ИЗВЕСТИЯ РАН. ТЕОРИЯ И СИСТЕМЫ УПРАВЛЕНИЯ, 2015, № 1, с. 61–71

КОМПЬЮТЕРНЫЕ МЕТОДЫ

УДК 004.031.43

АЛГОРИТМ РАСПРЕДЕЛЕНИЯ РЕСУРСОВ В ЦЕНТРАХ ОБРАБОТКИ ДАННЫХ С ЕДИНЫМ ПЛАНИРОВЩИКОМ ДЛЯ РАЗЛИЧНЫХ ТИПОВ РЕСУРСОВ*

© 2015 г. И. А. Зотов, В. А. Костенко

Москва, МГУ

Поступила в редакцию 15.09.14 г.

Рассматривается алгоритм отображения запросов на физические ресурсы для центров обработки данных с единым планировщиком для различных типов ресурсов: вычислительных, сетевых и систем хранения данных. Алгоритм основан на “компактном” размещении в центре обработки данных запроса на создание виртуальной сети. Требуемый баланс между вычислительной сложностью и качеством получаемых отображений может задаваться путем ограничения максимально допустимой глубины перебора. Приводятся теоретические и экспериментальные результаты исследования его свойств.

DOI: 10.7868/S000233881501014X

Введение. В [1] представлены результаты сравнения алгоритмов отображения запросов на физические ресурсы центра обработки данных (ЦОД) для режима его использования Infrastructure-as-a-Service (IaaS) [2]. В [1] рассматривались следующие алгоритмы:

используемые в платформе OpenStack [3]: “назначение элемента запроса на первый подходящий физический ресурс” или “выбор случайным образом физического ресурса из множества подходящих ресурсов”;

основанный на применении схемы муравьиных колоний [4];

сочетающие жадные стратегии и стратегии ограниченного перебора.

Первый алгоритм, сочетающий жадные стратегии и стратегии ограниченного перебора, наиболее эффективен в случае, когда критическими ресурсами являются вычислительные ресурсы или ресурсы хранения данных. Он детально описан в работе [5].

Второй алгоритм, сочетающий жадные стратегии и стратегии ограниченного перебора, наиболее эффективен в случае, когда критическими ресурсами являются сетевые ресурсы. Основные подходы к обслуживанию ЦОД в подобных условиях рассмотрены в [6]. Алгоритм основан на “компактном” размещении в ЦОД запроса на создание виртуальной сети. В данной работе приведены детальное описание алгоритма и результаты экспериментального исследования его свойств. Предлагаемый алгоритм допускает возможность задания SLA (Service Level Agreement – соглашения об уровне услуг) для всех типов ресурсов ЦОД. 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. Типы и виды SLA представлены, в частности, в [7].

1. Математическая постановка задачи распределения ресурсов ЦОД. Приведем математическую постановку задачи из работы [4] без изменений, так как она необходима для понимания работы алгоритма.

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 скалярного аргумента, задающие соответственно характеристики вычислительных узлов, хранилищ данных, коммутационных элементов и каналов передачи данных:

* Работа выполнена при финансовой поддержке Министерства образования и науки Российской Федерации. Соглашение № 14.607.21.0070.

$$ph(p) = (ph_1(p), ph_2(p), \ldots, ph_{n_1}(p)),$$

$$mh(m) = (mh_1(m), mh_2(m), \ldots, mh_{n_2}(m)),$$

$$kh(k) = (kh_1(k), kh_2(k), \ldots, kh_{n_3}(k)),$$

$$lh(l) = (lh_1(l), lh_2(l), \ldots, lh_{n_4}(l)).$$

Здесь $p \in P, m \in M, k \in K, l \in L$. Компоненты векторной функции могут принимать целочисленные и вещественные значения. В данной работе предполагается, что для коммутационных элементов и каналов передачи данных определяются одинаковые характеристики.

