УДК 519.687

**Полежаев П.Н.**
Оренбургский государственный университет
E-mail: peter.polezhaev@gmail.com

# МАТЕМАТИЧЕСКАЯ МОДЕЛЬ РАСПРЕДЕЛЕННОГО ВЫЧИСЛИТЕЛЬНОГО ЦЕНТРА ОБРАБОТКИ ДАННЫХ С ПРОГРАММНО-КОНФИГУРИРУЕМЫМИ СЕТЯМИ ЕГО СЕГМЕНТОВ

**В статье представлена математическая модель распределенного вычислительного центра обработки данных. Она является теоретической основой для разрабатываемого центра обработки данных, обеспечивающего эффективное планирования задач за счет их распределения по топологически близким вычислительным узлам и управления потоками данных.**

**Ключевые слова: распределенный центр обработки данных, облачные вычисления, программно-конфигурируемые сети, OpenFlow.**

Множество организаций используют облачные вычислительные центры обработки данных (ЦОД) с целью размещения своих научных и бизнес-приложений, что позволяет им избежать расходов на создание и поддержание функционирования собственных ЦОД. С другой стороны, владельцы облачных вычислительных ЦОД путем консолидации вычислительных ресурсов и систем хранения данных (СХД) способны снизить совокупную стоимость владения ИТ-инфраструктурой за счет обслуживания значительного количества клиентов, а также использования эффективных технических средств.

Существующие облачные решения недостаточно эффективно используют сетевые и вычислительные ресурсы. Это особенно отчетливо видно при исполнении коммуникационно-интенсивных задач, которые простаивают при выполнении синхронных сетевых операций, что в конечном счете ведет к снижению загрузки ресурсов ЦОД.

Данная проблема может быть решена с использованием программно-конфигурируемых сетей (ПКС) [1], [2], в основе которых лежит возможность динамического управления потоками данных в сети путем программной настройки таблицы потоков в коммутаторах OpenFlow. Каждый подобный коммутатор при получении пакета ищет подходящие для него правила в таблице потоков и применяет их к нему, в случае их отсутствия он посылает пакет контроллеру OpenFlow. Контроллер принимает решение о действиях над пакетом и устанавливает соответствующие правила в исходный и, возможно, в другие коммутаторы OpenFlow.

В рамках настоящего исследования программно-конфигурируемые сети позволят учитывать топологию и состояние сети при планировании групп экземпляров виртуальных машин, а также при диспетчеризации вычислительных задач, в случае реализации высокопроизводительных вычислений поверх вычислительного облака. Они также позволят эффективно распределять потоки данных в сети, чтобы минимизировать сетевую конкуренцию.

Для реализации предложенного подхода необходима разработка адекватной математической модели распределенного вычислительного ЦОД с ПКС его сегментов. Ранее предложенный в [3] авторский вариант модели не полностью учитывал виртуализацию вычислительных ресурсов, а также не принимал во внимание возможно организации высокопроизводительных вычислений поверх вычислительного облака. Предложенная в рамках данной статьи модель исправляет приведенные недостатки.

**Методы проведения исследования**

При разработке модели распределенного вычислительного ЦОД используется системный подход, позволяющий адекватно синтезировать структуру ЦОД, выделить отдельные его компоненты. Для представления всего ЦОД, отдельных его сегментов, облачной системы и грид-системы используются элементы теории графов.

**Результаты исследования**

В рамках модели облачной вычислительной системы выделены:

1. Структурная модель распределенного ЦОД с ПКС его сегментов.
2. Модель вычислительного облака.
3. Модель грид-системы, функционирующей поверх вычислительного облака.

Опишем более подробно каждую из данных моделей.

**Структурная модель распределенного ЦОД с ПКС его сегментов** может быть определена в виде ориентированного мультиграфа вида: $Datacenter = (Segments, Connections(t))$, где вершины $Segments = \{Segment_1, Segment_2, ..., Segment_s\}$ – множество отдельных территориально разделенных сегментов (автономных систем) ЦОД, связанных между собой через глобальные сети; дуги $Connections(t) = \{(Segment_i, Segment_j)\}$ – направленные связи между сегментами через глобальные сети. Для соединения сегментов используются граничные шлюзы и протокол BGP.

