# СОДЕРЖАНИЕ

**Номер 2, 2016**

## ТЕОРИЯ СИСТЕМ И ОБЩАЯ ТЕОРИЯ УПРАВЛЕНИЯ



## КОМПЬЮТЕРНЫЕ МЕТОДЫ



## СИСТЕМНЫЙ АНАЛИЗ И ИССЛЕДОВАНИЕ ОПЕРАЦИЙ



## НАВИГАЦИОННЫЕ СИСТЕМЫ



## СИСТЕМЫ УПРАВЛЕНИЯ ДВИЖУЩИМИСЯ ОБЪЕКТАМИ

Сдано в набор 04.12.2015 г. Подписано к печати 05.02.2016 г. Дата выхода в свет 23.04.2016 Формат $60 \times 88^{1}/_{8}$
Цифровая печать Усл. печ. л. 22.0 Усл. кр.-отт. 3.1 тыс. Уч.-изд. л. 22.1 Бум. л. 11.0
Тираж 136 экз. Зак. 74 Цена свободная

Учредители: Российская академия наук,
Государственный научно-исследовательский институт авиационных систем

Издатель: Российская академия наук. Издательство "Наука", 117997 Москва, Профсоюзная ул., 90
Оригинал-макет подготовлен МАИК "Наука/Интерпериодика"
Отпечатано в ППП «Типография "Наука"», 121099 Москва, Шубинский пер., 6

*ИЗВЕСТИЯ РАН. ТЕОРИЯ И СИСТЕМЫ УПРАВЛЕНИЯ, 2016, № 2, с. 51–63*

## КОМПЬЮТЕРНЫЕ МЕТОДЫ

*УДК 004.75*

# МЕТОД МУЛЬТИАГЕНТНОГО ДИСПЕТЧИРОВАНИЯ РЕСУРСОВ В ОБЛАЧНЫХ ВЫЧИСЛИТЕЛЬНЫХ СРЕДАХ

**© 2016 г. А. И. Каляев, И. А. Каляев**

*Таганрог, НИИ МВС ЮФУ*

*e-mail: kaliaev@mvs.sfedu.ru*

Поступила в редакцию 20.11.14 г., после доработки 30.06.15 г.

Описывается метод диспетчирования распределенных вычислительных ресурсов в облачных средах при решении пользовательских задач с помощью множества программных агентов, физически располагаемых на отдельных процессорных устройствах, подключенных к облачной инфраструктуре, и представляющих их интересы в процессе организации вычислений. К преимуществам предлагаемого подхода следует отнести: во-первых, возможность оперативного отслеживания с помощью агентов всех ресурсных изменений, происходящих с процессорными устройствами, и оперативной коррекции вычислительного процесса с учетом этих изменений, что в свою очередь позволяет использовать в составе облачной среды вычислительные ресурсы с динамически изменяемой производительностью (например, такие как персональные компьютеры частных владельцев), а во-вторых, снижение стоимости облачной инфраструктуры вследствие отсутствия необходимости введения в ее состав выделенных дорогостоящих узлов, выполняющих служебные функции.

**DOI:** 10.7868/S000233881601008X

**Введение.** В настоящее время уровень развития коммуникационных технологий открывает новые возможности организации распределенных вычислений на основе множества пространственно удаленных вычислительных ресурсов с помощью сервис-ориентированной инфраструктуры, обеспечивающей "вычисления по требованию". Эти возможности породили новую парадигму "облачных вычислений", т.е. крупномасштабных распределенных вычислений на основе пула абстрактных, виртуализованных, динамически перераспределяемых вычислительных ресурсов, предоставляемых по запросу внешним пользователям через Интернет по принципу "оплата по мере использования" [1]. При этом со стороны владельцев вычислительных ресурсов облачные вычисления – это получение дохода путем предоставления информационных ресурсов внешним пользователям, а со стороны пользователей облачные вычисления – это получение информационных ресурсов в виде услуги у внешнего поставщика, оплата за которую производится в зависимости от объема потребленных ресурсов согласно установленному тарифу.

Одним из основных преимуществ концепции облачных вычислений является возможность применения для решения пользовательских задач (ПЗ) самых разнообразных вычислительных ресурсов, подключенных к сетевой инфраструктуре. В этом плане очень перспективным является использование в облаке вычислительных ресурсов огромного количества компьютеров, находящихся в частном владении, которые зачастую простаивают без работы. Однако проблема заключается в том, что вычислительная производительность таких ресурсов может динамически изменяться в широком диапазоне в зависимости от действий их владельцев (например, владелец может неожиданно загрузить свой компьютер какой-либо дополнительной работой либо вообще отключить его от сети). При этом непредвиденные заранее изменения производительности облачных ресурсов, задействованных в решении некоторой ПЗ, могут приводить к сбоям вычислительного процесса и даже к его останову. Наиболее остро данная проблема проявляется при решении так называемых связных задач, отдельные части которых, решаемые на различных вычислительных ресурсах, входящих в облачную среду, информационно взаимосвязаны друг с другом [2].

Поэтому для организации работы облачной среды, использующей в своем составе вычислительные ресурсы с динамически изменяющейся производительностью, необходимо каким-то образом обеспечить постоянный мониторинг всех ресурсных изменений, происходящих в среде, и адаптацию вычислительного процесса к этим изменениям [3]. Указанная проблема может быть