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 скалярного аргумента, задающие характеристики запрашиваемого виртуального элемента (требуемое качество сервиса (SLA)):

$$wg(w) = (wg_1(w), wg_2(w), \ldots, wg_{n_1}(w)),$$

$$sg(s) = (sg_1(s), sg_2(s), \ldots, sg_{n_2}(s)),$$

$$eg(e) = (eg_1(e), eg_2(e), \ldots, eg_{n_4}(e)).$$

Здесь $w \in W, s \in S, e \in E$. Компоненты векторной функции также принимают целочисленные или вещественные значения. 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 элементов запросов и соответствующих характеристик физических ресурсов. Обозначим через x_i характеристику i-го элемента запроса, через y_j – соответствующую ей характеристику j-го физического ресурса. 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 – множество элементов запросов, назначенных на выполнение на физическом ресурсе y_j.

2. Соответствие типа запрашиваемого ресурса типу физического ресурса:

$$x_i = y_i.$$

3. Наличие требуемых характеристик у физического ресурса:

$$x_i \le y_i.$$

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:

$$ph_{res}(p) = ph(p) - \sum_{w \in W_p} wg(w),\ mh_{res}(m) = mh(m) - \sum_{s \in S_m} sg(s),$$

$$kh_{res}(k) = kh(k) - \sum_{e \in E_k} eg(e),\ lh_{res}(l) = lh(l) - \sum_{e \in E_l} eg(e). \qquad (1.1)$$

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. При этом часть 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 $(m', l_1, k_1, \ldots, k_{n-1}, l_n, m)$; $k_i \in K, l_i \in L, m' \in M$, 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.

Входом алгоритма назначения запросов на физические ресурсы являются 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 для ЦОД с единым планировщиком для всех типов ресурсов. 2.1. Общая схема работы алгоритма. Алгоритм осуществляет назначение элементов запроса в соответствии с жадной схемой, в случае невозможности назначения очередного элемента вызывается процедура ограниченного перебора. В качестве основного критерия оптимизации рассматривается наиболее компактное размещение элементов запроса с точки зрения использования сетевых ресурсов. Подобный подход рассмотрен в [8]. Для процедуры ограниченного перебора задается параметр, ограничивающий глубину перебора, что позволяет задавать требуемый баланс между вычислительной сложностью и точностью алгоритма.

Общая схема алгоритма выглядит следующим образом.

Шаг 1. Если множество ресурсных запросов $\{G_i\}$ не пусто, то выбрать очередной запрос G_i в соответствии с жадным критерием K_G, в противном случае – завершение работы алгоритма.

Шаг 2. Сформировать из элементов запроса G_i множество $U \equiv \{W \cup S\}$.

Шаг 3. Выбрать очередной элемент $e \in U$ в соответствии с жадным критерием K_f, поместить e в очередь Q элементов, ожидающих назначения.

Шаг 4. Выбрать из Q элемент e в соответствии с жадным критерием K_Q, выполнить следующие действия:

4a) сформировать множество физических ресурсов Ph, на которые может быть назначен данный элемент e, т.е. для которого выполнены отношения корректности отображения в случае назначения запроса e на физический ресурс. Если данное множество пусто, то вызвать процедуру ограниченного перебора. При неуспешном завершении процедуры удалить ранее назначенные элементы запроса G_i и переопределить значения характеристик физических ресурсов в соответствии с функциями (1.1), удалить запрос из множества $\{G_i\}$, *перейти к шагу 1*;

4b) выбрать физический ресурс $ph \in Ph$ в соответствии с жадным критерием K_{ph};

4c) сформировать пути между ph и физическими ресурсами, на которые назначены элементы запроса, связанные виртуальными каналами с элементом e, в случае неуспеха удалить ph из Ph, перейти к шагу 4b;

4d) назначить e на ph и переопределить значения характеристик физических ресурсов в соответствии с функциями (1.1);

4e) удалить e из Q и U;

4f) добавить в Q множество элементов запроса $\{e_t\}$, связанных виртуальными каналами с элементом e, $e_t \in U$.

Ш а г 5. Если Q не пусто, то перейти к шагу 4.

Ш а г 6. Если U не пусто, то перейти к шагу 3, в противном случае перейти к шагу 1.