Сегмент $Segment_k \in Segments$ распределенного вычислительного ЦОД может быть описан в форме взвешенного неориентированного мультиграфа:

$$Segment_k = (Devices_k, Links_k, Flows_k(t)).$$

Его вершины представляют собой разбиение множества

$$Devices_k = Nodes_k \cup Switches_k \cup OFSwitches_k \cup SBalancers_k \cup Storages_k.$$

Здесь $Nodes_k = \{Node_{k1}, Node_{k2}, ..., Node_{kn_k}\}$ обозначает множество вычислительных узлов,
$Switches_k = \{Switch_{k1}, Switch_{k2}, ..., Switch_{km_k}\}$ – коммутаторов без поддержки OpenFlow,
$OFSwitches_k = \{OFSwitch_{k1}, OFSwitch_{k2}, ..., OFSwitch_{kf_k}\}$ – коммутаторов OpenFlow,
$Controllers_k = \{Controller_{k1}, Controller_{k2}, ..., Controller_{kz_k}\}$ – контроллеров OpenFlow,
$Gateways_k = \{Gateway_{k1}, Gateway_{k2}, ..., Gateway_{kw_k}\}$ – граничных шлюзов,
$SBalancers_k = \{SBalancer_{k1}, SBalancer_{k2}, ..., SBalancer_{kb_k}\}$ – балансировщиков коммутаторов OpenFlow,
$Storages_k = \{Storage_{k1}, Storage_{k2}, ..., Storage_{kh_k}\}$ – хранилищ данных.

Ребра мультиграфа $Links_k = \{(p_{ki}, p_{kj})\}$ представляют собой двусторонние сетевые связи между портами устройств сети, причем допускается наличие нескольких параллельных связей между двумя устройствами через разные пары портов.

$Flows_k(t) = \{Flow_{ki}(t)\}$ – набор потоков, появившихся в сегменте $Segment_k$ к моменту времени $t$.

Каждый вычислительный узел $Node_{ki} \in Nodes_k$ характеризуется следующими параметрами и динамическими характеристиками:

$$Node_{ki} = (M_{ki}, D_{ki}, C_{ki}, S_{ki}, P_{ki}^{node}, m_{ki}(t), d_{ki}(t), u_{ki}(t), s_{ki}^{node}(t)),$$

где $M_{ki} \in N$ и $D_{ki} \in N$ – соответственно размеры его оперативной и дисковой памяти в Мб; $C_{ki} \in N$ – количество его вычислительных ядер, $S_{ki} \in R^+$ – величина их производительности относительно самого медленного ядра во всем распределенном ЦОД; $P_{ki}^{node} = \{p_{kij}^{node}\}_j$ – множество сетевых портов; $m_{ki}(t) \in [0,1]$ и $d_{ki}(t) \in [0,1]$ – соответственно удельные доли загруженности оперативной и дисковой памяти вычислительного узла в момент времени $t$; $u_{ki}(t) = (u_{ki1}(t), u_{ki2}(t), ..., u_{kiC_{ki}}(t))$ – вектор загруженностей каждого из вычислительных ядер узла в момент времени $t$, причем $u_{kij}(t) \in [0,1]$; $s_{ki}^{node}(t) \in \{"online", "offline"\}$ – состояние узла в момент времени $t$.

