Интернет-журнал «Науковедение» ISSN 2223-5167 http://naukovedenie.ru/

Том 8, №6 (2016) http://naukovedenie.ru/vol8-6.php

URL статьи: http://naukovedenie.ru/PDF/61TVN616.pdf

DOI: 10.15862/61TVN616 (http://dx.doi.org/10.15862/61TVN616)

Статья опубликована 13.12.2016

Ссылка для цитирования этой статьи:
Белоножко П.П., Белоус В.В., Куцевич Н.А., Храмов Д.А. Свободные облачные аппаратно-программные платформы. Аналитический обзор // Интернет-журнал «НАУКОВЕДЕНИЕ» Том 8, №6 (2016) http://naukovedenie.ru/PDF/61TVN616.pdf (доступ свободный). Загл. с экрана. Яз. рус., англ.

УДК 004

Белоножко Павел Петрович
ФГБОУ ВО «Московский государственный технический университет им. Н.Э. Баумана (национальный исследовательский университет)», Россия, Москва
Доцент кафедры «САПР»
Кандидат технических наук
Старший научный сотрудник
E-mail: byelonozhko@mail.ru

Белоус Валентина Владимировна
ФГБОУ ВО «Московский государственный технический университет им. Н.Э. Баумана (национальный исследовательский университет)», Россия, Москва
Старший преподаватель кафедры САПР
Кандидат технических наук
E-mail: Walentina.belous@gmail.com

Куцевич Надежда Александровна
АО «РТСофт», Россия, Москва
Технический директор
Доктор технических наук
E-mail: rtsoft@rtsoft.ru

Храмов Дмитрий Александрович
Институт технической механики Национальной академии наук Украины и Государственного космического агентства Украины, Украина, Днепропетровск
Старший научный сотрудник
Кандидат технических наук
E-mail: dkhramov@mail.ru

Свободные облачные аппаратно-программные платформы. Аналитический обзор

Аннотация. За последние десять лет в сфере информационных технологий наблюдается стремительный рост популярности облачных вычислений — комплекса технологий, направленных на то, чтобы дать пользователю простой и удобный доступ к вычислительным ресурсам. Всё чаще у разработчиков программного обеспечения (ПО) возникает необходимость создавать программы, способные работать в облачной среде. Однако, несмотря на многочисленные преимущества, которые дают облачные вычисления, с ними связан и ряд проблемных вопросов, касающихся зависимости от поставщика услуг, безопасности данных, хранения персональных данных и т.п. Смягчить остроту этих проблем может использование при создании «облаков» свободного программного обеспечения, распространяемого с открытым исходным кодом.

Проведен анализ основных видов облачных сервисов и направлений их использования. Указаны возникающие при этом проблемы и способы их решения. Проанализированы особенности ряда свободных облачных программных платформ.

Рассмотрены существующие виды облачных сервисов, аппаратные, программные и организационные особенности их реализации. Проведен сравнительный анализ достоинств и недостатков существующих технологий создания облачных сервисов. Выделены проблемы и риски, актуальные с точки зрения пользователей облачного сервиса. Выполнен аналитический обзор свободных облачных аппаратно-программных платформ.

Результаты обзора могут быть полезны при необходимости решения задачи выбора свободной облачной аппаратно-программной платформы в зависимости от требований к разрабатываемым программным комплексам, предполагающим реализацию облачного сервиса.

Ключевые слова: облачные вычисления; платформа как сервис; оборудование как сервис; гипервизор; контейнер

Введение

За последние десять лет в сфере информационных технологий наблюдается стремительный рост популярности облачных вычислений - комплекса технологий, направленных на то, чтобы дать пользователю простой и удобный доступ к вычислительным ресурсам. Всё чаще у разработчиков программного обеспечения (ПО) возникает необходимость создавать программы, способные работать в облачной среде. Однако, несмотря на многочисленные преимущества, которые дают облачные вычисления, с ними связан и ряд проблемных вопросов, касающихся зависимости от поставщика услуг, безопасности данных, хранения персональных данных и т.п. Смягчить остроту этих проблем может использование при создании «облаков» свободного программного обеспечения, распространяемого с открытым исходным кодом.

Вначале мы вкратце рассмотрим, какие виды облачных сервисов существуют, как они могут быть использованы, какие проблемы при этом возникают и как эти проблемы можно решить. Более подробную информацию об облачных технологиях можно найти в книгах [1, 2]. Затем мы разберём особенности конкретных облачных программных платформ.

1. Облачные технологии

Облачные вычисления (cloud computing) - это комплекс технологий, обеспечивающих повсеместный и удобный сетевой доступ к динамически масштабируемым вычислительным ресурсам (процессорному времени, оперативной памяти, устройствам хранения данных, сетям передачи данных, приложениям и т.п.). В идеале, пользователь в любой момент времени и в любом месте, где бы он ни находился, должен получить в своё распоряжение аппаратно-программную платформу, характеристики которой можно изменять «на лету». Под аппаратно-программной платформой понимается единый комплекс средств вычислительной техники и системных программ [3].

С точки зрения пользователя, облачные вычисления дают возможность получать вычислительные ресурсы по сети у внешнего поставщика в виде услуги, оплата за которую производится в зависимости от объема потреблённых ресурсов, то есть примерно так же, как это происходит с услугами водо- или электроснабжения. При этом объём вычислительных ресурсов - виртуальный компьютер, который пользователь получает в своё распоряжение -

оперативно подстраивается под текущие запросы пользователя. Удобство доступа к услуге обеспечивается поддержкой широкого спектра терминальных устройств: персональных компьютеров, мобильных телефонов, интернет-планшетов.

Национальный институт стандартов и технологий США (NIST) выделяет следующие обязательные характеристики облачных вычислений [4]:

- самообслуживание по требованию (self service on demand) - потребитель самостоятельно выбирает, каким набором вычислительных ресурсов он будет пользоваться, и может при необходимости оперативно изменять этот набор без согласования с поставщиком услуг;
- универсальный доступ по сети - услуги доступны по сети передачи данных независимо от того, какое терминальное устройство использует потребитель;
- объединение ресурсов (resource pooling) - поставщик услуг объединяет имеющиеся в его распоряжении вычислительные ресурсы в единый пул (pool - общий котел) для динамического перераспределения этих ресурсов между потребителями; при этом потребители контролируют только основные параметры услуги (например, объём данных, скорость доступа), а фактическое распределение предоставляемых ресурсов осуществляет поставщик;
- мгновенная эластичность (гибкость) - предоставляемые пользователю вычислительные мощности могут оперативно увеличиваться или уменьшаться в автоматическом режиме, исходя из потребностей пользователя;
- учёт потребления - поставщик услуг автоматически исчисляет потреблённые ресурсы (объём хранимых данных, объём переданных данных, количество пользователей, количество транзакций и т.п.), и на этой основе оценивает объём предоставленных потребителям услуг.

С точки зрения поставщика услуг, важнейшей составляющей облачных вычислений является центр обработки данных или дата-центр (data center), который содержит:

- информационную инфраструктуру, служащую для обработки и хранения информации - основных функций дата-центра;
- телекоммуникационную инфраструктуру для передачи данных внутри дата-центра, а также между дата-центром и пользователями;
- инженерную инфраструктуру, обеспечивающую нормальное функционирование основных систем центра (кондиционирование, бесперебойное электроснабжение, пожарно-охранная сигнализация и т.п.).

Ключевыми понятиями, связанными с реализацией «облаков» являются масштабируемость и виртуализация [5]. Масштабируемость представляет собой способность вычислительной системы увеличивать свою производительность так, чтобы справляться с увеличением рабочей нагрузки. Благодаря виртуализации облачные вычисления абстрагируются от базовой аппаратной и программной инфраструктуры. Виртуализованные ресурсы предоставляются пользователю через определённые интерфейсы (программные интерфейсы API или сервисы). Такая архитектура обеспечивает масштабируемость и гибкость физического уровня «облака», изолируя изменения в центре обработки данных от влияния на конечного пользователя.