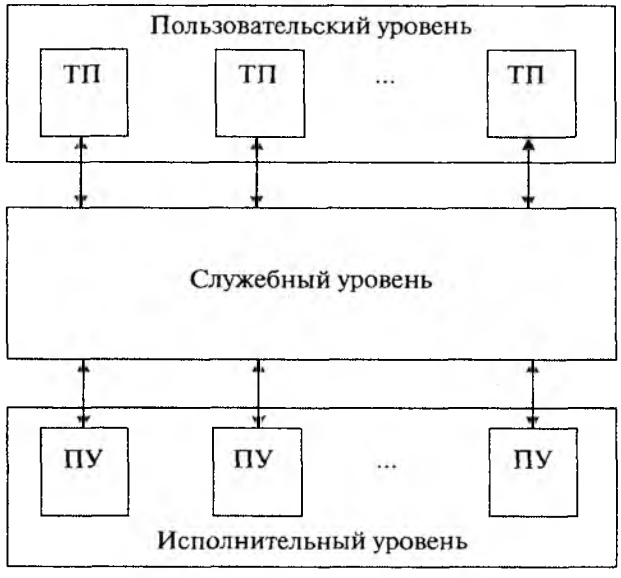


**Рис. 1.** Логические уровни облачной вычислительной среды

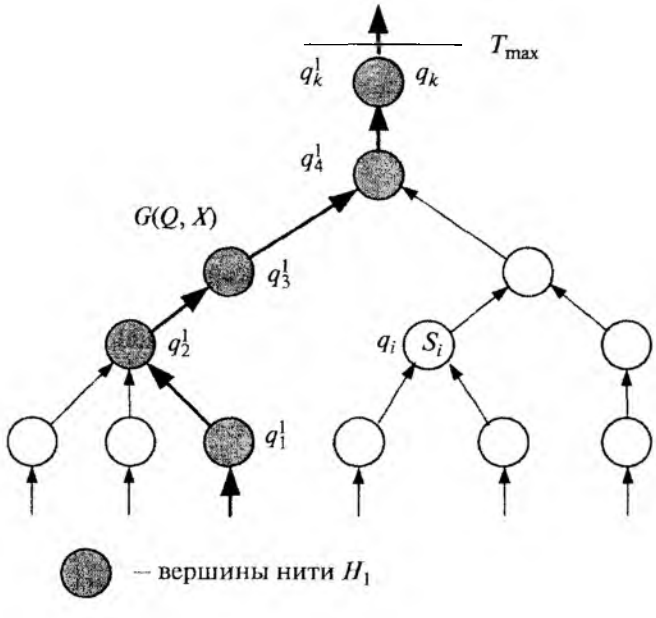


**Рис. 2.** Граф задачи $Z$

решена за счет разработки технологии мультиагентного диспетчирования облачных вычислительных ресурсов.

**1. Постановка задачи.** В укрупненном виде облачная инфраструктура включает три логических уровня (рис. 1):

*уровень пользователей* – программно-аппаратные средства (терминалы пользователей (ТП)), с помощью которых пользователь формируют свою задачу;

*уровень исполнителей* – множество распределенных в пространстве процессорных устройств (ПУ), объединенных некоторой коммуникационной сетью и предоставляющих свои вычислительные ресурсы для решения ПЗ;

*служебный уровень* – программные и аппаратные средства, служащие для организации (диспетчирования) процесса распределенных вычислений в облачной среде.

В общем случае ПЗ $Z$ должна быть представлена в виде ацикличного графового дерева $G(Q, X)$ (рис. 2), вершины которого соответствуют некоторым операциям (подзадачам), а дуги определя-

ют информационные обмены данными между ними. При этом каждой вершине (подзадаче) $q_i \in Q$ ставится в соответствие значение ее вычислительной трудоемкости $s_i$, которая оценивается либо максимальным количеством элементарных операций, выполняемых при решении данной подзадачи, либо на основе накопленной статистики. Кроме того, пользователь должен указать, к какому моменту времени $T_{\max}$ он желает получить решение задачи, а также количество бонусов (баллов) $O$, которое он готов заплатить за решение задачи.

Процесс решения ПЗ $Z$ в облачной вычислительной среде можно разбить на следующие основные этапы [4].

1. Пользователь, заинтересованный в решении некоторой задачи $Z$ в облаке, направляет ее формализованное описание (дескриптор) на служебный уровень системы.

2. С помощью служебного уровня из всего множества ПУ, входящих в облачную инфраструктуру, выделяется подмножество (сообщество) ПУ $C$ для решения поступившей ПЗ $Z$.

3. Вершины (подзадачи) $q_i \in Q$ графа задачи $G(Q, X)$ распределяются между ПУ, входящими в сообщество $C$, таким образом, чтобы обеспечить получение ее решения к требуемому моменту времени $T_{\max}$.

4. Каждое ПУ, входящее в сообщество $C$, решает закрепленную за ним часть ПЗ $Z$. При этом обмен данными между ПУ осуществляется в соответствии с топологией графа задачи $G(Q, X)$.

5. После завершения процесса решения задачи $Z$ в сообществе ПУ ее результаты отправляются на ТП, а ПУ, участвовавшим в ее решении, начисляются бонусы (баллы) пропорционально объему выполненных ими вычислений.

Очевидно, что основную роль в процессе облачных вычислений играет служебный уровень и от того, как он будет организован, в сильной степени будут зависеть технические, а также экономические характеристики облачной вычислительной среды. В настоящее время служебный уровень, как правило, имеет централизованную организацию, когда перечисленные выше функции возлагаются на специально выделенный сервер или, иначе говоря, – центральный диспетчер [5]. Однако такая централизованная организация диспетчера затрудняет использование в облачной среде ПУ с динамически изменяющейся производительностью, например компьютеры частных владельцев. Действительно, если производительность каждого ПУ, входящих в сообщество по решению ПЗ, может меняться заранее непредсказуемым образом, то в какой-то момент времени может оказаться, что решение ПЗ к требуемому моменту времени данным сообществом невозможно. Поэтому служебный уровень должен постоянно мониторить текущую производительность всех ПУ, входящих в сообщество, и в случае, если их ресурсные изменения приводят к невозможности решения ПЗ к требуемому моменту времени, осуществлять процедуру корректировки состава сообщества. В то же время организация такого мониторинга в облачной среде, включающей в свой состав большое число распределенных в пространстве ПУ, с помощью одного центрального диспетчера крайне затруднительна. Использование в качестве диспетчера облачной среды многоуровневых деревьев выделенных серверов [6] также не решает проблему, поскольку при этом, с одной стороны, резко усложняется организация служебного уровня системы, а с другой – возникает не менее сложная проблема синхронизации работы множества служебных серверов. Поэтому современные облачные сервисы рассчитаны, как правило, на использование вычислительных ресурсов, производительность которых заранее известна и не может существенно изменяться во времени [7].