2.2. 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 при отсутствии возможности назначить очередной элемент запроса e ни на один физический ресурс ph. Процедура ограниченного перебора выполняет поиск среди множества A уже назначенных элементов запросов такого подмножества $A_m \subseteq A$, что элемент e может быть назначен после переназначения элементов из A_m. Перебор ограничивается числом m, задающим максимальное количество физических элементов, для которых возможно переназначение. При переборе рассматриваются только такие множества из m ресурсов, которые в сумме имеют достаточное количество остаточных ресурсов для назначения текущего элемента.

Общая схема процедуры ограниченного перебора.

1. Для заданного m сформировать все множества $A_m \subseteq A$ из m физических ресурсов.

2. Если для некоторого множества A_m суммарная остаточная емкость ресурсов достаточна для назначения текущего элемента:

2.1) произвести снятие элементов, которые назначены на физические ресурсы из множества A_m и для которых разрешена миграция;

2.2) произвести снятие всех виртуальных каналов, связанных с элементами из множества A_m;

2.3) произвести попытку переназначить элементы множества A_m и назначить элемент e, а также всех связанных с ними виртуальных каналов. Возможны три варианта:

i) назначить элемент e, а затем все элементы множества A_m с использованием жадной схемы (шаги 2–4 основной схемы алгоритма без вызова процедуры ограниченного перебора);

ii) назначить элементы множества $A_m \bigcup \{e\}$ с использованием жадной схемы (шаги 2–4 основной схемы алгоритма без вызова процедуры ограниченного перебора);

iii) назначить элементы множества $A_m \bigcup \{e\}$ с использованием полного перебора;

2.4) в случае успешного назначения всех элементов вернуть успех, в противном случае восстановить исходные назначения элементов из A_m.

3. Если после перебора всех сформированных множеств элемент e назначить не удается, вернуть неуспех.

Указанная схема при фиксированном m может позволить за полиномиальное время произвести переназначение с возможностью назначения элемента e.

2.3. П р о ц е д у р а н а з н а ч е н и я в и р т у а л ь н ы х к а н а л о в н а ф и з и ч е с к и е р е с у р с ы с е т и о б м е н а. Назначение виртуальных каналов производится после назначения очередного элемента. При этом назначаются виртуальные каналы, связывающие этот элемент с уже назначенными элементами запроса. Поиск пути для назначения виртуального канала производится на основе алгоритма Дейкстры [9], модифицированного так, что в путь могут входить только коммутационные элементы и каналы обмена физической сети ЦОД, для которых выполняются отношения корректности отображения. При невозможности назначения виртуального канала, связывающего storage-элемент, вызывается процедура репликации, которая создает реплику storage-элемента. При создании реплики необходимо также назначить канал, обеспечивающий консистентность между storage-элементом и его репликой.

При построении маршрутов для каналов может использоваться один из следующих критериев выбора очередного элемента маршрута.

1. Максимальная остаточная величина пропускной способности физического элемента (коммутатора или канала). Данный критерий позволяет сохранять связность графа физических ресурсов для упрощения назначения последующих запросов и утилизации возможно большего числа ресурсов. Применение этого критерия повышает качество работы алгоритма в ЦОД с древовидной топологией.

2. Минимальная остаточная величина пропускной способности физического элемента. При этом увеличивается утилизация сетевых ресурсов при возможно более компактном размещении запросов. Применение этого критерия повышает точность алгоритма в ЦОД с топологией fat-

tree [10] и другими аналогичными топологиями с дополнительными каналами обеспечения доступности.

2.4. Жадные критерии. Качество работы алгоритма существенно зависит от выбора жадных критериев. Критерии задаются для выбора очередного запроса K_G, очередного элемента запроса K_f и для выбора физического ресурса для назначения этого элемента K_{ph}.