Потребитель услуг облачных вычислений может сэкономить на создании собственных центров обработки данных, закупке соответствующего оборудования и найме

обслуживающего персонала. Для владельца центра обработки данных выгода от применения облачных технологий заключается в более эффективном использовании ресурсов центра.

Облачные вычисления являются результатом длительной эволюции целого ряда информационных технологий. Так, виртуализация была впервые использована в мэйнфреймах IBM ещё в середине 1960-х годов. Вычислительная инфраструктура предоставлялась в качестве сервиса задолго до появления облачных вычислений. Такой подход назывался «коммунальные вычисления» - термин, который сейчас широко применяется при описании инфраструктурного уровня облачных систем [5]. Grid-технологии, реализующие распределённые вычисления на удалённых компьютерах, приобрели известность в конце 1990-х годов благодаря проекту поиска внеземного разума SETI@home [6].

Начало широкомасштабного предоставления услуг по доступу к вычислительным ресурсам через Интернет относится к августу 2006 года, когда компанией Amazon.com был запущен сервис Elastic Compute Cloud. Термин «облако» своим появлением обязан диаграммам, изображающим взаимодействие пользователей с Интернет, на которых последний представлялся в виде облака, как некая сложная инфраструктура за которой скрываются все технические детали.

1.1. Виды сервисов

Архитектура облачных систем состоит из трёх основных уровней (рис. 1), которым соответствуют три вида облачных сервисов:

- инфраструктура как сервис (Infrastructure-as-a-Service, IaaS);
- платформа как сервис (Platform-as-a-Service, PaaS);
- программное обеспечение как сервис (Software-as-a-Service, SaaS).

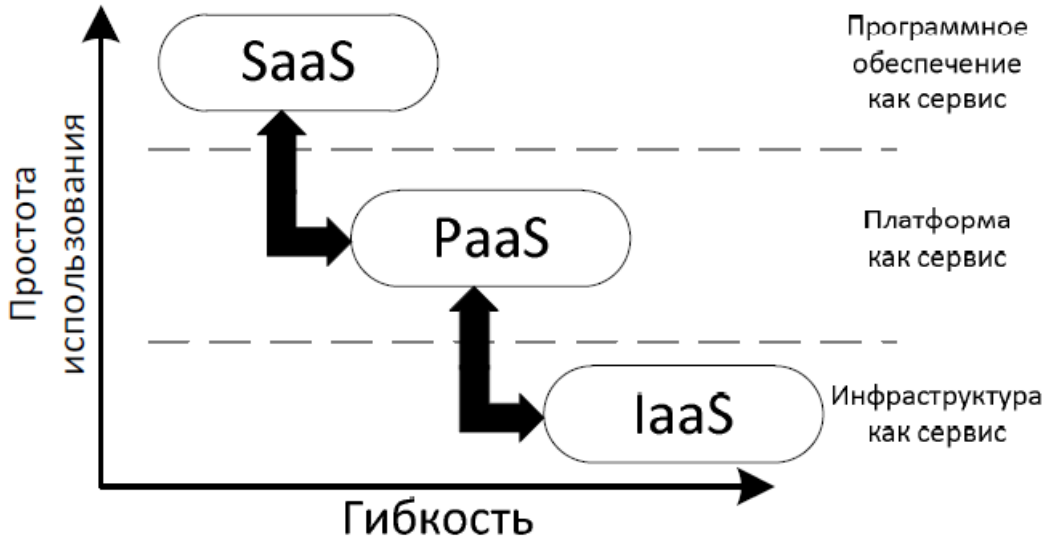

Рисунок 1. *Три уровня облачных вычислений [5]*

Инфраструктура как сервис - это форма предоставления услуг облачных вычислений, при которой пользователь получает возможность управлять средствами обработки и хранения данных, а также другими фундаментальными вычислительными ресурсами (виртуальными серверами и сетевой инфраструктурой) на которых он может самостоятельно устанавливать операционные системы (ОС) и прикладные программы для решения своих задач [8]. При этом контроль и управление основной физической и виртуальной инфраструктурой облака, в том

числе сетью, серверами, типами используемых ОС и систем хранения осуществляется поставщиком услуг. Такая форма предоставления услуг обладает максимальной гибкостью с точки зрения настройки на решение задач пользователя, однако и требует от последнего наибольшей, в сравнении с другими формами, квалификации.

Платформа как сервис позволяет пользователю получить доступ к использованию программной платформы: ОС, системам управления базами данных (СУБД), прикладному ПО, средствам разработки и тестирования ПО. Фактически, пользователь получает в аренду компьютерную платформу с установленной ОС и специализированными средствами для разработки, размещения и управления приложениями. Базовую комплектацию платформы определяет поставщик услуг. Потребитель может управлять развёрнутыми приложениями и, возможно, параметрами настройки конфигурации среды окружения.

Программное обеспечение как сервис даёт потребителю возможность использовать отдельные приложения поставщика услуг, запущенные в облачной инфраструктуре. В этом случае потребитель может управлять лишь ограниченным набором параметров настройки конкретного приложения. С другой стороны, таким видом облачных услуг проще всего пользоваться, и в настоящее время он наиболее распространён. Например, популярные сервисы Google - почта Gmail, программа для мгновенного обмена сообщениями Talk и офисный пакет Docs - являются SaaS.

В литературе упоминаются и другие уровни организации облачных сервисов. Так, естественным развитием облачных вычислений в направлении предоставления потребителю максимально готового к использованию вычислительного сервиса является компьютер как сервис (Desktop-as-a-Service, DaaS) - полностью готовое к использованию рабочее место. Если же двигаться в направлении максимальной приближенности к использованию реального оборудования, то предельным случаем будет оборудование как сервис (Metal-as-a-Service, MaaS). Фактически MaaS представляет собой аренду вычислительных мощностей, доступ к которым соответствует требованиям, предложенным NIST для облачных вычислений.

Для разработчиков ПО, предполагающих создавать и использовать свои приложения в облачной среде, наибольший интерес представляют уровни MaaS, IaaS и PaaS. Их мы и будем далее рассматривать.

1.2. Модели использования

В зависимости от аудитории, имеющей доступ к сервису, можно выделить «облака» следующих видов:

- частное облако (private cloud) - инфраструктура, предназначенная для использования облачных вычислений в масштабе одной организации;
- облако сообщества (community cloud) - вид облачной инфраструктуры, который предназначен для исключительного использования облачных вычислительных ресурсов конкретным сообществом пользователей, решающих общие задачи;
- общедоступное или публичное облако (public cloud) - инфраструктура, предназначенная для свободного использования облачных вычислений широкой публикой. Такое облако может находиться в собственности, управлении и эксплуатации коммерческих, научных и правительственных организаций и физически существует в юрисдикции владельца - поставщика услуг;
- гибридное облако (hybrid cloud) - представляет собой комбинацию различных видов облачных инфраструктур. Например, гибридное облако может объединять

частное и публичное облака, позволяя организациям обрабатывать конфиденциальные данные внутри своего частного облака, а другие данные передавать на обработку в публичное облако. Облака, составляющие гибридное облако, остаются уникальными объектами, связанными между собой стандартизованными или частными технологиями передачи данных и приложений.

Наиболее простым с точки зрения технической реализации является частное облако. Оно может не иметь API, поскольку все его ресурсы используются в рамках одной организации. Сложнее всего реализовать гибридные облака, так как в этом случае появляется необходимость в мониторинге внутренней и внешней вычислительной инфраструктуры, усложнении политик безопасности, стандартизации взаимодействия между разнородными облачными инфраструктурами и т.п.

