УДК 519.687

Тарасов В.Н., Полежаев П.Н., Шухман А.Е., Ушаков Ю.А., Коннов А.Л.
Оренбургский государственный университет
E-mail: peter.polezhaev@gmail.com

МАТЕМАТИЧЕСКИЕ МОДЕЛИ ОБЛАЧНОГО ВЫЧИСЛИТЕЛЬНОГО ЦЕНТРА ОБРАБОТКИ ДАННЫХ С ИСПОЛЬЗОВАНИЕМ OPENFLOW

В статье представлены модели облачного вычислительного центра обработки данных и вычислительной задачи. Они используются авторами в рамках предложенного варианта развития облачной управляющей системы OpenStack за счет эффективного планирования 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.

Ключевые слова: программно-конфигурируемые сети, распределенные системы, вычислительные центры обработки данных, облачные вычисления.

Введение

Облачные вычислительные системы стали стандартом де-факто для многих прогрессивных областей сферы информационных технологий. Множество организаций используют их с целью размещения своих научных и бизнес-приложений, что позволяет им избежать расходов на создание и поддержание функционирования собственных центров обработки данных (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.

К числу существующих сетевых протоколов облачных вычислительных ЦОД относятся: Fiber Channel, Infiniband и традиционный Ethernet. Они имеют ограниченные возможности по управлению трафиком и обеспечению QoS. Усовершенствованные варианты протокола Ethernet – Converged Enhanced Ethernet и Cisco Data Center Ethernet включают расширения по управлению потоками на основе приоритетов, разделению пропускной способности, управлению перегрузками и логическим состоянием полос передачи данных, по обеспечению передачи без потерь, а также по одновременному использованию нескольких параллельных путей передачи данных между узлами. Основные недостатки данных решений – сложная децентрализованная схема управления потоками данных, основанная на множестве закрытых протоколов, значительная стоимость сетевого оборудования, сложность его расширения.

Программно-конфигурируемые сети (ПКС) – перспективные технологии, которые могут быть использованы в облачных вычислительных ЦОД. Принципы ПКС впервые возникли в исследовательских лабораториях Стэнфорда и Беркли [1] и в настоящее время развиваются в рамках консорциума Open Network Foundation и европейского проекта OFELIA.

В основе подхода ПКС лежит возможность динамически управлять пересылкой данных в сети с помощью открытого протокола OpenFlow. Все активные сетевые устройства объединяются под управлением сетевой операционной системы, которая обеспечивает приложениям доступ к управлению сетью. Сетевые контроллеры могут быть централизованными, использовать общие абстракции для пересылки пакетов.

За счет управления пересылкой данных в сети использование ПКС в облачных ЦОД позволяет реализовать схемы одновременной многопутевой передачи данных, управление потоками на основе приоритетов, виртуализацию сети, обеспечить QoS, эффективно распределить нагрузку на сеть. Открытость стандарта OpenFlow и вынесение логики управления в отдельный контроллер упрощает программное и аппаратное обеспечение активного сетевого оборудования, что позволит в будущем производителям снизить его стоимость, а следовательно уменьшатся затраты на создание ЦОД.

В настоящее время отсутствуют решения на основе ПКС для облачных вычислительных ЦОД, направленных на управление потоками данных задач с целью их топологической локализации и снижения сетевой конкуренции между ними, что позволит повысить эффективность

работы облачного вычислительного ЦОД. Это определяет новизну настоящей работы. Для реализации данного подхода необходимы адекватные математические модели облачного вычислительного ЦОД и вычислительной задачи.

Математическая модель облачного вычислительного ЦОД

Облачный вычислительный ЦОД может быть описан в форме взвешенного неориентированного мультиграфа вида

$$Cloud = (Devices, Links, type, w_{n.}, w_{sw.}, w_{st.}, w_{l.}, Flavors, Users), \quad (1)$$

в котором вершины представляют собой разбиение множеств

$$Devices = Nodes \cup Switches \cup \cup Storages \cup Controllers,$$

здесь $Nodes = \{N_1, N_2, ..., N_n\}$ обозначает множество вычислительных узлов, $Switches = \{S_1, S_2, ..., S_m\}$ – коммутаторов, $Storages = \{F_1, F_2, ..., F_q\}$ – хранилищ данных, $Controllers = \{H_1, H_2, ..., H_z\}$ – контролеров OpenFlow. Ребра мультиграфа $Links = \{L_{ij}\}$ представляют собой двусторонние сетевые связи между устройствами сети, причем допускается наличие нескольких параллельных связей между двумя устройствами.

