначением вместе со снятыми элементами) или полный перебор (см. разд. 2.1.1).

3. Зависимость качества получаемых отображений и вычислительной сложности алгоритма от выбора метода назначения: компактное или сбалансированное назначение (см. разд. 2.1.2).

4. Эффективность процедуры репликации.

4.1. Зависимость от выбранного способа вычисления стоимости элемента. Для данного исследования проводилась серия запусков для случайно сгенерированного ЦОД, состоящего только из вычислительных узлов, со следующими характеристиками:

1) количество вычислительных узлов – 1000;

2) количество запросов – 100, в каждом запросе в среднем по 50 виртуальных машин;

Таблица 6. Результаты исследований процедуры ограниченного перебора при полной загрузке

Способ ограниченного перебора	Точность: среднее количество назначенных запросов	Средняя загрузка элементов, %	Среднее время работы алгоритмов, с
Без ограниченного перебора	93.22	97.4	7.5
Жадный алгоритм, текущий элемент назначается первым с глубиной перебора 1	93.77	97.7	34
Жадный алгоритм, текущий элемент назначается первым с глубиной перебора 2	93.9	97.7	230
Жадный алгоритм, текущий элемент назначается вместе со всеми снятыми элементами	93.78	97.7	40
Полный перебор	93.5	97.5	210

Таблица 7. Результаты исследований процедуры ограниченного перебора при неравномерной загрузке

Способ вычисления стоимости	Точность: среднее количество назначенных запросов	Средняя загрузка элементов	Максимальная загрузка по характеристикам	Минимальная загрузка по характеристикам	Среднее время работы алгоритмов, с
		%			
Без ограниченного перебора	97.09	56.6	98.6	10	9.8
Жадный алгоритм, текущий элемент назначается первым с глубиной перебора 1	97.58	56.8	98.9	10	21
Жадный алгоритм, текущий элемент назначается первым с глубиной перебора 2	97.7	56.8	99	10	95
Жадный алгоритм, текущий элемент назначается вместе со всеми снятыми элементами	97.48	56.7	98.9	10	22
Полный перебор	97.3	56.6	98.7	10	96

3) количество характеристик – 4: ⟨число ядер⟩, ⟨частота⟩, ⟨объем оперативной памяти⟩, ⟨объем дисковой памяти⟩; для каждой характеристики должно выполняться отношение недопустимости перегрузки физического ресурса;

4) тесты формировались таким образом, что возможно назначение всех запросов;

5) общая загрузка ресурсов (отношение общей требуемой емкости виртуальных элементов к емкости физических элементов по данной характеристике) задавалась двумя способами:

полная загрузка: загрузка по всем характеристикам равна 100%;

неравномерная загрузка: загрузка формировалась следующим образом: ⟨число ядер⟩ – 10%, ⟨частота⟩ – 40%, ⟨объем оперативной памяти⟩ – 70%, ⟨объем дисковой памяти⟩ – 100%.

Для каждой комбинации “способ вычисления стоимости элемента–загрузка ресурсов” проводилось 100 запусков, полученные результаты усреднялись. При этом использовалась операция ограниченного перебора с жадным переназначением с глубиной перебора 1.

Результаты проведенных исследований в зависимости от выбранного метода вычисления стоимости при полной загрузке по характеристикам показаны в табл. 4. Результаты при неравномерной загрузке – в табл. 5.

Из приведенных результатов видно, что динамическое вычисление стоимости с выделением критической характеристики назначает в среднем больше запросов при полной загрузке, в случае же неравномерной загрузки больше запросов назначается с применением взвешенной суммы. При использовании взвешенной суммы для вычисления стоимости в обоих случаях загрузка элементов получалась большей.

Рис. 3. Процент назначенных запросов при компактном (–◆–) и сбалансированном (–■–) назначениях

4.2. Зависимость от выбранной процедуры ограниченного перебора. Тестовые наборы генерировались таким же образом, как и в разд. 4.1. Исследование проводилось для следующих вариантов процедуры ограниченного перебора:

без ограниченного перебора;

с жадным алгоритмом в процедуре ограниченного перебора, текущий виртуальный элемент назначался первым с глубиной перебора 1;

с жадным алгоритмом в процедуре ограниченного перебора, текущий виртуальный элемент назначался первым с глубиной перебора 2;

с жадным алгоритмом в процедуре ограниченного перебора, текущий виртуальный элемент назначался вместе со всеми снятыми элементами согласно стоимости;

с полным перебором в процедуре ограниченного перебора[3].

Функция стоимости во всех случаях вычислялась как взвешенная сумма по характеристикам с учетом их дефицита. Результаты при полной загрузке показаны в табл. 6, результаты при неравномерной загрузке – в табл. 7.

Как видно из результатов, использование различных вариантов ограниченного перебора не позволяет получить значительного улучшения. Увеличение глубины перебора также не дает существенного выигрыша. Можно пояснить данные результаты тем, что генерировались независимые запросы и ограниченный перебор не позволяет получить выигрыш для таких запросов за приемлемое время. В сравнении с запуском без ограниченного перебора среднее время работы при глубине перебора 1 увеличилось более чем вдвое. При глубине перебора 2 или при использовании полного перебора время увеличилось уже на порядок.