1.3. Основные компоненты облачной платформы

В составе облачной платформы можно выделить несколько основных компонентов [5, 9].

Ядро платформы: среда и набор утилит, обеспечивающих предоставление, разработку и интеграцию облачных сервисов. Выбор того или иного ядра накладывает определенные ограничения на методы разработки и предоставления приложения. Например, может понадобиться использовать только поддерживаемые данной платформой языки программирования и средства разработки.

Интерфейс, через который пользователь взаимодействует с облаком. Чаще всего используется веб-интерфейс и различные API (например, EC2 или OCCI).

Хранилище данных. Современные облачные системы могут хранить огромные объёмы пользовательских данных (речь идёт о десятках и сотнях петабайт) и эти объёмы постоянно растут. В такой ситуации классические реляционные базы данных уже не дают удовлетворительных результатов по скорости обработки. Более того, облачным платформам часто приходится иметь дело со связанными структурами данных (графами, деревьями), что при использовании реляционного подхода вызывает дополнительные затруднения. В связи с этим, в последнее десятилетие активно развиваются NoSQL-СУБД (колоночные СУБД, такие как Hadoop; документ-ориентированные СУБД, вроде CouchDB и MongoDB), а также альтернативные подходы к обработке сверхбольших объемов информации.

Управление пользователями. Информация об основных потребителях облачных ресурсов используется для оптимизации и подстройки облака под их задачи. В первую очередь, необходимо обеспечить прозрачную авторизацию пользователей во всех сервисах облачной платформы. Кроме того, большинству приложений требуется умение отличать пользователей друг от друга для предоставления им релевантной информации. При этом, в силу распределённой природы облачных сервисов, необходимо обеспечить высочайший уровень безопасности при работе с пользовательской информацией.

Мониторинг и поддержка функционирующих приложений. Администрирование исполняющегося приложения может оказаться весьма непростой задачей, учитывая большое количество отдельных сервисов, из которых облачное приложение состоит. В связи с этим необходимо обеспечить интеграцию процессов администрирования и управления сервисами, а также пользовательскими задачами в виде единого «центра управления сервисами».

1.4. Взаимодействие с оборудованием

В настоящее время можно говорить о существовании двух типов облачных технологий, отличающихся характером взаимодействия с оборудованием: технологии, опирающейся на виртуализацию, и технологии, предполагающей непосредственную установку и функционирование «облака» на компьютерном «железе».

1.4.1. Виртуализация

Технологии виртуализации развиваются, начиная с середины 1960-х годов. В то время ОС называли супервизором (supervisor). Впоследствии, когда стало возможным запускать одну ОС поверх другой, появился термин гипервизор (hypervisor), обозначающий программу (или, реже, устройство), позволяющую выполнять одновременный запуск нескольких ОС на одном компьютере. Гипервизор осуществляет управление ресурсами и их разделение между различными ОС, выполняет изоляцию запущенных ОС друг от друга, а также обеспечивает их взаимодействие (обмен файлами, сетевое взаимодействие и т.п.).

Гипервизор сам по себе в некотором роде является минимальной операционной системой. Он предоставляет запущенным под его управлением ОС сервис виртуальной машины. При этом он эмулирует реальное (физическое) аппаратное обеспечение конкретной машины и управляет этими машинами, выделяя и освобождая ресурсы для них. Гипервизор позволяет независимое «включение», перезагрузку, «выключение» любой из виртуальных машин с той или иной ОС. С точки зрения гостевой (работающей под управлением гипервизора) ОС, виртуальная система ничем не отличается от физической.

Существуют три основных типа программных гипервизоров: автономный, гостевой и гибридный. Автономный гипервизор (VMware ESXi, Xen) способен работать на «голом» железе, то есть не требует ОС и напрямую предоставляет доступ операционным системам к оборудованию. Такой тип гипервизоров обеспечивает наилучшую производительность и поэтому используется для работы с серверными ОС. Гостевой гипервизор работает на основе какой-нибудь базовой ОС и все операции ввода-вывода осуществляет через процесс пользовательского уровня, запущенный под этой системой. Гибридный гипервизор работает автономно и содержит в себе специальную сервисную ОС, через которую гостевые системы получают доступ к оборудованию.

Кроме виртуализации при помощи гипервизоров существует так называемая контейнерная виртуализация. Отличие между этими подходами заключается в следующем [10].

Гипервизор эмулирует аппаратное обеспечение, поверх которого запускаются гостевые ОС. При этом всё, что «умеет» делать оборудование, должно быть доступно гостевой ОС со стороны базовой ОС (т.е. машины-«хозяина»). Напротив, контейнеры - это виртуализация на уровне операционной системы, а не на уровне оборудования: каждая гостевая ОС использует то же самое ядро (а в ряде случаев - и другие части ОС), что и базовая. В результате контейнеры меньше и компактнее гипервизорных гостевых сред, поскольку у них с «базой» гораздо больше общего.

Контейнерная виртуализация имеет и свои ограничения. Так, из-за совместного использования ядра, на одном сервере нельзя запускать гостевые ОС разных типов. Например, на системе с Linux-контейнерами невозможно запустить FreeBSD или Windows, хотя при этом можно запускать различные дистрибутивы Linux. Для гипервизоров такой проблемы не существует.

Наиболее известными открытыми программными средствами для контейнерной виртуализации являются пакеты OpenVZ, LXC и Docker, представляющий собой надстройку над LXC.

Заметим, что хотя технологии виртуализации и играют важную роль в организации облачных вычислений, но в предложенном NIST списке неотъемлемых свойств этих вычислений понятия виртуализации не содержится.

1.4.2. Непосредственное использование

Виртуализация позволяет при создании «облака» абстрагироваться от реального оборудования, имеющегося в дата-центре. За это удобство приходится расплачиваться накладными расходами, снижающими производительность системы. Если же требования к производительности высоки, а оборудование достаточно однородно, удобнее обойтись без виртуализации.

Словосочетание «bare metal provisioning» переводится на русский язык примерно, как «подготовка пустого компьютерного оборудования к запуску в эксплуатацию», и в настоящее время указывает на технологию создания облачного сервиса на «голом» компьютерном оборудовании.

Эти технологии развиваются с середины 1990-х годов, когда в связи с созданием высокопроизводительных вычислительных кластеров понадобилось ПО, обеспечивающее развёртывание, управление функционированием и обслуживанием подобных систем [11].

Несмотря на тенденцию к интеграции обеих технологий создания «облаков» (программы для управления высокопроизводительными кластерами всё чаще используют виртуальные машины, а ряд облачных платформ поддерживает установку на «голое железо»), существует ряд специализированных облачных сервисов, предоставляющих услуги высокопроизводительных вычислений, в частности, IBM SoftLayer.

Обе технологии имеют свои достоинства и недостатки, и выбор между ними диктуется потребностями проекта. Так, существенный перепад потребляемых вычислительных мощностей с резкими пиками потребления, делает предпочтительным использование виртуализации. Напротив, стабильное потребление вычислительных ресурсов с высокими требованиями к производительности даёт преимущество технологии непосредственной работы с оборудованием.

1.5. Проблемы и риски

Применение облачных вычислений, наряду с многочисленными преимуществами, влечёт за собой некоторые риски, связанные, в первую очередь, с зависимостью пользователя от поставщика «облачных» услуг. Эта зависимость гораздо сильнее и имеет больше аспектов, чем зависимость от поставщиков традиционного ПО.

Одним из аспектов проблемы является привязка пользователя к определённому поставщику (так называемый vendor-lock). Если поставщик начнёт диктовать пользователям неприемлемые условия, то последние будут вынуждены либо принять эти условия, либо перестать пользоваться сервисом. Если же поставщик по каким-то причинам уйдёт с рынка, то вместе с ним может исчезнуть и предоставляемый им сервис. При использовании традиционного ПО эти проблемы не возникали: законно приобретенный экземпляр программы можно использовать и после того, как её производитель изменит свои условия или прекратит существование.