Тип сетевого устройства может быть определен с помощью классифицирующей функции

$$type : Devices -> \{"node", "switch", "storage", "controller"\}.$$

Отображение $w_{n.}$ для каждого вычислительного узла N_i задает вектор его характеристик:

$$w_{n.}(N_i) = (w_{n.\,stat.}(N_i), w_{n.\,dyn.}(N_i, t)),$$

где $w_{n.\,stat.}(N_i)$ и $w_{n.\,dyn.}(N_i, t)$ соответственно обозначают статические параметры и динамические характеристики N_i. Статические параметры узла представляются вектором

$$w_{n.\,stat.}(N_i) = (M_i, D_i, C_i, P_i),$$

содержащим: размер его оперативной памяти M_i, дисковой памяти D_i, число вычислительных ядер C_i и их характеристики производительности $P_i = (P_{i1}, P_{i2}, ..., P_{iC_i})$. Динамические характеристики могут быть заданы вектором:

$$w_{n.\,dyn.}(N_i, t) = (m_i(t), d_i(t), c_i(t), u_i(t), vm_i(t)).$$

Здесь $m_i(t)$ и $d_i(t)$ описывают соответственно объем доступной оперативной и дисковой памяти узла в момент времени $t \geq 0$, $c_i(t) = (c_{i1}(t), c_{i2}(t), ..., c_{iC_i}(t))$ – вектор признаков занятости вычислительных ядер ($c_{ik}(t) = 0$ – ядро свободно, $c_{ik}(t) = 1$ – занято) в момент времени t, $u_i(t) = (u_{i1}(t), u_{i2}(t), ..., u_{iC_i}(t))$ – вектор загруженности вычислительных ядер в момент времени t. $vm_i(t)$ описывает набор вычислительных задач, которые выполняются на узле N_i в момент времени t. Данная информация может собираться менеджером сети облачного вычислительного ЦОД через регулярные интервалы времени с помощью протокола SNMP.

Каждый OpenFlow-коммутатор, также как и узел, имеет статические параметры и динамические характеристики:

$$w_{sw.}(S_j) = (w_{sw.\,stat.}(S_j), w_{sw.\,dyn.}(S_j, t)).$$

Статические параметры S_j включают следующие значения:

$$w_{sw.\,stat.}(S_j) = (Et_j, Pc_j, OF_j, Ts_j),$$

где $Et_j \in \{"100Mbit\ Ethernet", "1\ Gbit\ Ethernet", "10\ Gbit\ Ethernet"\}$ определяет поддерживаемую версию протокола Ethernet, Pc_j – количество портов коммутатора, $OF_j \in \{"1.0", "1.1", "1.2", "1.3"\}$ – поддерживаемая версия протокола OpenFlow. В коммутаторе таблица OpenFlow имеет максимум Ts_j записей о потоках.

Динамические характеристики коммутатора могут быть представлены вектором:

$$w_{sw.\,dyn.}(S_j, t) = (Ft_j(t), Q_j(t), Pt_j(t)). \quad (2)$$

Здесь $Ft_j(t)$ – состояние таблицы потоков (flow table) OpenFlow в момент времени t. Каждая запись – правило OpenFlow (flow entry) следующего вида:

$$Rule_l = (Match_l, Counters_l, Actions_l).$$

$Match_l$ представляет собой набор полей для проверки на совпадение с заголовками пакета, $Counters_l$ – статистические счетчики, $Actions_l$ – действия, выполняемые над пакетом.

Все пакеты, поступающие в коммутатор, сопоставляются со всеми правилами из таблицы потоков. Если подходящее правило найдено (его $Match_l$ соответствует заголовкам пакета), то выполняются все действия $Actions_l$ этого правила и обновляются значения счетчиков $Counters_l$, в противном случае пакет отправляется контроллеру, ассоциированному с коммутатором. 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.