Каждый сетевой порт узла $p_{kij}^{node} \in P_{ki}^{node}$ имеет следующие параметры и динамические характеристики:

$$p_{kij}^{node} = (IP_{kij}^{node.port}, MAC_{kij}^{node.port}, MaxBand_{kij}^{node.port}, Load_{kij}^{node.port}(t), Q_{kij}^{node.port}(t), s_{kij}^{node.port}(t)),$$

где $MaxBand_{kij}^{node.port} \in N$ – максимальная пропускная способность передачи порта в Кбит/с; $IP_{kij}^{node.port}$ и $MAC_{kij}^{node.port}$ – соответственно его IP-адрес и MAC-адрес; $Load_{kij}^{node.port}(t) \in [0,1]$ – удельная загрузка исходящей связи порта в момент времени $t$; $Q_{kij}^{node.port}(t)$ – набор очередей, ассоциированных с портом, и их состояния в момент времени $t$; $s_{kij}^{node.port}(t) \in \{"on", "off"\}$ – состояние порта в момент времени $t$.

Заметим, что все вычислительные узлы являются SMP-узлами, состоящими из однотипных многоядерных процессоров.

Динамические характеристики узлов могут собираться через регулярные интервалы времени с помощью протокола SNMP.

$Q_{kij}^{node.port}(t) = \{Q_{kije}^{node.port}(t)\}_e$ обозначает набор очередей пакетов, ассоциированных с конкретным портом $p_{kij}^{node.port}$ вычислительного узла $Node_{ki}$ и значением PCP равным $e$. Они используются, чтобы обеспечить согласно QoS мини-

мальную гарантированную пропускную способность и максимальную гарантированную задержку для заданных сетевых связей.

С каждой подобной очередью $Q_{kije}^{node.port}(t)$ связаны следующие параметры и динамические характеристики:

$$Q_{kije}^{node.port}(t) = (MinBand_{kije}^{node.port}(t), MaxDelay_{kije}^{node.port}(t)),$$

где $MinBand_{kije}^{node.port}(t) \in N \cup \{0\}$ и $MaxDelay_{kije}^{node.port}(t) \in N \cup \{0\}$ представляют собой соответственно минимальную пропускную способность (в Кб/с) и максимальную задержку для соответствующей очереди порта (в мкс), которые были установлены механизмом обеспечения QoS.

Каждый коммутатор без поддержки OpenFlow $Switch_{ki} \in Switches_k$ определяется следующими параметрами и динамическими характеристиками:

$$Switch_{ki} = (MAC_{ki}^{switch}, P_{ki}^{switch}, s_{ki}^{switch}(t)).$$

Здесь $MAC_{ki}^{switch}$ – его MAC-адрес,

$P_{ki}^{switch} = \{p_{kij}^{switch}\}_j$ – множество сетевых портов. Динамическая характеристика $s_{ki}^{switch}(t) \in \{"online","offline"\}$ определяет состояние коммутатора.

Сетевые порты коммутатора без поддержки OpenFlow имеют аналогичные параметры и характеристики, что и вычислительный узел, за исключением двух первых отсутствующих значений IP и MAC-адресов

Каждый OpenFlow-коммутатор $OFSwitch_{ki} \in OFSwitches_k$ имеет статические параметры и динамические характеристики, описываемые набором:

$$OFSwitch_{ki} = (OFVer_{ki}, IP_{ki}, MAC_{ki}^{ofswitch}, P_{ki}^{ofswitch}, Contr_{ki}(t), FlowEntries_{ki}(t), s_{ki}^{ofswitch}(t)),$$

где $OFVer_{ki} \in \{"1.0","1.1","1.2","1.3"\}$ – поддерживаемая коммутатором версия протокола OpenFlow;

$IP_{ki}$ – его IP для управления;

$MAC_{ki}^{ofswitch}$ – его MAC-адрес;

$P_{ki}^{ofswitch} = \{p_{kij}^{ofswitch}\}_j$ – множество его сетевых портов.

К числу динамических характеристик относятся: $Contr_{ki}(t) \in Controllers_k \cup \{\Theta\}$ – контроллер, к которому подключен коммутатор OpenFlow в момент времени $t$ (если коммутатор еще не подключен к контроллеру, то $Contr_{ki}(t) = \Theta$, $FlowEntries_{ki}(t)$ – состояние его таблицы записей о потоках в момент времени $t$, $s_{ki}^{ofswitch}(t) \in \{"online","ofline"\}$ – состояние в момент времени $t$.