Одним из способов борьбы с этой проблемой является стандартизация облачных сервисов. Для разработчиков ПО, работающего в «облаках», наиболее существенно наличие стандартного интерфейса (API) для работы с облачными сервисами. Здесь за последние несколько лет достигнут ряд успехов. С одной стороны, существует стандарт де-факто - EC2, с другой - разработан и приобретает всё большую популярность открытый стандарт OCCI (Open Cloud Computing Interface) [12]. При этом всё чаще появляются облачные платформы, поддерживающие оба стандарта.

Другим способом является изоляция работы пользовательского ПО от остальной системы, в частности, при помощи контейнерной виртуализации. Так, с помощью пакета Docker [13] можно запускать процессы в изолированном окружении - своеобразной «песочнице», где помимо самого процесса существуют только его процессы-потомки. Хотя при этом процесс работает в той же ОС, что и другие, обычные, процессы, он просто их не видит. Таким образом, с помощью Docker разработчик может отделить своё приложение от системы, поместить его в Docker-контейнер и в случае необходимости перенести на другую однотипную систему.

Следующая группа вопросов касается безопасности данных. В чьей собственности находится дата-центр? Под чьим управлением? Находится ли дата-центр внутри или вне юрисдикции владельца облачного сервиса? Широко распространена ситуация, когда компания-поставщик использует дата-центры, находящиеся в разных странах. При этом государство, на территории которого расположен дата-центр, может получить доступ к любой информации, которая в нём хранится. Например, по законам США, где находится самое большое число дата-центров, в этом случае компания-поставщик даже не имеет права разглашать факт передачи конфиденциальной информации кому-либо, кроме своих адвокатов.

В случае облачных сервисов персональные данные пользователей хранятся на удалённых серверах, что требует более высокого уровня доверия к поставщику. К тому же, в соответствии с российским законодательством, компании обязаны хранить имеющиеся в их распоряжении персональные данные россиян на серверах, расположенных на территории России (Федеральный закон от 21 июля 2014 г. № 242-ФЗ «О внесении изменений в отдельные законодательные акты Российской Федерации в части уточнения порядка обработки персональных данных в информационно-телекоммуникационных сетях»).

Кроме того, облачные сервисы работают на неконтролируемых со стороны пользователя компьютерах. Это ограничивает возможности изучения программы в работе и обратную разработку с целью обеспечения совместимости, что закреплено в российском законодательстве [14].

В этой связи логичными решениями представляются либо самостоятельное развёртывание облачной инфраструктуры (IaaS), либо использование уже имеющейся облачной платформы (PaaS), удовлетворяющей требованиям законодательства Российской Федерации.

2. Свободные облачные платформы

Современный этап развития облачных технологий начался с запуска в 2006 году компанией Amazon.com сервиса облачных вычислений Elastic Compute Cloud (EC2) и онлайнового хранилища файлов Simple Storage Service (S3). В настоящее время Amazon Web Services (AWS) объединяет более семидесяти облачных сервисов, среди которых: хранение данных, аренда виртуальных серверов, предоставление вычислительных мощностей и др. [15], а EC2 стал стандартом де-факто облачных вычислений.

Вслед за Amazon.com свои облачные платформы предложили Google и Microsoft. Однако, начавшись с проприетарных решений, по мере роста конкуренции на рынке стали появляться свободно распространяемые облачные платформы с открытым исходным кодом.

Первой из таких платформ, добившейся коммерческого успеха, стала IaaS-система Eucalyptus. Она и сейчас остаётся одним их наиболее популярных инструментов для создания «облаков». Однако по мере развития технологий и роста требований пользователей Eucalyptus стали критиковать за недостаточную масштабируемость. На рынке появились другие проекты с открытым кодом, призванные решить эту проблему: OpenStack, CloudStack, OpenNebula и др. Все указанные платформы принадлежат к сегменту IaaS.

Параллельно с ними развивались MaaS-платформы. Так, один из старейших инструментов этого типа - xCAT, появившись как средство управления высокопроизводительными вычислительными кластерами, впоследствии стал поддерживать технологии виртуализации и управления «облаками».

Несколько позднее появились PaaS-системы с открытым исходным кодом. Первой из систем такого класса, получивших широкое распространение, стала платформа Cloud Foundry компании VMware - одного из лидеров в сфере разработки технологий виртуализации.

2.1. MaaS

В настоящее время под MaaS-платформами понимается около двух десятков программ [11, 16] с весьма различающимися функциями. Поскольку сам термин MaaS появился в 2012 году, в связи с разработкой компанией Canonical платформы Juju [17], то обсуждение вопросов классификации подобных платформ ещё далеко от завершения.

Мы будем понимать под MaaS-платформами инструменты, ведущие своё происхождение от программ управления вычислительными кластерами, и рассмотрим одну из наиболее широко распространённых платформ такого рода - xCAT, первая версия которой была выпущена компанией IBM ещё в 1999 году.

2.1.1 xCAT

xCAT (Extreme Cluster/Cloud Administration Toolkit) [18] представляет собой инструмент управления кластерами высокопроизводительных вычислений, Grid-системами и «облаками».

Основные возможности xCAT:

- позволяет запускать операционные системы на физических и виртуальных машинах: RHEL, CentOS, Fedora, SLES, Ubuntu, AIX, Windows, VMware, KVM, PowerVM, PowerKVM, zVM;
- предоставляет скрипты для установки, stateless, statelite, iSCSI или для клонирования;
- поддерживает системы удалённого управления: lights-out, консоль и распределённые оболочки;
- настройка и управление узлами: DNS, HTTP, DHCP, TFTP, NFS;
- поддерживает установку на жёсткий диск, запуск без диска (diskless), запуск без диска с наименьшим состоянием бита (statelite), установку на iSCSI (специальная аппаратура не требуется);

- поддержка гипервизоров (VMWare, KVM, Xen, PowerVM, ZVM) и технологии Docker.

Система реализована на языке Perl и распространяется под лицензией Eclipse Public License.

Существенным преимуществом xCAT является хорошая масштабируемость: разработчики декларируют поддержку кластеров, состоящих из тысячи и более узлов. Кроме того, поскольку разработка xCAT координируется IBM, то данное ПО имеет преимущества при установке на оборудование, произведенное этой компанией [18], по сравнению с ПО сторонних разработчиков. Заметим, что доля IBM на рынке мэйнфреймов составляет около 90% [19].

2.2. IaaS

Несколько упрощая, работа IaaS-платформы выглядит следующим образом. Пользователь делает запрос к головному серверу о выделении новой виртуальной машины с указанной ОС и другими параметрами. Сервер обрабатывает запрос и выбирает наименее загруженную машину. Существуют различные алгоритмы выбора узла. Например, запрос может направляться первому из узлов с имеющимися свободными ресурсами, вплоть до полного его заполнения (режим Greedy), либо нескольким узлам поочередно (режим Round-robin).

После того, как машина выбрана, сервер предоставляет ей образ указанной ОС и даёт указание на старт системы. Машина запускает образ, а пользователю возвращается IP-адрес его виртуального сервера. Примерно так работает инфраструктура EC2.

Сервисы AWS приобрели большую популярность, что привело к появлению ряда открытых реализаций сервисов, совместимых с EC2. Наибольшее распространение среди таких платформ получила платформа Eucalyptus.

2.2.1. Eucalyptus

Первая версия Eucalyptus (Elastic Utility Computing Architecture for Linking Your Programs To Useful Systems) была выпущена одноимённой компанией в 2008 году. В настоящее время платформой владеет Hewlett-Packard, а её полное название: HPE Helion Eucalyptus [20].