Критерий K_f основан на функции стоимости $r(e)$ назначения элемента e. Элемент характеризуется вектором значений требуемых характеристик ресурса $(r_{e,1}, r_{e,2}, \ldots, r_{e,n})$. Для определения стоимости вычисляется дефицит ресурса по каждой характеристике:

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

Данная величина вычисляется как частное, в котором числитель – разница между общим значением требуемой характеристики ресурса по всем запросам и значением имеющихся физических ресурсов по данной характеристике. Знаменателем является общая сумма требуемых ресурсов по данной характеристике. В случае, если имеющихся ресурсов достаточно для назначения запроса, дефицит $d(i)$ может принимать отрицательные значения. Функция стоимости вычисляется как взвешенная сумма требуемых характеристик ресурса с учетом их дефицита:

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

В качестве критерия K_f предлагается выбирать элемент, для которого физические ресурсы в текущий момент времени имеют наибольший дефицит. Таким образом, среди элементов запроса выбирается такой элемент e, стоимость $r(e)$ которого максимальна.

Для выбора физического ресурса $ph \in Ph$ для назначения очередного элемента запроса e предлагается критерий K_h, при котором выбирается элемент ph с минимальной суммарной длиной путей до всех элементов, на которых размещены назначенные элементы запроса, связанные виртуальными каналами с назначаемым элементом e. Длина пути измеряется в количестве дуг физической сети обмена входящих в путь. Для определения ph вместо построения реальных путей на графе ресурсов используется итеративный поиск в ширину относительно физических элементов, содержащих указанные назначения. Данный подход позволяет улучшить временную сложность алгоритма за счет отказа от применения поиска в глубину и конструктивного алгоритма поиска путей (алгоритма Дейкстры). В качестве критериев поиска в ширину принимаются возможность назначения элемента e на данный физический ресурс и возможность каналов данных и коммутаторов поддерживать виртуальные каналы до данного элемента с требуемыми характеристиками SLA. В целях оптимизации в алгоритме предусматривается генератор подходящих кандидатов на назначение для элемента e, реализующий поиск в ширину, что сокращает время перебора за счет отказа от формирования полного множества кандидатов. Критерий выбора очередного запроса для назначения K_G выбирает запрос, взвешенная сумма запрашиваемых ресурсов которого максимальна.

2.5. Генератор кандидатов на размещение. Генератор кандидатов на размещение используется в алгоритме для сокращения пространства поиска. Генератор получает на вход очередной элемент запроса, ожидающий назначения, и множество пар “ресурс–канал”, где ресурс – физический ресурс, на который уже назначен элемент запроса, связанный с назначаемым элементом, а канал – виртуальный канал до этого элемента запроса. Для каждой пары “ресурс–канал” производится ненаправленный поиск в ширину с сохранением состояния поиска. Поиск в ширину продолжается до первого элемента, для которого возможно назначение рассматриваемого элемента запроса. Дальнейший поиск производится из текущего состояния с сохранением меток о посещенных или отброшенных элементах пространства поиска. При поиске в ширину производится отсечение коммутационных элементов и каналов, не способных поддерживать данный виртуальный канал из-за невозможности выполнения SLA.

Для каждой из пар “ресурс–канал” составляется множество отобранных таким образом кандидатов, т.е. физических ресурсов, которые удовлетворяют требованиям SLA при назначении на

них рассматриваемого элемента запроса. При обнаружении нового кандидата в процессе поиска в ширину производится пересечение таких множеств. Если пересечение не пусто, то для каждого из элементов пересечения строится маршрут с минимальной длиной для каждого из каналов. Данные маршруты получаются автоматически в процессе поиска в ширину, однако существует возможность взаимного влияния каналов друг на друга в том случае, если они назначаются на один и тот же физический элемент. Если в процессе назначения физический элемент, входящий в состав очередного маршрута, не способен поддерживать искомый виртуальный канал, производится повторный поиск в ширину с учетом изменившихся остаточных ресурсов.

В случае неуспеха поиск в ширину продолжается для следующей пары "ресурс–канал". Поиск оканчивается полным неуспехом, если поиск в ширину завершился для каждой пары, т.е. рассмотрены все возможные физические ресурсы сети, а кандидата для размещения не найдено. В этом случае в алгоритме фиксируется неуспех и вызывается процедура ограниченного перебора.