Проблема организации облачной среды на базе ПУ с динамически изменяемой производительностью может быть решена путем использования принципов децентрализованного мультиагентного диспетчирования [4]. При этом основными активными элементами системы предлагается сделать программных агентов [8], физически располагаемых на каждом процессорном устройстве (ПУ), которые входят в состав облачной вычислительной среды, и представляющих их "интересы" в процессе диспетчирования. Основным преимуществом такого мультиагентного подхода к организации облачной среды является то, что каждый такой агент может осуществлять постоянный мониторинг текущей производительности "своего" ПУ и оперативно использовать эти данные для корректировки вычислительного процесса. Однако использование такого подхода требует разработки новых методов, обеспечивающих реализацию процедур создания сообщества ПУ для решения ПЗ, распределения подзадач между ними и динамической корректировки состава сообщества в случае недопустимых ресурсных изменений с помощью множества распределенных в пространстве программных агентов. Именно разработке таких методов и посвящена настоящая статья.

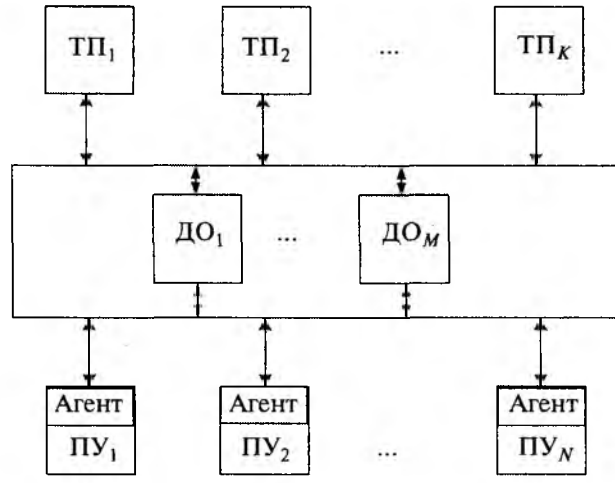


**Рис. 3.** Структура облачной среды с мультиагентным диспетчером

При этом с целью упрощения дальнейших построений введем ряд допущений. Будем считать, что:

1) в облаке достаточно ПУ для решения пользовательской задачи $Z$ за отведенное время;

2) любая из подзадач $q_i \in Q$ ПЗ $Z$ может быть решена любым ПУ, входящим в облачную инфраструктуру;

3) агенты ведут "честную игру", т.е. все данные, передаваемые агентами, достоверны.

**2. Принципы мультиагентного диспетчирования ресурсов облачной вычислительной среды.** Первый вопрос, который возникает при организации мультиагентного диспетчирования ресурсов облачной среды – каким образом пользователи и агенты должны взаимодействовать друг с другом? При централизованной организации диспетчера облачной среды такое взаимодействие осуществляется посредством служебного уровня. При децентрализованной, мультиагентной организации диспетчера взаимодействие пользователей и агентов может осуществляться посредством некоторого множества специальных пассивных узлов, выполняющих функции досок объявлений (ДО), на которых пользователи размещают свои задачи. Агенты должны периодически опрашивать ДО с целью загрузки своего ПУ решением той или иной ПЗ. При этом если задача не может быть решена с помощью одного ПУ, то агенты формируют некоторую виртуальную организацию – сообщество [9], состоящее из множества распределенных ПУ, общей целью которых является решение ПЗ к требуемому моменту времени. При этом каждый агент должен быть заинтересован в максимальной загрузке своего ПУ, для чего должен быть предусмотрен некоторый механизм вознаграждения за предоставленные вычислительные ресурсы, величина которого определяется общей стоимостью решения задачи, установленной пользователем, и вкладом конкретного ПУ в ее решение. С другой стороны, также должен быть предусмотрен механизм наказания агента за невыполнение к заданному сроку принятой на себя работы.

Основываясь на приведенных выше соображениях, можно предложить следующие основные принципы мультиагентного диспетчирования работы облачной вычислительной среды (рис. 3).

1. С помощью ТП пользователь формирует свою задачу $Z$ в виде графа $G(Q, X)$, определяет требуемый момент времени $T_{\max}$, к которому ее решение должно быть получено, а также величину бонусов (баллов) $O$, которую он готов заплатить за ее решение. Дескриптор представленной таким образом задачи $Z$ отправляется на одну из досок объявлений.

2. Агент ПУ, не задействованного при решении других задач, опрашивает доски объявлений в поисках работы для "своего" ПУ. В случае обнаружения на ДО ПЗ $Z$ агент оценивает целесообразность участия в ее решении. Для этого он определяет соотношение величины бонусов, предлагаемых пользователем за решение данной задачи, к ее суммарной вычислительной трудоемкости, и если это соотношение не ниже некоторого установленного для него порога, то принимает

решение о возможности вхождения в состав сообщества по ее решению. В противном случае он переходит к дальнейшему опросу ДО.

3 Агент выделяет наиболее трудоемкий фрагмент задачи, не закрепленный за другими агентами, и оценивает возможность его выполнения к заданному моменту времени с помощью "своего" ПУ. Если производительности ПУ хватает, чтобы обеспечить решение выделенного фрагмента за отведенное время, то агент вступает в сообщество $C$ по решению данной задачи.

4. Агент контролирует ход решения принятого к исполнению фрагмента задачи, периодически оценивая время получения конечного результата с учетом текущей производительности "его" ПУ. В случае, если текущей производительности "его" ПУ оказывается недостаточно для решения оставшейся части фрагмента задачи к заданному моменту времени, то агент сообщает об этом на доску объявлений и выходит из состава сообщества. При этом оставшаяся нерешенной часть данного фрагмента задачи вновь "вывешивается" на доске объявлений, а агент, отказавшийся от ее решения, наказывается штрафными баллами.

5. После успешного завершения решения принятого на себя фрагмента задачи агент отправляет полученные результаты решения на ТП, а на доску объявлений – сообщение о завершении своей работы. При этом если решение получено к заданному моменту времени, то ему начисляются премиальные баллы, пропорциональные объему выполненных им вычислений. Если же решение было получено агентом позже установленного времени, то агент наказывается штрафными баллами. После этого агент снова переходит в режим опроса ДО с целью поиска новой работы для своего ПУ.