Eucalyptus представляет собой инфраструктуру для реализации модели облачных вычислений уровня IaaS, отличительными чертами которой являются [21]:

- интерфейс, совместимый с EC2 и S3 (веб-сервисы и интерфейс Query/REST);
- поддержка гипервизоров Xen, KVM и VMware ESX/ESXi;
- поддержка большинства дистрибутивов Linux;
- безопасное взаимодействие компонентов с использованием SOAP и WS-security;
- наличие развитых инструментов администрирования облака для управления системой и аккаунтинга пользователей;
- возможности объединения множества кластеров, каждый из которых располагается в отдельном сегменте сети, в единое «облако».

Платформа Eucalyptus состоит из пяти основных компонентов (см. рис. 2):

1. Контроллер узла (*Node Controller*, NC). Запускается на каждом узле, из которых состоит «облако», и отвечает за запуск, работу и остановку виртуальных машин;
2. Контроллер кластера (*Cluster Controller*, CC) управляет контроллерами узлов и принимает решение, на каких узлах будут запущены виртуальные машины;
3. Контроллер облака (*Cloud Controller*, CLC). Устанавливается на машине, имеющей доступ к внешней сети, выступает в роли головного интерфейса для доступа к облаку. Обрабатывает пользовательские запросы на запуск виртуальных машин и собирает данные о загруженности узлов от контроллеров кластеров;
4. Контроллер хранилища (*Storage Controller*, SC). Взаимодействует с контроллерами кластеров и узлов и управляет сохранением томов и их снимков в пределах определённого кластера. Если при этом требуется записать данные в область памяти, лежащую за пределами кластера, то такие данные записываются в Walrus, который доступен из любого кластера «облака». Данный контроллер является аналогом AWS Elastic Block Store;
5. Walrus — хранилище данных, доступное с помощью ReSTful и SOAP API, и представляющее собой свободный аналог сервиса хранения данных S3. Walrus позволяет хранить образы виртуальных машин, моментальные снимки томов, данные приложений и т.п.

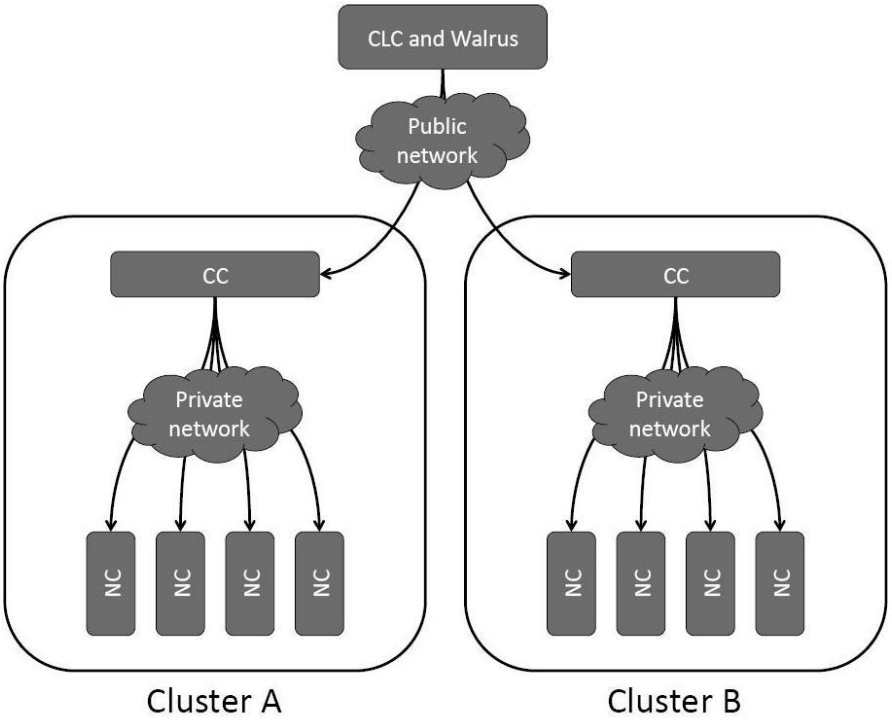

Рисунок 2. *Иерархическая структура платформы Eucalyptus [22]*

В отличие от таких систем, как CloudStack и OpenStack, платформа Eucalyptus нацелена, в первую очередь, на организацию работы частных и гибридных облаков. Создания публичных облаков, в которых организуется доступ «посторонних» пользователей, выходит за рамки Eucalyptus.

Исходный код Eucalyptus распространяется под лицензией GNU GPL версии 3.

2.2.2. OpenStack

Работы над проектом OpenStack начались летом 2010 года по инициативе NASA и хостинг-провайдера Rackspace, сделавших доступными исходные коды своих облачных платформ. По мысли инициаторов проекта, OpenStack должен со временем стать стандартной открытой облачной платформой.

В настоящее время OpenStack [23] представляет собой комплекс из 17 проектов свободного ПО, который может быть использован для создания инфраструктурных облачных сервисов - как частных, так и публичных. Все проекты комплекса распространяются под лицензией Apache License.

Главный компонент OpenStack - это OpenStack Compute или Nova, контроллер, управляющий работой виртуальных машин [24]. Nova обрабатывает запросы на создание виртуальных машин, соединяет их с внешним миром, следит за работоспособностью и распределением нагрузки на физические машины и каналы связи, реагирует на сбои и т.п. Nova написана на языке программирования Python и опирается на протокол обмена сообщениями AMQP.

OpenStack состоит из примерно двух десятков обособленных компонентов, среди которых в качестве основных можно выделить следующие:

1. контроллер облака (Cloud Controller) следит за состоянием системы и выступает в роли связующего звена всех остальных компонентов системы;
2. сервер API (API Server) реализует веб-интерфейс, позволяющий управлять контроллером облака;
3. контроллер вычислений (Compute Controller) отвечает за запуск виртуальных машин и их связь со всей остальной инфраструктурой «облака»;
4. хранилище (Object Store) предоставляет сервис хранения данных, совместимый с S3;
5. менеджер аутентификации (Auth Manager) обеспечивает аутентификацию и авторизацию пользователей;
6. контроллер томов (Volume Controller) позволяет подключать виртуальные устройства хранения к виртуальным машинам;
7. сетевой контроллер (Network Controller) создаёт виртуальные сети, позволяя виртуальным машинам взаимодействовать друг с другом и внешней сетью;
8. планировщик (Scheduler) отвечает за выбор подходящего контроллера вычислений для запуска новой виртуальной машины.

Обработка пользовательских запросов в OpenStack выглядит следующим образом [24]: пользователь посылает запрос на создание виртуальной машины к серверу API. После аутентификации и авторизации пользователя сервер API разбирает запрос и посылает его контроллеру облака, который инициирует три новых запроса: к сетевому контроллеру, к хранилищу и к планировщику. Сетевой контроллер выделяет IP-адрес для новой виртуальной машины (ВМ) и возвращает его контроллеру облака. В сетевом хранилище происходит поиск подходящего образа жёсткого диска для будущей ВМ, адрес которого возвращается контроллеру облака. Теперь, имея всё необходимое для создания новой ВМ, контроллер облака посылает запрос планировщику, который выбирает наиболее подходящий контроллер вычислений и отдает ему запрос на создание ВМ. После того, как новая ВМ будет запущена, контроллер облака завершает свою работу и сообщает серверу API об успехе всей операции.

Теперь пользователь может подключиться к ВМ, а всю деятельность по поддержанию её в работоспособном состоянии, выделении дискового пространства, маршрутизации сетевых пакетов и т.п. берёт на себя система.

Хотя разработчики OpenStack, в отличие от Eucalyptus, не ставят цели добиться совместимости интерфейсов с другими облачными платформами, тем не менее в рамках OpenStack развиваются проекты EC2 API и GCE API, направленные на разработку интерфейсов, совместимых с EC2 и Google Compute Engine соответственно.