Также можно заметить, что при неравномерной загрузке алгоритм показывал лучшие результаты (большее среднее значение назначенных запросов) за меньшее время. Это можно пояснить тем, что процедура ограниченного перебора вызывается только в случаях, когда элемент не может быть назначен, т.е. когда физические элементы уже имеют большую загрузку. При полной загрузке такие случаи возникают чаще, чем при неравномерной загрузке, и поэтому процедура ограниченного перебора вызывается чаще.

Алгоритм полного перебора в процедуре ограниченного перебора при ограничении времени работы каждого запуска процедуры не позволил получить преимущества. При запуске без ограничения времени работы процедуры время каждого запуска алгоритма увеличивалось на порядки, поэтому не включено в результаты.

4.3. Зависимость от выбора метода назначения. В данном разделе исследуется критерий выбора физических элементов при назначении виртуальных элементов:

1) компактное назначение: выбирается физический элемент с наименьшей стоимостью;

[3] При использовании полного перебора и жадного алгоритма с глубиной перебора 2 дополнительно налагалось ограничение на время одного запуска процедуры перебора: если за заданное время назначение не проводилось, то процедура возвращала неудачу.

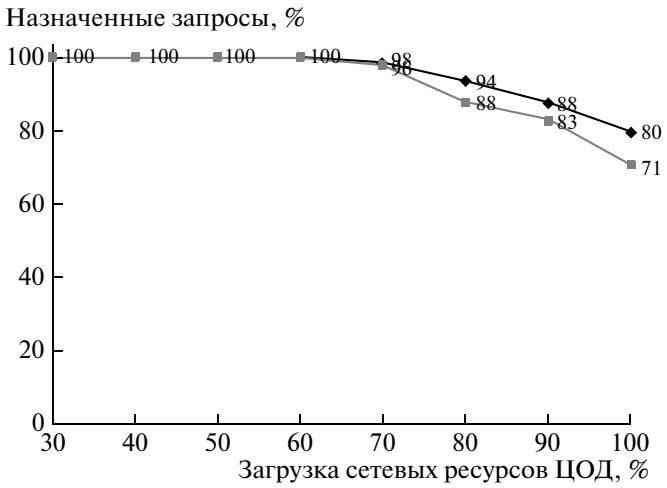

Рис. 4. Процент назначенных запросов в зависимости от использования процедуры репликации: –◆– алгоритм с репликацией, –■– алгоритм без репликации

2) сбалансированное назначение: выбирается элемент с наибольшей стоимостью.

Примером, показывающим различие между данными способами назначения, является ЦОД, состоящий только из вычислительных узлов, со следующими характеристиками:

1) количество вычислительных узлов – 1000;

2) количество запросов – 100, в каждом из которых – 100 виртуальных машин;

3) количество характеристик – 1: ⟨частота⟩;

4) общая загрузка равна 50%.

При компактном назначении получаются следующие результаты:

1) из 1000 вычислительных узлов виртуальные машины назначаются только в 500 узлов;

2) в каждом из узлов, на которые назначены виртуальные машины, загрузка близка к 100%.

При сбалансированном назначении получаются следующие результаты:

1) виртуальные машины назначаются во все вычислительные узлы;

2) в каждом из узлов средняя загрузка равна 50%.

Как видно из примера, компактная загрузка позволяет оставить часть вычислительных узлов неиспользуемыми, что может привести к экономии энергии. Сбалансированная нагрузка дает возможность уменьшить среднюю загрузку узлов.

Компактное назначение предлагается использовать, когда физические ресурсы сети не являются критическим ресурсом, т.е. в случаях, когда отношение стоимостей требуемых сетевых ресурсов к имеющимся сетевым ресурсам меньше заданного порога. В противном случае предлагается применить сбалансированное назначение. Указанное свойство поясняют экспериментальные исследования с ЦОД со следующими характеристиками:

использовались ЦОД с топологией fattree [2] с тремя уровнями коммутационных элементов;

всего в ЦОД имелось 60 вычислительных узлов и 60 хранилищ данных;

назначение проводилось для 100 запросов, в каждом из которых – в среднем 10 виртуальных каналов;

средняя загрузка вычислительных узлов и хранилищ данных фиксирована и равна 75%;

средняя загрузка сетевых ресурсов ЦОД варьировалась от 30 до 100%.

Результаты проведенных запусков при компактном и сбалансированном назначении в зависимости от загрузки виртуальных каналов показаны на рис. 3. Как видно из графика, при загрузке виртуальных каналов больше 60% сбалансированное назначение позволяет увеличить процент назначенных запросов.

4.4. Эффективность процедуры репликации. Эффективность процедуры репликации исследовалась на тех же наборах запросов, что и в разд. 4.3. При этом проводился запуск с и без использования процедуры репликации. При каждом запуске применялось сбалансированное назначение. Результаты изображены на графике на рис. 4. Исследования показывают, что процедура репликации позволяет повысить процент назначаемых запросов.