Каждая запись о потоке $FlowEntry_{kij} \in FlowEntries_{ki}(t)$ имеет вид:

$$FlowEntry_{kij} = (Match_{kij}, Actions_{kij}, TimeOut_{kij}^{flow.entry}, Flow_{kii}, Counters_{kii}^{flow.entry}(t)),$$

где $Match_{kij}$ представляет собой набор полей для проверки на совпадение с заголовками пакета; $Actions_{kij}$ – набор действий, выполняемых над пакетом, при совпадении его заголовков с $Match_{kij}$; $TimeOut_{kij}^{flow.entry} \in N \cup \{0\}$ – время, на которое установлена запись о потоке; $Flow_{kij} \in Flows_k \cup \{\Theta\}$ – поток, к которому относится данное правило OpenFlow (если запись о потоке не относится ни к какому глобальному потоку, являясь потоком внутри коммутатора, то $Flow_{kij} = \Theta$); $Counters_{kij}^{flow.entry}(t)$ – статистические счетчики OpenFlow.

Все пакеты, поступающие в коммутатор, сопоставляются со всеми правилами из таблицы потоков. Если подходящее правило найдено (его $Match_{kij}$ соответствует заголовкам пакета), то выполняются все действия из набора $Actions_{kij}$ и обновляются значения счетчиков $Counters_{kij}^{flow.entry}(t)$, в противном случае пакет отправляется контроллеру, ассоциированному с коммутатором OpenFlow. Контроллер ответственен за определения способа обработки пакетов, для которых не нашлось подходящих правил в таблице коммутатора. После принятия решения контроллер добавляет или удаляет правила в таблицах потоков данного и других коммутаторов, подключенных к нему.

Сетевые порты коммутатора OpenFlow имеют аналогичные параметры и характеристики, что и коммутатор без поддержки OpenFlow. Отличие в том, что дополнительно имеется $Counters_{kije}^{ofswitch.port}(t)$ – набор счетчиков OpenFlow уровня очереди пакетов порта, также имеется набор счетчиков $Counters_{kij}^{ofswitch.port}(t)$ уровня порта коммутатора OpenFlow. Учитываются все основные счетчики стандарта OpenFlow [2].

Любой контроллер $Controller_{ki} \in Controllers_k$ сегмента $Segment_k$ обладает следующими параметрами и динамическими характеристиками:

$$Controller_{ki} = (IP_{ki}^{controller}, P_{ki}^{controller}, Switches_{ki}^{controller}(t), s_{ki}^{controller}(t)),$$

где $IP_{ki}^{controller}$ – IP-адрес контроллера;

$P_{ki}^{controller} = \{p_{kij}^{controller}\}_j$ – множество его сетевых портов;

$Switches_{ki}^{controller}(t) \subseteq OFSwitches_k$ – множество коммутаторов OpenFlow, подключенных к контроллеру в момент времени $t$;

$s_{ki}^{controller}(t) \in \{"online","offline"\}$ – состояние контроллера в момент времени $t$.

Множество $Switches_{ki}^{controller}(t)$ в каждом сегменте динамически формируется балансировщиком коммутаторов OpenFlow, находящимся в активном состоянии.

Каждый граничный шлюз $Gateway_{ki} \in Gateways_k$ сегмента $Segment_k$ имеет следующие параметры и динамические характеристики:

$$Gateway_{ki} = (P_{ki}^{gateway}, Preficies_{ki}(t), s_{ki}^{gateway}(t)).$$

Здесь $P_{ki}^{gateway} = \{p_{kij}^{gateway}\}_j$ – множество сетевых портов граничного шлюза,

$Preficies_{ki}(t) = \{(IP_{kij}^{prefix}, Mask_{kij})\}$ – набор префиксов (IP-адресов и масок) других сегментов ЦОД, известных граничному шлюзу на момент времени $t$,

$s_{ki}^{gateway}(t) \in \{"online","offline"\}$ – состояние граничного шлюза в момент времени $t$.