OpenStack поддерживает развёртывание поверх кластера, работающего под управлением xCat, а также развивает собственный проект Ironic для непосредственной установки «облака» на компьютерное оборудование [25].

Более подробно устройство, принципы и практика работы с OpenStack описаны в [26].

2.2.3. Apache CloudStack

Apache CloudStack [27] - программная платформа, предназначенная для создания облачной инфраструктуры IaaS и позволяющая автоматизировать развёртывание, настройку и поддержание частной, гибридной или публичной облачной инфраструктуры.

CloudStack не зависит от типа гипервизора и позволяет использовать в единой облачной инфраструктуре одновременно Xen (XenServer и Xen Cloud Platform), KVM, Oracle VM (VirtualBox) и VMware (это свойство платформы называется «hypervisor agnostic»). CloudStack позволяет организовать работу как публичного IaaS-сервиса, сходного с Amazon EC2, так и частной облачной инфраструктуры, развёрнутой на локальных серверах и обслуживающей нужды конкретного предприятия. Облачная инфраструктура на базе CloudStack в простейшем случае состоит из одного управляющего сервера и набора вычислительных узлов, на которых организуется выполнение гостевых ОС в режиме виртуализации.

Основные возможности платформы [28, 29]:

- поддержка API основных облачных платформ, в частности, EC2, Citrix Cloud Center (C3) и vCloud API, а также стандарта OCCI;
- поддержка полной изоляции вычислительных, сетевых и дисковых ресурсов;
- поддержка автоматического выделения и ограничения ресурсов;
- наличие инструментов для генерации отчетов и мониторинга в режиме реального времени;
- веб-интерфейс, основанный на использовании технологии AJAX;
- инструменты, обеспечивающие наглядное управление инфраструктурой и выполнение ежедневных задач;
- возможность организации сервиса, обеспечивающего предоставление в аренду вычислительных ресурсов;
- поддержка виртуализации сети через изоляцию сегментов сети в отдельные VLAN;
- предоставление вычислительных ресурсов по запросу, в зависимости от создаваемой виртуальным окружением нагрузки;

- полная автоматизация распределения места для хранения данных, вычислительных и сетевых ресурсов для всей физической инфраструктуры, включая возможность определения политики выделения ресурсов и поддержку балансировки нагрузки;
- наличие средств для управления созданием «снимков» (snapshot) окружений и резервного копирования;
- средства для обеспечения отказоустойчивости, поддерживающие автоматическое восстановление виртуальных машин после сбоя сервера, на котором они выполнялись.

CloudStack обладает гибкими возможностями масштабирования, поддерживает развёртывание инфраструктур, обслуживающих тысячи хостов и позволяет управлять облачными системами, которые охватывают несколько территориально разделённых дата-центров.

2.2.4. OpenNebula

OpenNebula [30] представляет собой открытую и расширяемую IaaS -платформу, позволяющую развернуть на уже имеющихся серверах частный, публичный или гибридный облачный сервис, сходный по своим возможностям с Amazon EC2. На физическом сервере и кластере одновременно можно использовать разные гипервизоры (Xen, KVM, VMware ESX/ESXi). В качестве «гостевых» ОС могут работать любые из систем, поддерживаемых этими гипервизорами.

Модульная архитектура позволяет интегрировать OpenNebula с любой платформой виртуализации, хранилищем данных или менеджером управления. Поддерживаются все присущие облакам технологии и функции, включая Live Migration (виртуальную машину можно легко перенести на другой сервер) [31].

В OpenNebula реализован собственный API, а также поддерживаются API EC2, OCCI и vCloud.

Более подробно устройство OpenNebula описано в [9].

2.2.5. Nimbus

Открытая платформа Nimbus [32] предназначена для создания облачных сервисов уровня IaaS, совместимых с EC2 и S3, и ориентирована на использование в научных сообществах.

Nimbus поддерживает гипервизоры Xen и KVM, а также планировщики виртуальных машин Portable Batch System и Oracle Grid Engine, обеспечивая все основные функции, присущие облачным IaaS-платформам. Исходный код системы распространяется под лицензией Apache License, версии 2.

2.2.6. OpenQRM

OpenQRM [33] - свободное ПО с открытым исходным кодом, позволяющее управлять инфраструктурой центров обработки данных и создавать частные, публичные и гибридные облачные системы уровня IaaS.

Разработчики платформы декларируют приверженность принципу строгого разделения аппаратных средств (физических серверов и виртуальных машин) и программного обеспечения (образов ОС) [33]. Любое оборудование при этом понимается как вычислительный ресурс, который подлежит замене, без необходимости перенастройки ПО.

Особенности OpenQRM:

- возможность интеграции со всеми основными технологиями создания хранилищ данных (как открытыми, так и коммерческими);
- поддержка ОС Windows, Linux, OpenSolaris и FreeBSD;
- поддержка большого числа гипервизоров и контейнеров: KVM, Xen, Citrix XenServer, VMWare ESX/ESXi, lxc, OpenVZ и VirtualBox;
- поддержка настройки гибридных «облаков»;
- поддерживает популярные открытые инструменты управления облачным сервисом - Puppet, Nagios/Icinga и collectd;
- свыше 50 плагинов, расширяющих возможности системы и поддерживающих интеграцию с инфраструктурой пользователя.

Поддерживается сообществом разработчиков и компанией openQRM Enterprise (ФРГ). Распространяется под лицензией GNU GPL.

2.3. PaaS

Если IaaS предоставляет пользователю «голый» виртуальный компьютер, который ещё только предстоит настроить, то PaaS обеспечивает доступ к «компьютеру» с уже установленной и настроенной ОС, системными программами, языками программирования, средствами разработки, СУБД и т.п.

Появление PaaS-платформ стало закономерным этапом развития облачных инфраструктур, направленных на решение типовых задач, например, на предоставление услуг веб-хостинга.

2.3.1. Cloud Foundry

Cloud Foundry [34] - открытая PaaS-платформа, которая позволяет сформировать инфраструктуру для выполнения в облачных окружениях конечных приложений на языках Java (с поддержкой фреймворка Spring), Groovy (Grails), Ruby (Rails, Sinatra), JavaScript (Node.js), Scala, а также других языках, работающих в Java Virtual Machine. Платформа даёт возможность работы совместно с контейнерной виртуализацией на базе Docker и поддерживает функционирование поверх развёрнутых IaaS-платформ VMware vSphere, Amazon Web Services и OpenStack, а также развертывание на локальных системах. Из СУБД, Cloud Foundry позволяет работать с MySQL, Redis и MongoDB.

Cloud Foundry была разработана компанией VMware. Код платформы был открыт в 2011 году и с тех пор распространяется под лицензией Apache License 2.0. В настоящее время платформа разрабатывается фондом Cloud Foundry, ключевыми учредителями которого являются компании EMC, HP, IBM, Intel, Pivotal, SAP и VMware.

Существует также версия Pivotal Cloud Foundry, разрабатываемая компанией Pivotal [35], а также основанная на коде Cloud Foundry платформа HPE Helion Stackato, разрабатываемая Hewlett-Packard [36].

2.3.2. OpenShift

OpenShift [37] - облачная PaaS-платформа, разработанная компанией Red Hat, и ориентированная на разработку веб-приложений и организацию хостинга.

В настоящее время Red Hat поддерживает несколько видов этой платформы. Код OpenShift Origin открыт и распространяется под лицензией Apache License 2.0.

OpenShift предназначен для установки программ, работающих на Red Hat Enterprise Linux. Основные языки программирования: JavaScript, Ruby, Python, PHP, Perl, Java, Haskell и .NET. Среди СУБД OpenShift поддерживает MySQL, PostgreSQL, MongoDB и SQLite. Платформа поддерживает ряд популярных фреймворков для разработки веб-приложений: Rack (Ruby), WSGI (Python), PSGI (Perl) и Node.js (JavaScript). Кроме них в базовый набор приложений входят также фреймворки Laravel, CodeIgniter, CakePHP, Ruby on Rails, Django, Perl Dancer, Flask, Sinatra, Tornado и Web2py.