Основным преимуществом предлагаемого мультиагентного подхода к диспетчированию ресурсов в облачной среде является то, что вычислительный процесс будет постоянно контролироваться каждым из агентов и, соответственно, оперативно адаптироваться ко всем ресурсным изменениям, происходящим с "его" ПУ. Это в свою очередь позволяет строить облачную вычислительную среду на базе процессорных устройств, производительность которых может изменяться в широких пределах непредсказуемым образом, например компьютеров частных владельцев. Кроме того, по сравнению с классической, централизованной организацией диспетчера облачной среды в данном случае существенно упрощаются требования к служебным серверам (доскам объявлений), что позволяет существенно снизить накладные расходы на организацию облачной среды и, как следствие, снизить стоимость облачных вычислений, а также упростить процесс масштабирования облачной среды.

**3. Алгоритм работы программного агента.** Для того, чтобы задача $Z$ была "понятна" агентам облачной среды, ее необходимо представить в формализованном виде – в виде дескриптора. Дескриптор задачи $Z$ должен включать следующие данные:

идентификатор задачи – уникальный номер задачи;

идентификатор пользователя – уникальный номер ТП, откуда задача поступила в облако;

граф задачи $G(Q, X)$ (например, представленный в виде таблицы смежности);

– вычислительные трудоемкости $s_i$ отдельных подзадач $q_i \in Q$, а также суммарную трудоемкость задачи

$$S = \sum_{i=1}^{|Q|} s_i;$$

список ПУ, входящих в сообщество по решению данной задачи (заполняется по мере формирования сообщества $C$);

момент времени $T_{\max}$, к которому пользователь хочет получить решение задачи;

величину бонусов (баллов) $O$, предлагаемую пользователем за успешное решение задачи к моменту времени $T_{\max}$.

Сформированный таким образом дескриптор задачи размещается на одной из досок объявлений.

Как показано выше, основной функцией агента является поиск работы для своего ПУ. Для этого агент $A$ опрашивает доски объявлений с целью поиска нерешенных задач, и если такая задача $Z$ обнаруживается, то агент оценивает целесообразность своего участия в сообществе $C$ по ее решению.

Целесообразность своего участия в решении задачи агент может оценить путем определения величины $P = O/S$ относительной стоимости вычислений, предлагаемой пользователем. Если величина $P$ ниже некоторого порога $P^*$, установленного, например, владельцем данного ПУ, то участие агента в решении данной задачи считается экономически невыгодным. Иначе агент делает попытку войти в сообщество $C$ по ее решению.

Для этого агент $A$ выделяет в графе задачи $G(Q, X)$ некоторую последовательность вершин (подзадач) $H_j = \langle q_1^j, q_2^j, \ldots, q_k^j \rangle$, которая может быть решена с помощью его ПУ за отведенное время (рис. 2). В дальнейшем такую последовательность подзадач будем называть "нитью". Поскольку каждая последующая подзадача нити в качестве исходных данных использует результаты решения предыдущей подзадачи, то решение подзадач $q_1^j, q_2^j, \ldots, q_k^j$, входящих в нить $H_j$, может осуществляться только последовательно. Такое последовательное решение подзадач, входящих в "нить", целесообразно осуществлять с помощью одного и того же ПУ, поскольку при этом отпадает необходимость передачи промежуточных данных по сети. При этом под длиной нити будем понимать суммарную вычислительную трудоемкость $S_j$ подмножества принадлежащих ей вершин (подзадач) графа $G(Q, X)$, т.е.

$$S_j = \sum_{k=1}^{k} s_i^j,$$

где $s_i^j$ – вычислительная трудоемкость подзадачи $q_i^j \in H_j$, $i = \overline{1,k}$.

Изначально, в момент размещения дескриптора задачи $Z$ на ДО, список ПУ, участвующих в сообществе $C$ по ее решению, пуст, что означает, что никакая из нитей графа $G(Q, X)$ пока не принята к исполнению никаким ПУ. Поэтому первый из агентов $A_1$, обнаруживших данную задачу, выделяет в графе задачи $G(Q, X)$ наиболее длинную нить $H_1 = \|q_1^1, q_2^1, \ldots, q_k^1\|$, которая, очевидно, начинается на одной из входных вершин графа, а заканчивается на выходной вершине графа $G(Q, X)$, т.е. $q_k^1 = q_k$ (рис. 2). Для поиска самой длинной нити агент может использовать один из алгоритмов поиска максимального пути в графе, например волновой алгоритм [10].

Поскольку найденная таким образом нить определяет самую вычислительно трудоемкую последовательность подзадач в задаче, то решение этой последовательности, очевидно, должно быть завершено к заданному пользователем моменту решения всей задачи $T_{\max}$, который содержится в ее дескрипторе. На основании данных о времени $T_{\max}$ и суммарной вычислительной трудоемкости подзадач $S_1$, входящих в выделенную нить $H_1$, агент определяет производительность ПУ, требуемую для исполнения данной нити к моменту $T_{\max}$, как

$$V_1^T = \frac{S_1}{T_{\max} - T_{\text{тек}}},$$

где $V_1^T$ – требуемая производительность ПУ; $T_{\text{тек}}$ – текущий момент времени; после чего сравнивает ее с текущей производительностью $V_1$ своего $\text{ПУ}_1$.

Если оказывается, что $V_1^T > V_1$, то это означает, что агент не может обеспечить решение данной нити $H_1$ за отведенное время и поэтому он вновь переходит в режим поиска.

Если же условие $V_1^T \le V_1$ выполняется, то в этом случае агент принимает на себя исполнение выделенной нити $H_1$. При этом агент $A_1$ осуществляет модификацию дескриптора задачи $Z$ на ДО, а именно: идентификатор его $\text{ПУ}_1$ записывается в список членов сообщества $C$ по ее решению, а вершины, входящие в нить $H_1$, исключаются из графа $G(Q, X)$ (т.е. в таблице смежности вычеркиваются соответствующие строки и столбцы), в результате чего формируется новый граф $G_1(Q_1, X_1) = G(Q, X)/H_1$ (рис. 4).