Протокол OpenFlow версии 1.0 поддерживает следующие поля для задания части $Match_l$ правил коммутации: номер входящего порта, Ethernet адреса источника и получателя, тип Ethernet пакета, идентификатор и приоритет VLAN, IP-адреса источника и получателя, тип протокола IP, Type of Service (ToS) биты протокола IP и TCP/UDP порты источника и получателя.

Каждому полю $Match_l$ может задаваться конкретное значение или «ANY» (любое значение). Наличие данных полей позволяет анализировать заголовки пакетов на уровнях L2–L4, что дает возможность на основе OpenFlow реализовать коммутацию, маршрутизацию, сетевой брандмауэр, систему обнаружения вторжений и т. п.

$Actions_l$ может содержать следующие действия [2]:

– «Forward». Отправить пакет на конкретный порт коммутатора;

– «Forward all». Отправить пакет на все порты, исключая входящий;

– «Forward controller». Выполнить инкапсуляцию пакета и отправить его OpenFlow-контроллеру;

– «Forward local». Отправить пакет в локальный сетевой стек коммутатора;

– «Forward table». Выполнить действия в таблице;

– «Forward in port». Отправить пакет назад во входящий порт;

– «Forward normal». Обработать пакет без OpenFlow стандартными средствами коммутатора;

– «Forward flood». Отправить пакет по минимальному покрывающему дереву, не включая входящий порт;

– «Enqueue». Поместить пакет в заданную очередь QoS;

– «Modify field». Изменить заданное поле заголовка пакета.

Пустой список действий $Actions_l$ обозначает удаление пакета. Наиболее часто используемое действие – коммутация пакета на указанный порт.

OpenFlow поддерживает счетчики для каждой записи о потоке в отдельности. $Counters_l$ содержит следующие значения [2]: общее число пакетов, обработанных данным правилом, общий размер всех пакетов в байтах, обработанных данным правилом, и продолжительность действия правила после добавления в коммутатор в секундах и миллисекундах.

Данные метрики дают представление об использовании записей о потоках.

На уровне таблицы $Ft_j(t)$ имеются следующие счетчики [2]: количество активных правил, количество поисков пакетов, количество сопоставленных пакетов.

Данные значения могут собираться через регулярные интервалы времени с помощью протокола OpenFlow.

В формуле (2) $Q_j(t) = \{Q_{jkr}(t)\}$ обозначает набор очередей пакетов, ассоциированных с конкретным портом коммутатора и значением ToS. Они используются, чтобы обеспечить согласно QoS минимальную гарантированную пропускную способность для заданных сетевых связей.

С каждой подобной очередью $Q_{jkr}(t)$ связаны следующие метрики [2]: количество переданных пакетов, число переданных байтов и количество ошибок, возникающих при переполнении.

$$Pt_j(t) = \{Pt_{j1}(t), Pt_{j2}(t), ..., Pt_{jPc_j}(t)\} \quad (\text{см. } (2))$$

представляет собой набор динамических характеристик портов коммутатора S_j. К их числу относятся: состояние порта (включен или выключен), общие количества полученных и переданных пакетов, общее число полученных байтов, общее число переданных байтов, общие количества полученных и переданных уведомлений об удалении пакетов, общие количества полученных и переданных ошибок, общее количество полученных ошибок выравнивания фрейма, общее количество полученных CRC-ошибок, общее число коллизий.

Данные метрики также собираются с помощью протокола OpenFlow через регулярные интервалы времени. Они помогают обнаруживать перегрузки сетевых связей и коммутаторов, их отказы.

Сетевые хранилища содержат вычислительные задачи (образы экземпляров виртуальных машин), базы данных приложений, а также инфраструктурные компоненты облачной вычислительной системы. Каждое хранилище

имеет следующий вектор значений, являющийся частью формулы (1):

$$w_{st.}(F_k) = (Vl_k, vl_k(t), r_k(t), w_k(t)),$$

где Vl_k – общий размер хранилища, $vl_k(t)$ – размер свободной его части в момент времени t, $r_k(t)$ и $w_k(t)$ – соответственно установившиеся средние скорости чтения и записи данных на момент времени t.