2.6. Процедура репликации. Процедура репликации возможна в случае, если одним из элементов запроса, соединяемым виртуальным каналом $e = (w, s)$, 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). Кандидаты рассматриваются в порядке возрастания суммарной длины маршрутов всех виртуальных каналов $e = (w \in W, s)$ (включающих данный storage-элемент s) к данной реплике m в соответствии с критерием K_{ph}. Кроме того, необходимо проложить канал l для поддержания консистентности между репликой и исходным storage-элементом (требуемая пропускная способность канала l определяется интенсивностью потока данных, необходимых для поддержания консистентности). В случае невозможности прокладки канала l с требуемой пропускной способностью процедура переходит к рассмотрению следующего кандидата на размещение. Подобная техника в отношении виртуальных машин, однако, без учета эффектов увеличения доступности ресурса, рассмотрена в [11].

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

Данная процедура позволяет устранять перегрузку входящих в физическое хранилище данных каналов обмена. Использование данной процедуры может увеличить число размещенных запросов в случае, 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.

3. Вычислительная сложность алгоритма. Пусть N – число вычислительных узлов, S – число хранилищ данных, W – число коммутаторов в сети передачи данных, L – число физических каналов передачи данных. Пусть далее RV – общее число виртуальных машин всех ресурсных запросов, RS – общее число storage-элементов, RL – общее число виртуальных каналов. Через m обозначим коэффициент перебора в процедуре ограниченного перебора (см. разд. 2.2).

Основная процедура алгоритма (см. разд. 2.1).

1. Формирование множества элементов для назначения и выбор очередного элемента для назначения (шаги 2, 3). Сложность этапа эквивалентна сложности сортировки массива и равна $O((RV + RS)\log(RV + RS))$. Здесь и далее log – десятичный логарифм.

2. Назначение очередного элемента (шаг 4). В процессе выполнения этапа определяются назначения виртуальных каналов в соответствии с политикой генерации кандидатов на размещение (см. разд. 2.5). В качестве худшего случая можно рассматривать назначение последнего неназначенного элемента из запроса с полносвязным графом. В этом случае требуется решить $RV + RS - 1$ задач поиска в ширину на графе с $N + S + W$ вершинами и L вершинами. Сложность данного этапа, таким образом, составляет $O((RV + RS - 1)(N + S + W + L))$.

3. Процедура ограниченного перебора. Количество перебираемых сочетаний вычисляется как $\max(C_N^m, C_S^m)$. На каждом этапе производится попытка переназначения элементов, сложность которой зависит от выбранного способа ограниченного перебора:

3.1) жадный алгоритм переназначения, сложность составляет $O(t\log t+\max(N,S))$, где $t=\max(RV,RS)$ – число элементов запроса, подходящих для переназначения;

3.2) алгоритм полного перебора, сложность которого в худшем случае достигает $O(t!m)$, где t определена в п. 1).

Процедура ограниченного перебора может быть инициирована в процессе назначения любого элемента запроса. Принимая во внимание данный факт и вводя обозначение $Q=2\max(RV, RS)$, получаем следующую оценку временной сложности основной процедуры алгоритма в наихудшем случае: $O(Q\log(Q)+Q(N+S+W+L)C^m_{N+S}(Q\log(Q)+N+S))$ для случая использования жадного алгоритма в процедуре ограниченного перебора.

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

1. Зависимость качества получаемых отображений и вычислительной сложности алгоритма от используемого алгоритма назначения в процедуре ограниченного перебора (см. алгоритмы i), ii), iii) разд. 2.2).

2. Зависимость качества получаемых отображений и вычислительной сложности алгоритма от выбранного способа расчета стоимости элемента: динамическое вычисление с выбором критической характеристики или взвешенная сумма по критериям (см. разд. 2.4).

3. Зависимость качества получаемых отображений и вычислительной сложности алгоритма от использования генератора кандидатов на размещение по сравнению с прямым поиском кандидатов (см. разд. 2.5).

4. Эффективность процедуры репликации (см. разд. 2.6).

4.1. Зависимость качества отображений от алгоритма, используемого в процедуре ограниченного перебора. Для данного исследования проводилась серия запусков для случайно сгенерированной архитектуры ЦОД, не содержащего систем хранений данных. Тесты формировались в соответствии со следующими характеристиками:

количество вычислительных узлов – 1000;

количество запросов – 100, в каждом запросе в среднем 50 виртуальных машин, не связанных между собой виртуальными каналами, отклонение от ожидания – не более 50% (от 25 до 75 виртуальных машин);

количество критериев SLA – 4: число ядер процессора физического сервера или виртуальной машины, объем кэш-памяти, объем оперативной памяти, объем дисковой памяти;

запросы сформированы таким образом, что существует хотя бы одно отображение, размещающее все запросы.