Платформой OpenShift используется концепция «картриджей» - подключаемых модулей [38], служащих для расширения возможностей системы. Пользуясь интерфейсом OpenShift Cartridge API, разработчики могут добавлять в систему поддержку дополнительных языков программирования, СУБД и компонентов промежуточного ПО (middleware). Кроме того, Red Hat поддерживает репозитории приложений, позволяющий пользователю загружать нужные ему языки, средства разработки, СУБД и т.п.

2.3.3. Cloudify

Платформа Cloudify [39] основана на TOSCA (Topology and Orchestration Specifications for Cloud Applications) - стандарте и языке описания облачных сервисов, разработанном фондом OASIS.

Целью выработки стандарта TOSCA является расширение возможностей по переносимости облачных приложений. По мысли разработчиков [40], TOSCA позволит описывать взаимодействие приложений и инфраструктуры облачных сервисов, взаимосвязь между компонентами сервиса и поведение самих сервисов, независимо от поставщика услуг или разработчика конкретной облачной технологии.

Платформа Cloudify может быть развёрнута поверх OpenStack, AWS, CloudStack, Microsoft Azure и VMware.

2.3.3. Apache Stratos

Apache Stratos [41] - расширяемый PaaS-фреймворк, который предоставляет средства для запуска приложений Apache Tomcat, PHP и MySQL в специальных окружениях, запущенных поверх распространённых облачных инфраструктур. Stratos предоставляет готовое облачное окружение для разработки, тестирования и запуска масштабируемых приложений, беря на себя выполнение таких задач, как автоматическое управление ресурсами, обеспечение оптимального распределения нагрузки, мониторинг и биллинг.

Среди основных особенностей Stratos:

- простота развертывания PaaS-инфраструктуры (тестовую конфигурацию можно запустить на компьютере разработчика);
- наличие средств поддержки многопользовательских (multi-tenant) PaaS-хостингов приложений, сходных с Google App Engine. Платформа предоставляет средства для контроля потребления ресурсов и координации списание имеющихся на лицевом счете средств;
- упрощённые методы формирования картриджей (модулей) для запуска новых типов приложений. Stratos можно быстро адаптировать для поддержки новых языков программирования, операционных систем и СУБД;
- нейтральная модель программирования, которая не требует переработки программ, запускаемых в PaaS;
- автоматическое масштабирование при росте потребности в ресурсах. Использование Complex Event Processor (CEP) для принятия решений о перераспределении ресурсов в режиме реального времени на основании правил и состояния системы;
- возможность распределения инфраструктуры по различным облачным платформам. Например, при недостатке ресурсов в локальной частном облаке можно задействовать мощности публичных облачных систем;
- поддержка работы поверх различных IaaS-окружений, созданных на основе OpenStack, CloudStack, Amazone ES2 SUSECloud и VMWare vCloud. Теоретически, Stratos может быть использован совместно с любой системой для которой доступен API Apache jclouds;
- средства оперативного восстановления после сбоев и обеспечения высокой доступности;
- наличие веб-интерфейса и интерфейса командной строки для администраторов инфраструктуры и пользователей. Для интеграции с другими облачными системами предоставляется REST API.

Исходный код платформы написан на языке Java и передан фонду Apache компанией WSO2 в июне 2013 года. Впоследствии, к разработке Stratos присоединились Cisco, Citrix и NASA.

Выводы

Рассмотрены существующие виды облачных сервисов, аппаратные, программные и организационные особенности их реализации. Проведен сравнительный анализ достоинств и недостатков существующих технологий создания облачных сервисов. Выделены проблемы и риски, актуальные с точки зрения пользователей облачного сервиса. Выполнен аналитический обзор свободных облачных аппаратно-программных платформ. Результаты обзора могут быть полезны при необходимости решения задачи выбора свободной облачной аппаратно-программной платформы в зависимости от требований к разрабатываемым программным комплексам, предполагающим реализацию облачного сервиса.

ЛИТЕРАТУРА

1. Риз Дж. Облачные вычисления. СПб.: БХВ-Петербург, 2011.
2. Srinivasan S. Cloud Computing Basics. Springer, 2014.
3. ГОСТ Р 53622–2009. Информационные технологии. Информационно-вычислительные системы. Стадии и этапы жизненного цикла, виды и комплектность документов. М.: Стандартинформ, 2011. 12 с.
4. Mell P., Grance T. The NIST Definition of Cloud Computing (September 2011). SP 800-145. [Электронный ресурс] // National Institute of Standards and Technology [Офиц. сайт]. URL: http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-145.pdf (дата обращения: 13.05.2016).
5. Радченко Г.И. Распределенные вычислительные системы. Челябинск: Фотохудожник, 2012. 108 с.
6. SETI@home [Электронный ресурс] // SETI@home [Офиц. сайт проекта]. URL: http://setiathome.ssl.berkeley.edu/ (дата обращения: 12.05.2016).
7. UML – Deployment Diagrams. [Электронный ресурс] // Tutorials Point [Офиц. сайт]. URL: http://www.tutorialspoint.com/uml/uml_deployment_diagram.htm (дата обращения: 12.05.2016).
8. Ткаченко В. А. Облачные вычисления и сервисы на базе облачных вычислений [Электронный ресурс] // Обучение в Интернет [Персональный сайт]. URL: http://www.lessons-tva.info/archive/nov031.html (дата обращения: 12.05.2016).
9. Toraldo G. OpenNebula 3 Cloud Computing. Packt Publishing, 2012.
10. Боттомли Дж. Ажиотаж вокруг контейнеров [Электронный ресурс] // Журнал сетевых решений / LAN. №10, 2014. URL: http://www.osp.ru/lan/2014/10/13043208/ (дата обращения: 13.05.2016).
11. Черняк Л. HPC, пятнадцать лет эволюции [Электронный ресурс] // Открытые системы. №4, 2008. URL: http://www.osp.ru/os/2008/04/5114370/ (дата обращения: 25.05.2016).
12. Open Cloud Computing Interface. [Электронный ресурс] // Open Cloud Computing Interface Working Group [Офиц. сайт]. URL: http://occi-wg.org/ (дата обращения: 14.05.2016).
13. Docker – Build, Ship, and Run Any App, Anywhere. [Электронный ресурс] // Docker [Офиц. сайт]. URL: https://www.docker.com/ (дата обращения: 14.05.2016).
14. Курс «Технологии облачных вычислений». Практическая работа 1: Основы облачных вычислений. [Электронный ресурс] // Национальный открытый университет «ИНТУИТ» [Офиц. сайт]. URL: http://www.intuit.ru/studies/courses/3508/750/lecture/27413?page=4#sect12 (дата обращения: 14.05.2016).
15. Amazon Web Services (AWS) – сервисы облачных вычислений [Электронный ресурс] // Amazon Web Services [Офиц. сайт]. URL: http://aws.amazon.com/ru/ (дата обращения: 13.05.2016).

16. Puppet, Chef, Orchestration and DevOps [Электронный ресурс] // Au courant technology [Персональный сайт]. URL: https://aucouranton.com/2013/07/31/puppet-chef-orchestration-and-devops/ (дата обращения: 25.05.2016).

17. MAAS: Metal As A Service [Электронный ресурс] // MAAS dev documentation [Офиц. сайт]. URL: https://maas.ubuntu.com/docs/ (дата обращения: 25.05.2016).