Каждому префиксу $(IP_{kij}^{prefix}, Mask_{kij})$ взаимно-однозначно соответствует дуга, принадлежащая множеству $Connections(t)$ мультиграфа сегментов ЦОД.

Балансировщик коммутаторов OpenFlow $SBalancer_{ki} \in SBalancers_k$ сегмента $Segment_k$ обладает следующими параметрами и динамическими характеристиками:

$$SBalancer_{ki} = (Policy_{ki}, P_{ki}^{sbalancer}, s_{ki}^{sbalancer}(t)),$$

где $Policy_{ki} \in \{"RoundRobin","MinLoad"\}$ – политика балансировки коммутаторов OpenFlow,

$P_{ki}^{sbalancer} = \{p_{kij}^{sbalancer}\}_j$ – множество сетевых портов балансировщика,

$s_{ki}^{sbalancer}(t) \in \{"active","reserved","failed"\}$ – его состояние в момент времени $t$.

Сетевые хранилища содержат образы экземпляров виртуальных машин, вычислительные задачи грид-системы, базы данных приложений, а также инфраструктурные компоненты вычислительного облака.

Каждое хранилище $Storage_{ki} \in Storages_k$ сегмента $Segment_k$ имеет следующие параметры и динамические характеристики:

$$Storage_{ki} = (MaxVolume_{ki}, P_{ki}^{storage}, Volume_{ki}(t),$$

$$\overline{R}_{ki}(t), \overline{W}_{ki}(t), s_{ki}^{storage}(t)),$$

где $MaxVolume_{ki} \in N$ – максимальный объем хранилища в килобайтах;

$P_{ki}^{storage} = \{p_{kij}^{storage}\}_j$ – множество его сетевых портов;

$Volume_{ki}(t) \in N \cup \{0\}$ – доступный объем хранилища в килобайтах в момент времени $t$;

$\overline{R}_{ki}(t)$ и $\overline{W}_{ki}(t)$ – соответственно средние установившиеся к моменту времени $t$ скорости чтения и записи данных;

$s_{ki}^{storage}(t) \in \{"online","offline"\}$ – состояние хранилища в момент времени $t$.

Все сетевые порты контроллера, граничного шлюза, балансировщика и хранилища имеют параметры и динамические характеристики, аналогичные параметрам и характеристикам порта вычислительного узла.

На основе структурной модели распределенного вычислительного ЦОД с ПКС его сегментов разработана **модель вычислительного облака**.

Основное ее отличие от структурной модели заключается в том, что вводятся виртуальные вычислительные узлы, представляющие собой экземпляры виртуальных машин, соединенных на каждом физическом вычислительном узле с помощью виртуального коммутатора. В соответствии с рисунком 1 в структуре физического вычислительного узла выделяются: виртуальные вычислительные узлы, ОС физического вычислительного узла и виртуальный коммутатор.

Изображенные линии обозначают сетевые соединения между одной или более парой портов соответствующих сетевых устройств. Все виртуальные вычислительные узлы подключены с помощью одного или нескольких виртуальных портов к виртуальному коммутатору. Сам физический вычислительный узел с помощью виртуального порта соединен с виртуальным коммутатором. Виртуальный коммутатор подключен к физическим портам физического вычислительного узла, через которые он связан с другими устройствами сети.

Виртуальный коммутатор в зависимости от используемого монитора виртуальных машин может представлять собой как коммутатор без поддержки OpenFlow, так и OpenFlow-коммутатор.

Далее по тексту переменные с символом штриха обозначают величины, введенные ранее и дополненные в настоящем пункте.

Модель облачной системы распределенного вычислительного ЦОД с ПКС его сегментов формализована в виде ориентированного мультиграфа вида:

$$Cloud = (Segments', Connections(t), CloudSoftware(t)),$$

где вершины

$Segments' = \{Segment'_1, Segment'_2, ..., Segment'_s\}$ представляют собой дополненные параметрами и динамическими характеристиками обозначения сегментов ЦОД,

$Connections(t) = \{(Segment_i, Segment_j)\}$ – направленные связи между сегментами,

$CloudSoftware(t)$ – состояния компонентов программного обеспечения облачной системы.