Потенциальная загрузка ресурсов (отношение требуемой емкости виртуальных машин к емкости физических элементов) задавалась двумя способами:

полная загрузка: загрузка по всем характеристикам равна 100%;

неравномерная загрузка: 100% – по числу ядер, 70% – по объему кэш-памяти, 30% – по объему оперативной памяти, 10% – по объему дисковой памяти.

Рассматриваемый пример на практике соответствует запросам на выделение виртуальных серверов.

В ходе исследования по указанным шаблонам генерировались 100 случайных тестовых примеров, полученные в ходе исследования значения усреднялись. При выборе начального элемента запроса на размещение использовался критерий K_f динамического определения критического ресурса (см. разд. 2.4)

Результаты проведенных исследований в зависимости от выбора алгоритма перебора (см. обозначения разд. 2.2) в процедуре ограниченного перебора при полной загрузке приведены в табл. 1, для неполной загрузки – в табл. 2. Значение в столбцах с максимальной (минимальной) загрузкой по характеристике соответствует характеристике с наибольшей (наименьшей) загрузкой.

Как видно из приведенных результатов, использование различных вариантов ограниченного перебора не ведет к существенному улучшения результатов по точности (количеству назначенных запросов). При увеличении глубины ограниченного перебора время работы алгоритма вырастает экспоненциально. Следует отметить, что в силу независимости запросов стратегии ограниченного перебора не позволяют получить существенного выигрыша за приемлемое время. С этим же связан незначительный по сравнению с ограниченным перебором выигрыш полного перебора, поскольку на время работы процедуры полного перебора накладывалось ограничение. Это ограничение задавалось числом рассматриваемых возможных сочетаний физических ресур-

Таблица 1. Результаты исследований процедуры ограниченного перебора при полной загрузке

Способ ограниченного перебора	Точность: среднее количество назначенных запросов	Средняя загрузка вычислительных узлов, %	Среднее время работы алгоритма, с
Без ограниченного перебора	93.31	97.4	5
Жадный алгоритм A1, глубина перебора 2	93.7	97.7	19
Жадный алгоритм A1, глубина перебора 3	93.9	97.8	195
Жадный алгоритм A2, глубина перебора 2	93.81	97.8	20
Полный перебор A3	93.7	97.7	210

Таблица 2. Результаты исследований процедуры ограниченного перебора при неравномерной загрузке

Способ ограниченного перебора	Точность: среднее количество назначенных запросов	Средняя загрузка вычислительных узлов	Максимальная загрузка по характеристике	Минимальная загрузка по характеристике	Среднее время работы алгоритмов, с
		%			
Без ограниченного перебора	96.0	55.6	98.5	10	6
Жадный алгоритм i, глубина перебора 2	97.61	55.7	98.8	10	17
Жадный алгоритм i, глубина перебора 3	97.81	55.8	99	10	91
Жадный алгоритм ii, глубина перебора 2	97.80	55.8	98.9	10	20
Полный перебор iii	93.80		99		95

Таблица 3. Результаты исследований метода вычисления стоимости при полной загрузке

Способ вычисления стоимости	Точность: среднее количество назначенных запросов	Средняя загрузка элементов, %	Среднее время работы алгоритмов, с
Динамический с выделением критической характеристики	95.1	96.2	56
Взвешенная сумма с учетом дефицита ресурсов	93.8	96.3	21

сов. Данное ограничение сокращает пространство поиска в 100 раз относительно эмпирически прогнозируемого.

Уменьшение времени работы алгоритма в условиях неравномерной загрузки объясняется тем, что процедура ограниченного перебора вызывается только в случае отказа размещения элемента основной процедурой алгоритма. При неравномерной загрузке число таких ситуаций уменьшается (средняя загрузка значительно ниже).