Для каждой сетевой связи $l_{ij} \in Links$ функция $w_{l.}(l_{ij})$ (см. формулу (1)) определяет вектор статических параметров и динамических характеристик:

$$w_{l.}(l_{ij}) = (B_{ij}, b_{ij}(t), lat_{ij}(t)),$$

где B_{ij} – максимальная пропускная способность сетевой связи, измеряемая при условии отсутствия сетевой конкуренции, $b_{ij}(t)$ – пропускная способность в момент времени t, $lat_{ij}(t)$ – значение латентности в момент времени t. Последние две характеристики учитывают сетевую конкуренцию.

Типы возможных экземпляров виртуальных машин задаются набором

$Flavors = \{Fl_1, Fl_2, ..., Fl_e\}$ из формулы (1). Каждый тип характеризуется следующими параметрами:

$$Fl_k = (Cr_k, Mr_k, Dr_k),$$

где Cr_k – количество виртуальных процессов, Mr_k и Dr_k – соответственно объемы оперативной и дисковой памяти.

Все пользователи облачного вычислительного ЦОД формируют множество

$Users = \{U_1, U_2, ..., U_v\}$ из формулы (1).

Данная модель адекватно описывает облачные вычислительные ЦОД с произвольными топологиями, в том числе с распределенными сегментами, управляемыми отдельными контроллерами OpenFlow. Модель ориентирована на поддержку версии 1.0 протокола OpenFlow, что позволяет за счет обратной совместимости моделировать произвольные контроллеры и OpenFlow-коммутаторы. Выбор данной версии также мотивирован тем, что в настоящее время более новые версии протокола не поддерживаются аппаратными коммутаторами.

Математическая модель вычислительной задачи

Каждая вычислительная задача J_k в облачном вычислительном ЦОД может быть представлена в виде набора экземпляров виртуальных машин. Они связаны между собой полносвязным образом, т. е. каждый экземпляр может по сети обмениваться пакетами с любым другим.

Вычислительная задача J_k описывается следующим вектором:

$$J_{k.} = (U_k, I_k, B_{\min.k}),$$

где U_k – пользователь-владелец задачи, $I_k = (I_{k1}, I_{k2}, ..., I_{kg_k})$ – набор экземпляров виртуальных машин, $B_{\min.k}$ – минимальная гарантированная пропускная способность связей между виртуальными машинами, задаваемая пользователями. Каждый экземпляр виртуальной машины I_{kj} определяется следующим набором значений:

$$I_{kj} = (Im_{kj}, Fl_{kj}, st_{kj}(t), h_{kj}(t), a_{kj}(t), cn_{kj}(t)).$$

Здесь Im_{kj} задает дисковый образ экземпляра виртуальной машины, Fl_{kj} определяет тип экземпляра, состояние задается значением

$$st_{kj}(t) \in \{"queued", "running", "migrating",$$

$$"stopped", "terminated"\}.$$

$h_{kj}(t)$ – номер узла, на котором выполняется данный экземпляр в момент времени t, если же $st_{kj}(t) \neq "running"$ и $st_{kj}(t) \neq "migrating"$, то $h_{kj}(t) = -1$ (несуществующему узлу). Номер узла может меняться во времени в случае миграции виртуальной машины. Отображение экземпляра на вычислительные ядра узла $N_{h_{kj}(t)}$ задается множеством

$$a_{kj}(t) = \{a_{kj1}(t), a_{kj2}(t), ..., a_{kj\,Cr_{Fkj}}(t)\},$$

где $a_{kji}(t)$ – номер ядра узла, занимаемого в момент времени t. $cn_{kj}(t)$ представляет собой набор метрик экземпляра виртуальной машины, включая время ее запуска, время остановки, суммарное процессорное время и др.

Предложенные модели ориентированы на поддержку облачной системы OpenStack [3] и полностью отражают особенности ее функционирования, включая поддержку IaaS и многоарендную архитектуру. Выбор данной системы мотивирован ее популярностью, открытостью, документированностью, наличием исходных кодов, а также возможностью ее адаптации под произвольные архитектуры вычислительных систем. Основные недостатки OpenStack – неэффективный алгоритм планирования экземпляров виртуальных машин, осуществляющий

выбор узлов для запуска на основе их ранжирования по линейным сверткам характеристик, отсутствие эффективных средств планирования и локализации сетевого трафика между экземплярами виртуальных машин, отсутствие гарантий по пропускной способности виртуальной сети.