Кроме того, всем вершинам графа $G_1(Q_1, X_1)$, инцидентным вершинам нити $H_1$, необходимо приписать максимально допустимое время их исполнения, которое определяется исходя из следующих соображений (рис. 4). Допустим, что некоторая вершина $q_f^2$ графа $G_1(Q_1, X_1)$ инцидентна вершине $q_b^1$, принадлежащей нити $H_1$. Поскольку для решения подзадачи $q_b^1$ необходимы данные,

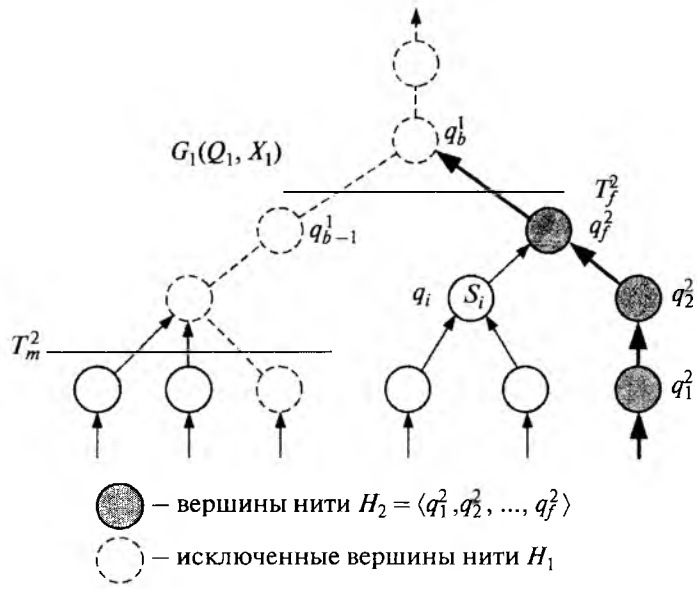


**Рис. 4.** Граф задачи, модифицированный агентом $A_1$

получаемые как в результате решения подзадачи $q_f^2$ графа $G_1(Q_1, X_1)$, так и в результате решения последовательности подзадач $H_1^{b-1} = \langle q_1^1, q_2^1, ..., q_{b-1}^1 \rangle$ нити $H_1$, предшествующих подзадаче $q_b^1$, то очевидно, что результаты решения подзадачи $q_f^2$ должны поступать не позже, чем данные, получаемые в результате решения последовательности подзадач $H_1^{b-1} = \langle q_1^1, q_2^1, ..., q_{b-1}^1 \rangle$ нити $H_1$. В противном случае ПУ$_1$ не успеет закончить решение нити $H_1$ к требуемому моменту времени $T_{\max}$. Поэтому максимально допустимый момент времени решения подзадачи $q_f^2$ графа $G_1(Q_1, X_1)$ определяется по формуле

$$T_f^2 = \frac{1}{V_T^1} \sum_{l=1}^{b-1} s_l^1,$$

где $b \le k$, $s_l^1$ – вычислительная трудоемкость вершин $q_l^1$, $l = \overline{1, b-1}$, нити $H_1$, предшествующих вершине $q_b^1$, которая в свою очередь инцидентна вершине $q_f^2$ (рис. 4).

Кроме того, вершине $q_f^2$ графа $G_1(Q_1, X_1)$ приписывается идентификатор ПУ$_1$, на который результаты исполнения подзадачи $q_f^2$ должны быть направлены.

Аналогичным образом определяются требуемые моменты $T_m^2$ исполнения всех остальных вершин графа $G_1(Q_1, X_1)$, инцидентных вершинам нити $H_1$ (рис. 4).

Поскольку агент заинтересован в максимальной загрузке своего ПУ, то далее он предпринимает попытку взять на себя дополнительную работу, т.е. исполнение следующей нити задачи $Z$. Для этого в графе $G_1(Q_1, X_1)$ он выделяет наиболее длинную нить $H_2 = \langle q_1^2, q_2^2, ..., q_f^2 \rangle$ (рис. 4) и определяет величину производительности $V_2^T$, необходимую для ее исполнения к моменту $T_f^2$, приписанному ее конечной вершине $q_f^2$, как

$$V_2^T = \frac{1}{T_f^2 - T_{\text{тек}}} \sum_{i=1}^{f} s_i^2.$$

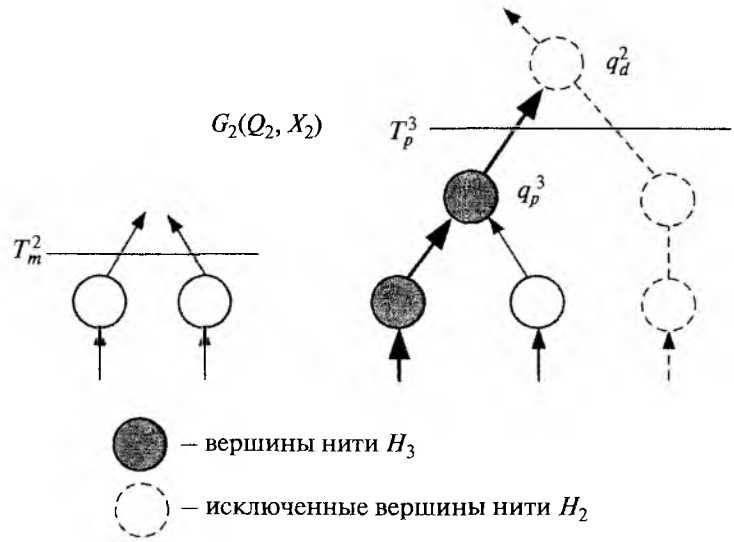


**Рис. 5.** Граф задачи, модифицированный агентом $A_2$

Если $V_2^T > V_1 - V_1^T$, где $V_1$ – производительность ПУ$_1$, $V_1^T$ – производительность, требуемая для исполнения нити $H_1$ к заданному моменту времени $T_f^2$, то это означает, что оставшейся производительности ПУ$_1$ не хватает для одновременного исполнения нитей $H_1$ и $H_2$, и поэтому агент $A_1$ переходит к процедуре реализации ранее принятой на себя нити $H_1$ задачи $Z$. Иначе агент $A_1$ дополнительно к нити $H_1$ принимает на себя решение нити $H_2$ и осуществляет на ДО модификацию дескриптора задачи: вершины, входящие в нить $H_2$, исключаются из графа $G_1(Q_1, X_1)$, в результате чего формируется новый граф $G_2(Q_2, X_2)$ (рис. 5); всем вершинам нового графа $G_2(Q_2, X_2)$, инцидентным вершинам нити $H_2$, приписывается идентификатор ПУ$_1$, куда результаты исполнения подзадачи $q_p^3$ должны быть направлены, а также необходимое время их исполнения, определяемое как

$$T_p^3 = \frac{\sum_{l=1}^{d-1} s_l^2}{V_2^T},$$

где $s_l^2$, $l = \overline{1, d-1}$, – вычислительная трудоемкость вершин $q_1^2, q_2^2, \ldots, q_{d-1}^2$, предшествующих в нити $H_2$ вершине $q_d^2$, которая в свою очередь инцидентна вершине $q_p^3$ графа $G_2(Q_2, X_2)$.