Заключение. В данной работе была расширена постановка задачи отображения запросов на физические ресурсы на возможность задания произвольного набора требований SLA для каждого типа ресурсов. Предложен алгоритм отображения запросов на физические ресурсы для ЦОД с раздельными планировщиками для различных типов ресурсов. На основе проведенного исследования свойств алгоритма сделаны выводы об эффективных областях его применения. Данный алгоритм может быть использован при построении облачных платформ, например, в платформе OpenStack [13] или в самоорганизующейся облачной платформе, разрабатываемой в центре прикладных исследований компьютерных сетей[4].

Для запуска алгоритма и исследования его применимости для вашего центра обработки данных необходимо:

1) написать запрос на электронный адрес ev@arccn.ru с указанием ФИО;

2) установить любой VNC-клиент;

3) прочитать инструкцию.

Инструментальная система предоставляет возможность описать ресурсы центра обработки данных, создать набор запросов и посмотреть, как эти запросы будут распределены на физические ресурсы выбранным алгоритмом.

СПИСОК ЛИТЕРАТУРЫ

1. *Вдовин П.М, Зотов И.А., Костенко В.А., Плакунов А.В., Смелянский Р.Л.* Задача распределения ресурсов центров обработки данных и подходы к ее решению // VII Московская междунар. конф. по исследованию операций (ORM2013). М.: ВЦ РАН, 2013. Т. 2. С. 30–32.
2. *Вдовин П.М, Зотов И.А., Костенко В.А., Плакунов А.В., Смелянский Р.Л.* Сравнение различных подходов к распределению ресурсов в центрах обработки данных // Изв. РАН. ТиСУ. 2014. № 5; *Vdovin P.M., Zotov I.A., Kostenko V.A., Plakunov A.V., Smelyansky R.L.* Comparing Various Approaches to Resource Allocating in Data Centers // J. Computer and Systems Sciences Intern. 2014. V. 53. № 5.
3. *Amies A., Sluiman H., Tong Q.G., et al.* Developing and Hosting Applications on the Cloud. Boston: IBM Press, 2012.
4. *Wustenhoff E.* Service Level Agreement in the Data Center. Sun BluePrints, Sun Microsystems Professional Series, 2002.
5. *Фуругян М.Г.* Некоторые алгоритмы анализа и синтеза многопроцессорных вычислительных систем реального времени // Программирование. 2014. № 1. С. 36–44.
6. *Фуругян М.Г.* Некоторые алгоритмы решения минимаксной задачи составления многопроцессорного расписания // Изв. РАН. ТиСУ. 2014. № 2. С. 48–54; *Furugyan M.G.* Some Algorithms of Solving Minimax Multiprocessor Scheduling Problem // J. Computer and Systems Sciences Intern. 2014. V. 53. № 2. P. 195–200.
7. *Костенко В.А., Плакунов А.В.* Алгоритм построения одноприборных расписаний, основанный на схеме муравьиных колоний // Изв. РАН. ТиСУ. 2013. № 6. С. 87–96; *Kostenko V.A., Plakunov A.V.* An Algorithm for Constructing Single Machine Schedules Based on Ant Colony Approach // J. Computer and Systems Sciences Intern. 2013. V. 52. № 6. P. 928–937.
8. *Костенко В.А.* Алгоритмы построения расписаний для вычислительных систем реального времени, допускающие использование имитационных моделей // Программирование. 2013. № 5. С. 53–71.
9. *Костенко В.А., Шестов П.Е.* Жадный алгоритм совместного планирования вычислений и обменов в системах реального времени // Изв. РАН. ТиСУ. 2012. № 5. С. 35–49; *Kostenko V.A., Shestov P.E.* A Greedy Algorithm for Combined Scheduling of Computations and Data Exchanges in Real_Time Systems // J. Computer and Systems Sciences Intern. 2012. V. 51. № 5. P. 648–662.
10. *Зорин Д.А., Костенко В.А.* Алгоритм синтеза архитектуры вычислительной системы реального времени с учетом требований к надежности // Изв. РАН. ТиСУ. 2012. № 3. С. 76–83; *Zorin D.A., Kostenko V.A.* Algorithm for Synthesis of Real-Time Systems under Reliability Constraints // J. Computer and Systems Sciences Intern. 2012. V. 51. № 3. P. 410–417.
11. *Coffman E.G., Garey M.R., Johnson D.S.* Approximation Algorithms for Bin Packing: A survey // Approximation Algorithms for NP-hard Problems. Boston: PWS Publishing Co., 1996. P. 46–93.
12. *Eppstein D.* Finding the k Shortest Paths // SIAM J. Computing. 1998. V. 28. № 2. P. 652–673.
13. *Pepple K.* Deploying OpenStack. Sebastopol: O'Reilly, 2011.

[4] Информация об исследованиях, проводимых в центре прикладных исследований, предоставлена в электронном виде по URL-адресу http://arccn.ru/research/762.