4.2. Зависимость качества отображений от метода вычисления стоимости элемента. Конфигурация тестовых примеров полностью аналогична конфигурациям, приведенным в разд. 4.1. В полной аналогии с вышеприведенной методологией значения усреднялись для 100 случайно сгенерированных наборов. Рассматривались два варианта расчета вычисления стоимости элемента:

приведенный в разд. 2.4 динамический критерий критичности ресурса;

взвешенная сумма всех параметров ресурсов с учетом их дефицита.

Во всех тестовых конфигурациях использовалась процедура ограниченного перебора с жадным алгоритмом назначения и глубиной 2. Результаты работы алгоритма для наборов с полной загрузкой приведены в табл. 3, для наборов с неравномерной загрузкой – в табл. 4. Значение в столбцах с максимальной (минимальной) загрузкой по характеристике соответствует характеристике с наибольшей (наименьшей) загрузкой.

Таблица 4. Результаты исследований метода вычисления стоимости при неравномерной загрузке

Способ вычисления стоимости	Точность: среднее количество назначенных запросов	Средняя загрузка элементов	Максимальная загрузка по характеристике	Минимальная загрузка по характеристике	Среднее время работы алгоритмов, с
		%			
Динамический с выделением критической характеристики	97.6	56.7	98.9	10	25
Взвешенная сумма с учетом дефицита ресурсов	97.58	56.7	98.7	10	18

Таблица 5. Результаты исследований эффективности генератора кандидатов

Метод подбора кандидатов на размещение	Точность: среднее число назначенных запросов	Загрузка вычислительных элементов и хранилищ данных, %			Время работы алгоритма, с
		средняя	максимальная	минимальная	
Генератор кандидатов	96.3	76	98	10	67
Поиск в ширину	96.4	76	94	10	194

Из представленных результатов видно, что в условиях полной загрузки динамическое вычисление стоимости позволяет получить незначительный прирост в точности, однако время работы возрастает в 1.5 раза. При неравномерной загрузке критичность ресурса перестает играть решающую роль, и оба подхода дают идентичные результаты.

4.3. Зависимость качества отображений от использования генераторов кандидатов на размещение. Для исследований применялась топология fat-tree с тремя уровнями коммутационных элементов со следующими характеристиками:

50 вычислительных узлов и 50 хранилищ данных, распределенных в соотношении 5 : 5 по 10 коммутаторам первого уровня;

по 10 коммутаторов второго и третьего уровней.

Запросы удовлетворяли следующим характеристикам:

количество запросов – 100,

в каждом запросе от 4 до 7 виртуальных машин и/или storage-элемента,

каждый запрос представляет собой полносвязный граф,

потенциальная загрузка вычислительных узлов и хранилищ данных составляет 80%.

Результаты проведенных исследований (табл. 5) показывают, что генератор кандидатов не увеличивает точность алгоритма, однако позволяет значительно уменьшить временную сложность алгоритма за счет оптимизации пространства поиска.

4. 2. Эффективность репликации. Исследования проводились для топологии ЦОД, рассмотренной в разд. 4.3. Тестовые примеры формировались по следующим шаблонам:

количество запросов – 100;

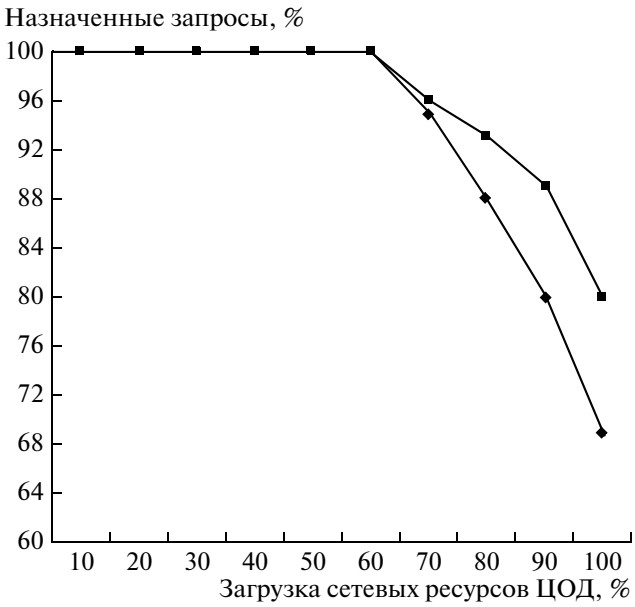
запрос представляет собой две виртуальные машины, подключенные к одному storage-элементу;

потенциальная загрузка вычислительных узлов – 100%;

потенциальная загрузка хранилищ данных – 100%;

потенциальная загрузка сетевых ресурсов ЦОД возрастает от 10 до 100%.