Процесс загрузки агентом $A_1$ "своего" ПУ$_1$ работой продолжается до тех пор, пока не окажется, что либо оставшейся производительности ПУ$_1$ уже не хватает для исполнения очередной нити $H_j$ графа $G_j(Q_j, X_j)$ за отведенное время $T_p^j$, либо все нити задачи уже разобраны, т.е. в результате исключения очередной нити из графа задачи он становится пустым, т.е. $Q_j \in \varnothing$. После этого ПУ$_1$ считывает из дескриптора задачи $Z$ идентификатор ТП, а также идентификаторы ПУ, на которые он должен передать результаты принятых к исполнению нитей. Отметим, что для первого агента $A_1$, обнаружившего задачу $Z$, результаты решения принятых к исполнению нитей должны быть направлены на ТП, поскольку агент $A_1$ исполняет наиболее длинную нить исходного графа $G(Q, X)$, оканчивающуюся конечной (результирующей) вершиной $q_k$ (рис. 3). Далее агент $A_1$ обращается к ТП, откуда получает исполняемые модули подзадач, входящих в исполняемые им нити, а также исходные данные, необходимые для их решения, после чего запускает их решение на своем ПУ$_1$.

Если после того, как агент $A_1$ завершил загрузку своего ПУ$_1$, граф $G_j(Q_j, X_j)$ задачи $Z$, размещенный в ее дескрипторе на ДО, еще не пустой, то процесс создания сообщества $C$ для решения данной задачи продолжается далее. Допустим, что через какое-то время другой агент $A_2$ свободного ПУ$_2$ обнаруживает на ДО дескриптор задачи $Z$. Первым делом агент $A_2$ оценивает экономическую целесообразность своего участия в решении задачи – он определяет значение относительной стоимости вычислений данной задачи как $P = O/S$ и если стоимость его устраивает, то предпринимает попытку взять на себя исполнение оставшихся нитей данной задачи. Для этого в графе $G_j(Q_j, X_j)$ (который остался в дескрипторе после всех модификаций, произведенных агентом $A_1$) агент $A_2$ выделяет наиболее длинную нить $H_j$ и определяет величину производительности, необходимую для ее исполнения к моменту $T_p^j$, приписанному ее конечной вершине $q_p^j$, как

$$V_j^T = \frac{\sum_{i=1}^{p} s_i^j}{T_p^j - T_{\text{тек}}},$$

где $s_i^j$, $i = \overline{1, p}$, – трудоемкость подзадач, входящих в нить $H_j$.

Если $V_j^T > V_2$, где $V_2$ – производительность ПУ$_2$, то агент $A_2$ переходит обратно в режим поиска задачи.

В случае же если $V_j^T \le V_2$, то агент $A_2$ принимает на себя исполнение данной нити $H_j$ и осуществляет очередную модификацию дескриптора задачи $Z$: вершины, входящие в нить $H_j$, исключаются из графа $G_j(Q_j, X_j)$, в результате чего образуется новый граф $G_{j+1}(Q_{j+1}, X_{j+1})$; вершинам этого графа, инцидентным вершинам нити $H_j$, приписывается необходимое время их исполнения, определяемое временем исполнения соответствующих вершин нити $H_j$, а также идентификатор ПУ$_2$, куда результаты исполнения этих подзадач должны быть направлены. После этого агент $A_2$ осуществляет попытку принять на себя исполнение следующей нити $H_{j+1}$ задачи $Z$, и если это невозможно, то переходит к исполнению принятых на себя нитей, получая их исполняемые модули и исходные данные с ТП.

Далее в процесс распределения нитей задачи $Z$ включается следующий агент $A_3$, обнаруживший ее дескриптор на ДО, и т.д. до тех пор, пока не окажется, что граф задачи стал пустым, т.е. все подзадачи разобраны агентами, вошедшими в сообщество $C$, идентификаторы ПУ которых записаны в дескрипторе задачи.

Поскольку ресурсные параметры ПУ, участвующих в решении задачи $Z$, могут динамически изменяться во времени вплоть до полного отключения ПУ от сети, то необходимо предусмотреть механизм коррекции состава сообщества в подобных случаях. Для этого агент $A_j$, участвующий в сообществе $C$ по решению задачи $Z$, должен периодически контролировать ход вычислительного процесса в своем ПУ$_j$. Для этого после выполнения очередной подзадачи $q_r^n$, входящей в нить $H_n$, реализуемую ПУ$_j$, агент $A_j$ должен оценивать момент времени $T_p^n$, к которому "его" ПУ$_j$ может завершить выполнение нити $H_n$, как

$$T_p^n = \frac{S_n^r}{V_j},$$

где $V_j$ – текущая производительность ПУ$_j$;

$S_n^r = \sum_{i=r}^{l} s_i$ – суммарная трудоемкость оставшихся (еще не выполненных) подзадач $q_{r+1}^n, q_{r+2}^n, \ldots, q_l^n$ нити $H_n$;

и сравнить полученный таким образом момент времени $T_p^n$ с требуемым моментом времени $T_n$ завершения нити $H_n$.

Если $T_p^n \le T_n$, то это означает, что вычислительный процесс удовлетворяет плану. В противном случае агент $A_j$ должен известить ДО о том, что произошло отставание от графика вычислительного процесса и он не может обеспечить решение принятой на себя нити $H_n$ к заданному моменту времени $T_n$. При этом полученные промежуточные данные исполнения нити $H_n$ отправляются агентом $A_j$ на ТП, а дескриптор задачи на ДО модифицируется следующим образом: ПУ$_j$ исключается из списка членов сообщества; оставшиеся нереализованными вершины нити $H_n$ возвращаются в граф задачи, причем конечной вершине этой нити вновь приписывается значение момента времени $T_n$, к которому необходимо получить результат ее решения, а также идентификатор ПУ, куда этот результат необходимо передать. Модифицированный таким образом дескриптор задачи вновь выставляется на ДО для поиска новых участников сообщества, которые смогли бы обеспечить решение оставшейся части задачи к требуемому моменту времени.