Сегмент $Segment'_k \in Segments'$ облачной системы может быть описан в форме взвешенного неориентированного мультиграфа:

$$Segment'_k = (Devices'_k, Links'_k, Flows_k(t)),$$

где вершины представляют собой разбиение множеств $Devices'_k = Devices_k \cup VirtualDevices_k$. Здесь $VirtualDevices_k$ – виртуальные сетевые устройства, они являются разбиением множеств

$$VirtualDevices_k = VirtualNodes_k \cup$$

$$\cup VirtualSwitches_k \cup VirtualOpenFlowSwitches_k,$$

где $VirtualNodes_k = \{VNode_{ki}\}_i$ – множество виртуальных вычислительных узлов,

$VirtualSwitches_k = \{VSwitch_{ki}\}$ – множество виртуальных коммутаторов без поддержки OpenFlow,

$VirtualOpenFlowSwitches_k = \{VOFSwitch_{ki}\}_i$ – множество виртуальных коммутаторов с поддержкой OpenFlow.

Будем условно считать, что каждый физический узел, включая случаи отсутствия назначенных на него экземпляров виртуальных машин или наличия ровного одного экземпляра, содержит ровно один виртуальный коммутатор. Тогда имеет место равенство:

$$|VirtualSwitches_k| + |VirtualOpenFlowSwitches_k| = |Nodes_k|$$

Множество сетевых связей сегмента $Links'_k$ представляет собой разбиение

$$Links'_k = Links_k \cup VirtualLinks_k,$$

где $VirtualLinks_k$ – виртуальные соединения.

Состояние компонентов программного обеспечения облачной системы включает следующие параметры и динамические характеристики:

$$CloudSoftware(t) = (Users, Flavors, Images, VMGroups(t)),$$

где $Users = \{User_1, User_2, ..., User_v\}$ – множество зарегистрированных пользователей системы;

$Flavors = \{Flavor_1, Flavor_2, ..., Flavor_e\}$ – типы виртуальных машин;

$Images = \{Image_1, Image_2, ..., Image_d\}$ – набор дисковых образов виртуальных машин;

$VMGroups(t)$ – набор параллельных вычислительных задач, представляющих собой группы экземпляров виртуальных машин, существующих в системе в момент времени $t$.

Каждый тип виртуальной машины включает в себя следующие параметры:

$$Flavor_l = (M_l^{flavor}, D_l^{flavor}, C_l^{flavor}).$$

Здесь $M_l^{flavor} \in N$ и $D_l^{flavor} \in N$ – объемы соответственно оперативной и дисковой памяти вир-

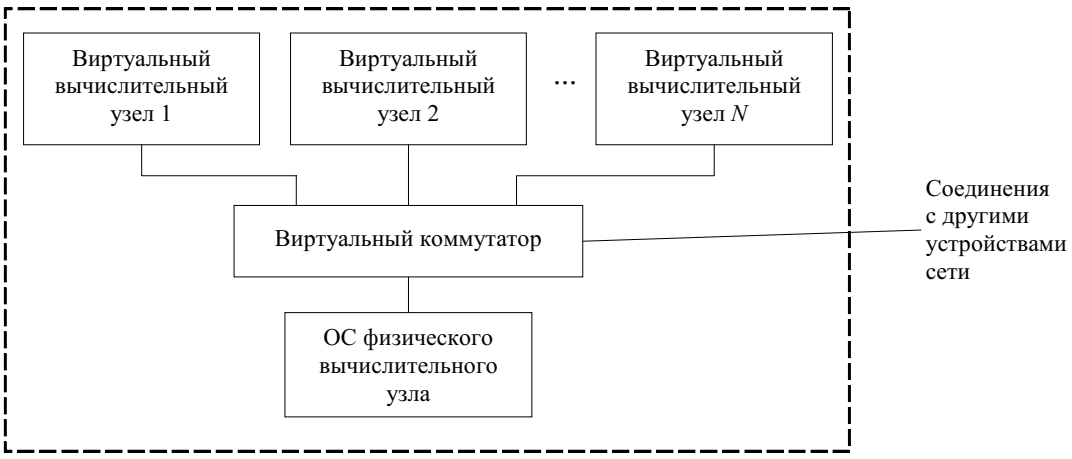


Рисунок 1. Структура физического вычислительного узла в модели облачной системы

туальной машины в мегабайтах, $C_l^{flavor} \in N$ – количество вычислительных ядер физического узла, отдаваемых виртуальной машине.