Настоящая работа направлена на устранение данных недостатков за счет использования адаптированных вариантов ранее разработанных авторами алгоритмов планирования задач для кластерных вычислительных систем [4], ПКС на основе протокола OpenFlow, а также сетевых средств обеспечения QoS.

Предлагаемое решение на основе OpenStack и OpenFlow.

На рисунке 1 приведена схема планирования задач, предлагаемая в рамках настоящей работы. Серым цветом обозначены элементы, разрабатываемые авторами.

Опишем основные элементы схемы. Вычислительные узлы связаны с помощью коммутаторов с поддержкой OpenFlow. Использование данных коммутаторов позволяет управлять потоками данных групп виртуальных машин с целью их топологической локализации и снижения сетевой конкуренции между ними.

Диспетчер занимает центральное место облачной вычислительной системы, он решает проблемы управления и планирования вычислительных задач.

Контроллер OpenFlow представляет собой центральную компоненту ПКС. Он сосредотачивает в себе всю программную логику управления маршрутизацией пакетов. Используется для формирования таблиц коммутации в OpenFlow-коммутаторах на основе рассчитанных потоков данных между назначенными на физические узлы вычислительными задачами. Также используется для сбора сетевой статистики с коммутаторов OpenFlow и передачи ее службе управления сетью.

Служба управления сетью – служба, реализующая механизмы сбора статической и динамической информации об облачном вычислительном ЦОД. Включает обнаружение топологии сети на основе протокола LLDP, получение статистики от контроллера OpenFlow, сбор информации о загрузке вычислительных ядер и памяти узлов с помощью SNMP. Данная служба предоставляет динамические характеристики для модели облачного вычислительного ЦОД.

Планировщик задач на основе запроса пользователя на выделение ресурсов для вычислительной задачи, а также сведений о состоянии облачной вычислительной системы подбирает оптимальные вычислительные узлы для запуска виртуальных машин. Он опирается на описанные в рамках настоящей работы модели,

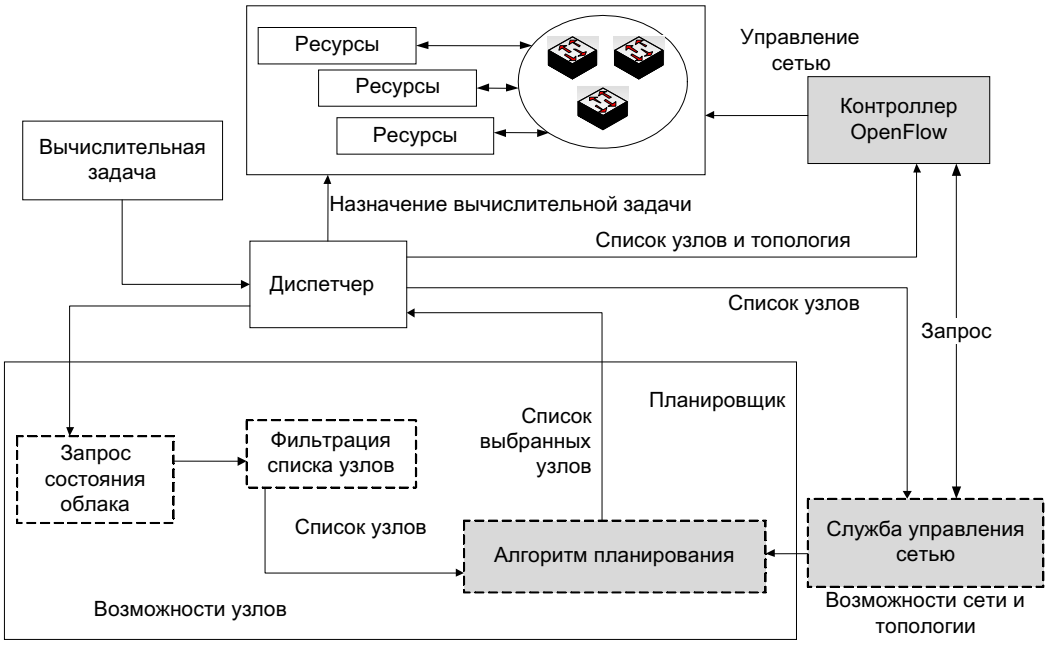

Рисунок 1. Схема планирования в улучшенном варианте системы OpenStack

а также на алгоритмы планирования, являющиеся модифицированными вариантами алгоритмов Backfill MDM и Backfill SDM для кластерных систем.