Если ПУ$_j$ успешно выполнил все принятые на себя нити задачи $Z$ к заданным моментам времени, то агент $A_j$ информирует об этом доску объявлений. При этом идентификатор данного ПУ$_j$ исключается из списка участников сообщества в дескрипторе задачи, а агенту $A_j$ начисляются премиальные баллы за выполненную работу, величина которой определяется следующим образом: $O_j = O \cdot S_j / S$, где $O$ – бонусы, гарантируемые пользователем за решение его задачи $Z$; $S_j$ – суммарная трудоемкость подзадач, выполненных ПУ$_j$; $S$ – суммарная трудоемкость всей задачи $Z$. В случае же, если ПУ$_j$ не обеспечил выполнение принятых на себя частей задачи $Z$ к требуемым моментам времени или отказался от их дальнейшего решения в связи с невозможностью удовлетворения заданным временным ограничениям, то агент $A_j$ наказывается штрафными баллами, величина которых должна зависеть от времени задержки.

Процесс решения задачи $Z$ в облачной вычислительной среде продолжается до тех пор, пока не окажется, что список агентов-участников сообщества по ее решению пуст, а также пуст модифицированный граф задачи $Z$. Это означает, что все нити задачи успешно выполнены. После этого задача снимается с ДО, а на ТП отправляется сообщение о успешном завершении процесса ее решения, после чего пользователь переводит оплату (премиальные баллы) $O_j$ всем ПУ$_j$, принявшим участие в решении данной задачи в зависимости от объема выполненной ими работы (суммарной сложности решенных подзадач), а также соблюдения временных ограничений.

Описанному выше процессу отвечает следующий формальный алгоритм работы агента $A$, представляющего интересы процессорного устройства ПУ в облачной среде.

1. Агент $A$ свободного ПУ опрашивает ДО.

2. Если агент $A$ обнаруживает на ДО дескриптор задачи $Z$, то он анализирует граф задачи $G_i(Q_i, X_i)$. Если $G_j(Q_j, X_j) = \varnothing$, то переход к п. 1.

3. Агент оценивает экономическую целесообразность своего участия в решении задачи $Z$, сравнивая предлагаемую пользователем относительную стоимость вычислений $P = O/S$ с порогом $P^*$, установленным владельцем его ПУ. Если $P < P^*$, то переход к п. 1.

4. Агент $A$ выделяет в графе $G_i(Q_i, X_i)$ наиболее длинную нить $H_j = \left\langle q_1^j, q_2^j, \ldots, q_l^j \right\rangle$ и рассчитывает производительность ПУ, необходимую для ее решения к моменту $T_j^i$, приписанному конечной вершине $q_l^j$ данной нити (если $H_j = H_1$, то $T_j^i = T_{\max}$), как

$$V^T = \frac{\sum_{f=1}^{l} s_f^j}{T_j^i - T_{\text{тек}}},$$

где $T_{\text{тек}}$ – текущий момент времени; $s_f^j$ – вычислительная трудоемкость, приписанная $f$-й вершине нити $H_j$.

5. Если $V^T > V$, где $V$ – текущая производительность ПУ, то переход к п. 1.

6. Агент принимает на себя выполнение нити $H_j$, для чего модифицирует дескриптор задачи: записывает в список участников сообщества идентификатор своего ПУ, формирует новый граф задачи $G_{i+1}(Q_{i+1}, X_{i+1})$ путем исключения вершин нити $H_j$, т.е. $G_{i+1}(Q_{i+1}, X_{i+1}) = G_i(Q_i, X_i)/H_j$; вершинам $q_b^{i+1}$ нового графа $G_{i+1}(Q_{i+1}, X_{i+1})$, инцидентным вершинам нити $H_j$, приписывает идентификатор ПУ агента $A$, куда необходимо передать результаты исполнения вершины $q_b^{i+1}$, а также необходимое время их исполнения, определяемое как

$$T_b^{i+1} = \frac{\sum_{b=1}^{d-1} s_b^j}{V^T},$$

где $d \le l$ – номер вершины нити $H_j$, инцидентной вершине $q_b^{i+1}$ графа $G_{i+1}(Q_{i+1}, X_{i+1})$, $s_l^j$ – вычислительная трудоемкость $l$-й вершины нити $H_j$; $V^T$ – требуемая производительность ПУ, необходимая для исполнения нити $H_j$ к заданному моменту $T_j^i$.

7. Агент $A$ оценивает свободную производительность $V = V - V^T$, оставшуюся после закрепления за ним нити $H_j$, $G_i(Q_i, X_i) = G_{i+1}(Q_{i+1}, X_{i+1})$, переход к п. 4.

8. Агент $A$ считывает из дескриптора задачи $Z$ идентификатор ТП, а также идентификаторы других ПУ, куда должны быть переданы результаты решения принятых им к исполнению нитей.

9. Агент $A$ обращается к ТП, получает оттуда исполняемые файлы подзадач, входящих в принятые им к исполнению нити, а также исходные данные для их решения.

10. Агент $A$ запускает решение очередной подзадачи $q_f^j$, $f = \overline{1,l}$, исполняемой нити $H_j$ на своем ПУ.

11. Если после завершения решения подзадачи $q_f^j$ оказывается, что список подзадач, исполняемых агентом $A$, пуст, т.е. все взятые нити успешно исполнены, то переход к п. 15.

12. Агент $A$ оценивает момент времени, к которому он может обеспечить исполнение оставшихся подзадач $\langle q_{f+1}^j, q_{f+2}^j, \ldots, q_l^j \rangle$ нити $H_j$, как

$$T_p^j = \frac{\sum_{d=f+1}^{l} s_d^j}{V},$$

где $V$ – текущая производительность его ПУ.

13. Если $T_p^j < T^j$, где $T^j$ – требуемый момент времени окончания исполнения нити $H_j$, то переход к п. 10.