Результаты приведены на графике рисунка. Исследования показывают, что использование процедуры репликации позволяет повысить качество работы алгоритма по числу назначаемых запросов.



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

Заключение. Рассмотренный в данной работе алгоритм отображения запросов на физические ресурсы центра обработки данных используется в планировщике самоорганизующейся облачной платформы [12]. Самоорганизующаяся облачная платформа обладает следующими основными особенностями.

1. Допускает согласованное планирование (в смысле соблюдения соглашений о качестве сервиса) вычислительных ресурсов, ресурсов хранения данных и сетевых ресурсов, рассматривая эти ресурсы как управляемые.

2. Допускает миграцию виртуальных ресурсов с целью устранения сегментации физических ресурсов, возникающую в ходе работы ЦОД.

3. Позволяет администрировать виртуальную сеть, например, задавать определенную политику маршрутизации потоков в виртуальной сети.

4. Допускает возможность предоставления и использования в виртуальной сети виртуальных сетевых функций, управляемых в том числе сторонними поставщиками. Позволяет объединять сети, управляемые различными организациями, и эффективно передавать потоки данных между ними.

СПИСОК ЛИТЕРАТУРЫ

1. *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 International. 2014. № 5.
2. *Amies A., Sluiman H., Tong Q.G. et al.* Developing and Hosting Applications on the Cloud. Boston: IBM Press, 2012.
3. *Pepple K.* Deploying OpenStack. Sebastopol CA: O'Reilly Media, 2011.
4. *Plakunov A.V., Kostenko V.A.* Data Center Resource Mapping Algorithm Based on the Ant Colony Optimization // Proc. Intern. Conf. on Networks 2014: SDN&NFV. Moscow: IEEE Press, 2014.
5. *Вдовин П.М., Костенко В.А.* Алгоритм распределения ресурсов в центрах обработки данных с раздельными планировщиками для различных типов ресурсов // Изв. РАН. ТиСУ. 2014. № 6; *Vdovin P.M., Kostenko V.A.* Algorithm for Resource Allocation in Data Centers with Independent Schedulers for Different Types of Resources // J. Computer and Systems Sciences International. 2014. № 6.

6. *Popa L., Kumar G., Chowdhuru M. et al.* FairCloud: Sharing the Network in Cloud Computing // Proc. ACM SIGCOMM'12, N.Y., 2012. P. 187–198.
7. *Wustenhoff E.* Service Level Agreement in the Data Center. Santa Clara CA: Sun BluePrints, 2012.
8. *Meng X., Pappas V., Zhang L.* Improving the Scalability of Data Center Networks with Traffic-aware Virtual Machine Placement // Proc. INFOCOM'10, IEEE. San Diego, 2010. P. 1–9.
9. *Cormen T.H., Leiserson C.E., Rivest R.L., Stein C.* Introduction to Algorithms. Cambridge MA: MIT Press and McGraw-Hill, 2001. P. 595–601.
10. *Al-Fares, Loukissas, Vahdat.* A Scalable, Commodity Data Center Network Architecture // Proc. SIGCOMM'08. Seatle, 2008.
11. *Zhao M., Figueiredo R. J.* Experimental Study of Virtual Machine Migration in Support of Reservation of Cluster Resources // Proc. VTDC'07, Article 5, N.Y.: ACM, 2007.
12. *Kostenko V., Plakunov A., Nikolaev A., Tabolin V., Smeliansky R., Shakhova M.* Selforganizing Cloud Platform // Proc. Intern. Conf. on Networks 2014: SDN&NFV. Moscow: IEEE Press, 2014.