Данная модель позволяет формализовать современные облачные системы, включая систему на базе программного обеспечения OpenStack.

Вычислительная грид-система, функционирующая поверх облачной системы, может быть описана с помощью набора значений вида:

$$Grid = (Clusters, Dispatcher, dist_{VM}, Cloud),$$

где $Clusters = \{Cluster_1, Cluster_2, ..., Cluster_q\}$ – набор виртуальных вычислительных кластеров; $Dispatcher$ – диспетчер грид-системы; $dist_{VM}$ – функция вычисления топологического расстояния между двумя экземплярами виртуальных машин; $Cloud$ – вычислительное облако, поверх которого функционирует грид-система.

Каждый виртуальный вычислительный кластер $Cluster_i$ представляет собой отдельную группу экземпляров виртуальных машин $VMGroup_{j_i}$ вычислительного облака $Cloud$, локализованную в пределах одного его сегмента $\Phi_{seg}(VMGroup_{j_i}) = Segment_k$. В каждом сегменте вычислительного облака не более одной группы экземпляров виртуальных машин:

$$\forall VMGroup_{j_{i_1}}, VMGroup_{j_{i_2}} \quad VMGroup_{j_{i_1}} \neq VMGroup_{j_{i_2}} \Rightarrow \Phi_{seg}(VMGroup_{j_{i_1}}) \neq \Phi_{seg}(VMGroup_{j_{i_2}}).$$

Каждая группа экземпляров виртуальных машин $VMGroup_{j_i}$ включает множество виртуальных машин вида

$$VMs_{j_i} = \{VM_{j_i 1}, VM_{j_i 2}, ..., VM_{j_i g_{j_i}}\}.$$

Пусть экземпляр виртуальной машины $VM_{j_i r}$ имеет тип

$Flavor_r = (M_r^{flavor}, D_r^{flavor}, C_r^{flavor})$ в вычислительном облаке и входит вместе с группой в его сегмент $Segment_k$, тогда он является виртуальным вычислительным узлом облака и одновременно узлом грид-системы, поэтому он описывается набором значений:

$$VM_{j_i r} = (M_r^{flavor}, D_r^{flavor}, C_r^{flavor}, S_{kr}^{vm}, P_{kr}^{vm}, I_{kr}^{vm}, m_{kr}^{vm}(t), d_{kr}^{vm}(t), u_{kr}^{vm}(t), s_{kr}^{vm}(t)).$$

Диспетчер включает единую для всей грид-системы очередь задач $Q_{jobs}$. В нее помещаются вновь поступающие по сети задачи от пользователей, алгоритм планирования обслуживает данную очередь, выбирая подходящие для планирования и диспетчеризации вычислительные задачи в каждом цикле планирования. Сам диспетчер выполняется в отдельной виртуальной машине $VM_{disp}$ облачной системы.

Обозначим в качестве

$$VM^{pool} = \bigcup_{\substack{VMGroup_{j_i} \in Clusters \\ VM_{j_i r} \in VMGroup_{j_i}}} VM_{j_i r}$$

– объединение виртуальных машин всех виртуальных кластеров грид-системы.