14. Агент $A$ отправляет результаты решения подзадачи $q_f^j$ на ТП и осуществляет модификацию дескриптора задачи $Z$: исключает идентификатор своего ПУ из списка сообщества; добавляет неисполненные вершины $q_{f+1}^j, q_{f+2}^j, \ldots, q_l^j$ всех принятых им к исполнению нитей в граф $G_i(Q_i, X_i)$, а также приписывает крайним вершинам этих нитей моменты времени $T^j$, к которому они должны быть исполнены, и идентификаторы ПУ, куда необходимо передать результаты их решения. Агенту $A$ начисляются штрафные баллы, пропорциональные времени задержки $\delta T = T_p^j - T^j$. Переход к п. 1.

15. Агент $A$ сообщает на ДО об успешном завершении решения принятых к исполнению нитей задачи $Z$. При этом идентификатор его ПУ исключается из списка членов сообщества по решению данной задачи, а агенту $A$ начисляется премиальные баллы $O_j = O \cdot S_j / S$, где $S_j$ – суммарная вычислительная трудоемкость подзадач, выполненных его ПУ; $S$ – суммарная трудоемкость всей задачи $Z$. Переход к п. 1.

**4. Результаты экспериментальных исследований.** С целью исследования работоспособности и эффективности предложенного выше подхода была разработана программная модель облачной

вычислительной среды с мультиагентным диспетчером [11]. Программная модель обеспечивает возможность исследования работы облачной среды при различных значениях таких исходных параметров как:

количество ПУ, подключенных к облаку (до 1000 шт.), их текущая производительность;

вычислительная трудоемкость и сложность (количество подзадач) задач, частота их появления;

требуемое время решения задач и величина оплаты за их успешное решение.

При этом в качестве критериев эффективности работы облачной вычислительной среды были приняты:

коэффициент полезной загрузки (КПЗ) процессорных устройств, подключенных к облаку – отношение количества процессорного времени ПУ, потраченного на решение задач, к общему времени их работы в облаке;

коэффициент гарантированности решения (КГР) задачи – отношение количества задач, решенных за требуемое время, к общему количеству задач, поступивших в облако для решения.

Результаты серии экспериментов, проведенных при различных значениях исходных параметров программной модели, показали, что: значение КПЗ не опускается ниже 75%, а его среднее значение составляет 84%; значение КГР не опускается ниже 91%, а среднее значение составляет 97%.

**Заключение.** В работе предложены обобщенные принципы организации распределенных вычислений в облачной среде с помощью мультиагентного диспетчера, а также базовый алгоритм функционирования программного агента процессорного устройства, подключенного к облачной инфраструктуре. Реализация предлагаемого подхода позволяет:

повысить адаптивность вычислительного процесса ко всем ресурсным изменениям, происходящим в облачной среде, и, как следствие, обеспечить возможность использования в облаке вычислительных ресурсов с динамически изменяемой производительностью, в частности простаивающих ресурсов компьютеров частных владельцев;

сократить накладные расходы на организацию облачной инфраструктуры за счет исключения дорогостоящих служебных серверов и, как следствие, снизить стоимость облачных вычислений;

обеспечить возможность неограниченного наращивания (масштабируемости) облачного вычислительного ресурса.

В то же время следует отметить, что за рамками работы остался ряд вопросов, связанных, в частности, с обеспечением отказоустойчивости облачных вычислений в условиях возможных отказов отдельных ПУ (или потери связи с ними), наличия неисправных ПУ, пересылающих недостоверные данные, и т.п. Эти проблемы могут быть, по-видимому, устранены путем дублирования решения отдельных частей задачи на различных ПУ, что, однако, ведет к увеличению стоимости облачных вычислений. Кроме того, не рассмотрены вопросы, связанные с возможным возникновением коллизий из-за устаревания данных, которыми оперирует агент при отправлении на ДО своего сообщения. Все эти вопросы требуют дополнительного исследования.

В заключение следует отметить, что предлагаемый подход может быть использован не только при создании коммерческих облачных вычислительных сред, но и облачных сред крупных отраслевых предприятий, а также информационно-вычислительных и управляющих сред государственного уровня. Кроме того, предложенные базовые принципы мультиагентного диспетчирования ресурсов могут быть использованы для решения задач других предметных областей, например, задач управления группами роботов, участвующих в сборочном производстве, создания сенсорных полей датчиков, осуществляющих сбор, обработку и предоставление заинтересованным пользователям информации от распределенных в пространстве источников, и т.п.

## СПИСОК ЛИТЕРАТУРЫ

1. *Mohamed A.* A History of Cloud Computing. URL: http://www.computerweekly.com/feature/A-history-of-cloud-computing.
2. *Каляев А.И.* Метод и алгоритмы адаптивной организации распределенных вычислений в децентрализованной GRID // Вестник компьютерных и информационных технологий. 2012. № 4. С. 28–33.

3. *Holland J.* Adaptation in Natural and Artificial Systems. Ann Arbor. Michigan: Univ. of Michigan Press, 1975.
4. *Kalyaev A.I., Korovin Y.S.* Adaptive Multiagent Organization of the Distributed Computations // AASRI Procedia. 2014. V. 6. P. 49–58.
5. *Коваленко В.Н., Корягин Д.А.* Вычислительная инфраструктура будущего // Открытые системы. 1999. № 11–12. С. 45–52.
6. *Berman F.* Adaptive Computing on the Grid Using AppLeS. URL: http://walfredo.dsc.ufcg.edu.br/papers/l0369.pdf.
7. *Lee G.* Cloud Computing: Principles, Systems and Applications // Computer Communications and Networks. Springer, 2010. 379 p.
8. *Тарасов В.Б.* От многоагентных систем к интеллектуальным организациям. М.: Эдиториал УРСС, 2002.
9. *Каляев А.И.* Децентрализованная организация диспетчера GRID на базе сообществ агентов // Изв. Южного федерального ун-та. Технические науки. 2011. Т. 121. № 8. С. 230–238.
10. *Router M.* Lee Algorithm URL:http://www.eecs.northwestern.edu/~haizhou/357/lec6.pdf.
11. *Kalyaev A.I.* Multiagent Approach for Building Distributed Adaptive Computing System // Procedia Computer Science. 2013. V. 8. P. 2193–2202.