Заключение

В работе предложены математические модели облачного вычислительного ЦОД и вычислительной задачи, которые адекватно описывают все компоненты ЦОД. Разработанные модели могут быть использованы при создании системы управления вычислительным облаком на основе OpenStack и OpenFlow, реализующей повышение производительности за счет эффективного планирования задач и управления потоками данных между ними. Дальнейшие исследования предполагают реализацию системы управления сетевыми ресурсами для вычислительного ЦОД и экспериментальное исследование эффективности предлагаемых решений.

10.09.2012

Исследования выполнены при поддержке Минобрнауки РФ (ФЦП «Исследования и разработки по приоритетным направлениям развития научно-технологического комплекса России на 2007–2013 годы» №07.514.11.4153, ФЦП «Научные и научно-педагогические кадры инновационной России» на 2009–2013 годы, №14.B37.21.1881) и Российского фонда фундаментальных исследований (проекты №12-07-31089 и №12-07-31022)

Список литературы:

1. OpenFlow – Enabling Innovation in Your Network [Электронный ресурс]. – Режим доступа: URL: http://www.openflow.org/
2. OpenFlow Switch Specification, Version 1.0.0. [Электронный ресурс]. – Режим доступа: URL: http://www.openflow.org/documents/openflow-spec-v1.0.0.pdf
3. OpenStack Open Source Cloud Computing Software. [Электронный ресурс]. – Режим доступа: URL: http://www.openstack.org/
4. Полежаев, П.Н. Исследование алгоритмов планирования параллельных задач для кластерных вычислительных систем с помощью симулятора // Параллельные вычислительные технологии (ПАВТ'2010): Труды международной конференции. – Челябинск: ЮУрГУ, 2010. – С. 287–298.

Сведения об авторах:

Тарасов Вениамин Николаевич, профессор кафедры системного анализа и управления Оренбургского государственного университета, доктор технических наук, профессор, e-mail: vt@ist.psati.ru

Полежаев Петр Николаевич, ассистент кафедры математического обеспечения информационных систем Оренбургского государственного университета, e-mail: newblackpit@mail.ru

Шухман Александр Евгеньевич, заведующий кафедрой администрирования информационных систем Оренбургского государственного университета, кандидат педагогических наук, доцент, e-mail: shukhman@gmail.com

Ушаков Юрий Александрович, заведующий сектором информационных технологий Оренбургского государственного университета, кандидат технических наук, e-mail: ushakov@unpk.osu.ru,

Коннов Андрей Леонидович, системный администратор сектора информационных технологий Оренбургского государственного университета, кандидат технических наук, e-mail: andrey_konnov@mail.ru

460018, г. Оренбург, пр-т Победы, 13

UDC 519.687

Tarasov V.N., Polezhaev P.N., Shukhman A.E., Ushakov Yu.A., Konnov A.L.
Orenburg State University
E-mail: peter.polezhaev@gmail.com

MATHEMATICAL MODELS OF CLOUD COMPUTING DATACENTER BASED ON OPENFLOW

This paper presents a model of cloud computing data center and a model of computing tasks. They are used by authors in the proposed version of cloud management system based on OpenStack. It implements efficient jobs scheduling through their allocation to topologically closest computing nodes and efficient data flow control between nodes for their topological localization and reduction of network contention.

Key words: software defined networks, distributed systems, computing data centers, cloud computing.

Bibliography:

1. OpenFlow – Enabling Innovation in Your Network [Electronic resource]. – Access mode: URL: http://www.openflow.org/
2. OpenFlow Switch Specification, Version 1.0.0. [Electronic resource]. – Access mode: URL: http://www.openflow.org/documents/openflow-spec-v1.0.0.pdf
3. OpenStack Open Source Cloud Computing Software [Electronic resource]. – Access mode: URL: http://www.openstack.org/
4. Polezhaev, P.N. Research of algorithms parallel task scheduling for cluster computing systems using a simulation // Parallel Computing Technologies (PAVT 2010): Proceedings of the International Conference. – Chelyabinsk, 2010. – P. 287–298.