Грид-система $Grid$ может быть представлена в виде неориентированного полносвязного взвешенного графа, в котором вершинами являются элементы множества $VM^{pool} \cup \{VM_{disp}\}$ – виртуальные машины пула и диспетчера. Вес каждой дуги $(VM', VM'')$, соединяющей две любые виртуальные машины $VM', VM'' \in VM^{pool}$ из пула, определяется с помощью функции $dist_{VM}(VM', VM'')$, которая на основе данных о состоянии сети всего вычислительного облака формирует оценку задержки передачи одного пакета между виртуальными машинами в виртуальной сети грид-системы. Для формирования оценки предполагается использовать метод двумерной диффузионной аппроксимации.

Также, для удобства реализации модели вычислительной грид-системы, функционирующей поверх облачной системы, узел диспетчера может быть включен в виртуальную сеть каждого виртуального кластера.

**Обсуждение полученных результатов**

Разработанная модель распределенного вычислительного ЦОД позволяет:

1) описывать современные облачные системы, а также грид-системы, функционирующие поверх вычислительных облаков;

2) учитывать наличие ПКС для сегментов распределенного ЦОД;

3) учитывать коммуникационную составляющую и сетевую конкуренцию;

4) формировать виртуальные вычислительные кластеры – однородные или гетерогенные по составу вычислительных узлов, в том числе многопроцессорные;

5) учитывать виртуальную и физическую топологию облачной вычислительной системы.

В дальнейшем данная модель послужит основой для разработки алгоритмов управления распределенным вычислительным ЦОД с ПКС его сегментов.

**Выводы**

Представлена математическая модель распределенного вычислительного центра обработки данных, включающая в себя структурную модель, модель вычислительного облака и модель грид-системы, функционирующей поверх вычислительного облака. Модель является теоретической основой для разрабатываемого распределенного центра обработки данных, обеспечивающего эффективное планирование задач, путем их распределения по топологически близким вычислительным узлам, а также за счет управления потоками данных между ними с целью их топологической локализации и снижения сетевой конкуренции.

28.02.2013

**Исследования выполнены при поддержке Министерства образования и науки Российской Федерации в рамках проекта № 07.514.11.4153 ФЦП «Исследования и разработки по приоритетным направлениям развития научно-технологического комплекса России на 2007–2013 годы» и РФФИ (проекты №12-07-31089 и №12-07-31022)**

**Список литературы:**

1. OpenFlow – Enabling Innovation in Your Network [Электронный ресурс]. – Режим доступа : http://www.openflow.org/
2. OpenFlow Switch Specification, Version 1.0.0. [Электронный ресурс]. – Режим доступа: http://www.openflow.org/documents/openflow-spec-v1.0.0.pdf
3. Математические модели облачного вычислительного центра обработки данных с использованием OpenFlow / В. Н. Тарасов [и др.] // Вестник Оренбургского государственного университета. – 2012. – № 9 (145). – С. 150–155.

Сведения об авторе:

**Полежаев Петр Николаевич,** преподаватель кафедры математического обеспечения информационных систем Оренбургского государственного университета

460018, г. Оренбург, пр-т Победы, 13, тел. (3532) 372534, e-mail: peter.polezhaev@mail.ru

**UDC 519.687**

**Polezhaev P.N.**

Orenburg state university, e-mail: peter.polezhaev@gmail.com

**MATHEMATICAL MODEL OF DISTRIBUTED COMPUTING DATACENTER BASED ON SOFTWARE DEFINED NETWORKS FOR ITS SEGMENTS**

This paper proposes mathematical model of distributed computing datacenter. This model is a theoretical basis for the developed distributed computing datacenter. It provides effective task scheduling by their assignment to topologically closest computing nodes and managing of data flows between them.

Key words: distributed datacenter, cloud computing, software defined network, OpenFlow.

Bibliography:

1. OpenFlow – Enabling Innovation in Your Network [Electronic resource]. – Access mode : http://www.openflow.org/
2. OpenFlow Switch Specification, Version 1.0.0. [Electronic resource]. – Access mode : http://www.openflow.org/documents/openflow-spec-v1.0.0.pdf
3. Mathematical models of cloud computing datacenter based on OpenFlow / V. N. Tarasov [et al.] // Vestnik OSU. – 2012. – № 9(45). – P. 150–155.