18. What is xCAT? [Электронный ресурс] // xCAT [Офиц. сайт]. URL: http://xcat.org/ (дата обращения: 25.05.2016).

19. IBM Tightens Stranglehold Over Mainframe Market; Gets Hit with Antitrust Complaint in Europe [Электронный ресурс] // CCIA: Computer & Communications Industry Association [Офиц. сайт]. URL: http://www.ccianet.org/2008/07/ibm-with-another-mainframe-antitrust-complaint-in-europe/ (дата обращения: 25.05.2016).

20. HPE Helion Eucalyptus. Open source hybrid cloud software for AWS users [Электронный ресурс] // Hewlett Packard Enterprise [Офиц. сайт]. URL: http://www8.hp.com/us/en/cloud/helion-eucalyptus-overview.html (дата обращения: 12.05.2016).

21. По дороге с облаками. Пошаговое руководство по созданию инфраструктуры облачных вычислений на базе Eucalyptus [Электронный ресурс] // Хакер. 2011, №11. URL: http://xakep-archive.ru/xa/131/122/1.htm (дата обращения: 13.05.2016).

22. Облако своими руками, или возможности Eucalyptus [Электронный ресурс] // OpenQuality.ru. Качество программного обеспечения [Интернет-портал]. URL: http://blog.openquality.ru/eucalyptus-cloud/ (дата обращения: 12.05.2016).

23. OpenStack Open Source Cloud Computing Software [Электронный ресурс] // OpenStack [Офиц. сайт]. URL: http://www.openstack.org/ (дата обращения: 16.05.2016).

24. Облако, открытое для всех: OpenStack - обзор и первые впечатления [Электронный ресурс] // Хакер. 2011, №3. URL: https://xakep.ru/2011/03/14/55033/ (дата обращения: 16.05.2016).

25. Ironic [Электронный ресурс] // OpenStack [Офиц. сайт]. URL: https://wiki.openstack.org/wiki/Ironic (дата обращения: 25.05.2016).

26. Shrivastwa A., Sarat S. Learning OpenStack. Packt Publishing, 2015.

27. Apache CloudStack. Open Source Cloud Computing [Электронный ресурс] // Apache CloudStack [Офиц. сайт]. URL: http://www.cloudstack.org/ (дата обращения: 16.05.2016).

28. Руденко А. CloudStack 4. Создание и запуск виртуальных машин [Электронный ресурс] // iVirt-it.ru [Персональный сайт]. URL: http://ivirt-it.ru/cloudstack-4-part-3-launchvm/ (дата обращения: 16.05.2016).

29. Sabharwal N., Shankar R. Apache CloudStack Cloud Computing. Packt Publishing, 2013.

30. OpenNebula | Flexible Enterprise Cloud Made Simple [Электронный ресурс] // OpenNebula [Офиц. сайт]. URL: http://opennebula.org/ (дата обращения: 16.05.2016).

31. Колыбель облаков. Пошаговое руководство по развертыванию IaaS-сервиса на базе OpenNebula [Электронный ресурс] // Хакер. 2013, №8. URL: https://xakep.ru/2013/08/28/61155/ (дата обращения: 16.05.2016).

32. Nimbus [Электронный ресурс] // Nimbus [Офиц. сайт]. URL: http://www.nimbusproject.org/ (дата обращения: 16.05.2016).

33. OpenQRM [Электронный ресурс] // Wikipedia, the free encyclopedia [Интернет-портал]. URL: https://en.wikipedia.org/wiki/OpenQRM (дата обращения: 16.05.2016).

34. Cloud Foundry | The Industry Standard For Cloud Applications [Электронный ресурс] // Cloud Foundry [Офиц. сайт]. URL: https://www.cloudfoundry.org/ (дата обращения: 17.05.2016).

35. Pivotal Cloud Foundry [Электронный ресурс] // Pivotal [Офиц. сайт]. URL: http://pivotal.io/platform (дата обращения: 17.05.2016).

36. HPE Helion Stackato [Электронный ресурс] // Hewlett Packard Enterprise [Офиц. сайт]. URL: http://www8.hp.com/us/en/cloud/helion-devplatform-overview.html (дата обращения: 17.05.2016).

37. OpenShift: PaaS by Red Hat, Built on Docker and Kubernetes [Электронный ресурс] // OpenShift [Офиц. сайт]. URL: https://www.openshift.com/ (дата обращения: 17.05.2016).

38. Miller A. Implementing OpenShift. Packt Publishing, 2013.

39. Pure-Play Cloud Orchestration & Automation Based on TOSCA [Электронный ресурс] // Cloudify [Офиц. сайт]. URL: http://getcloudify.org/ (дата обращения: 17.05.2016).

40. OASIS Topology and Orchestration Specification for Cloud Applications (TOSCA) TC [Электронный ресурс] // OASIS. Advancing open standards for the information society [Офиц. сайт]. URL: https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=tosca (дата обращения: 17.05.2016).

41. Apache Stratos – Open Enterprise PaaS [Электронный ресурс] // Apache Stratos [Офиц. сайт]. URL: http://stratos.apache.org/ (дата обращения: 17.05.2016).

Belonozhko Pavel Petrovich
Bauman Moscow state technical university, Russia, Moscow
E-mail: byelonozhko@mail.ru

Belous Valentina Vladimirovna
Bauman Moscow state technical university, Russia, Moscow
E-mail: Walentina.belous@gmail.com

Kutsevich Nadezhda Aleksandrovna
RTSoft, Russia, Moscow
E-mail: rtsoft@rtsoft.ru

Khramov Dmitriy Aleksandrovich
Institute of technical mechanics NASU and SSAU, Ukraine, Dnepropetrovsk
E-mail: dkhramov@mail.ru

Free cloud hardware and software platform. Analytical review

Abstract. The problem of choosing a free cloud-based hardware and software platform for creating a software system for quantitative assessment of interdisciplinary, meta-cognitive and metacreation skills of pupils are considered. To choose the platform an analysis of the main types of cloud services and directions for their use are performed. Issues arising from the use of cloud platforms and ways of their solution are given. Features of a number of free cloud-based software platforms are analyzed and the platform that best meets the project requirements is selected.

Keywords: cloud computing; metal-as-a-service; platform-as-a-service journal's paper; hypervisor; container

REFERENCES

1. Reez J. Cloud computing. St. Petersburg: BHV-Petersburg, 2011. (in Russian).
2. Srinivasan S. Cloud Computing Basics. Springer, 2014.
3. ГОСТ Р 53622–2009. Information technology. Information-computational system. The stages of the life cycle, types and completeness of documents. Moscow: Standartinform, 2011. 12 p. (in Russian).
4. Mell P., Grance T. The NIST Definition of Cloud Computing (September 2011). SP 800-145. Available at: http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-145.pdf, accessed 13.05.2016.
5. Radchenko G.I. A distributed computing system. Chelyabinsk: Fotohudozhnik, 2012. 108 p. (in Russian).
6. SETI@home. Available at: http://setiathome.ssl.berkeley.edu/, accessed 12.05.2016.
7. UML – Deployment Diagrams. Available at: http://www.tutorialspoint.com/uml/uml_deployment_diagram.htm, accessed 12.05.2016.
8. Tkachenko V.A. Cloud computing and services based on cloud computing. Available at: http://www.lessons-tva.info/archive/nov031.html, accessed 12.05.2016. (in Russian).
9. Toraldo G. OpenNebula 3 Cloud Computing. Packt Publishing, 2012.

10. Bottomly J. The hype around containers // Network solutions magazine/LAN. № 10, 2014. Available at: http://www.osp.ru/lan/2014/10/13043208/, accessed 13.05.2016. (in Russian).
11. Cheryak L. HPC fifteen years of evolution // Open systems. №4, 2008. Available at: http://www.osp.ru/os/2008/04/5114370/, accessed 25.05.2016. (in Russian).
12. Open Cloud Computing Interface. Available at: http://occi-wg.org/, accessed 14.05.2016.
13. Docker – Build, Ship, and Run Any App, Anywhere. Available at: https://www.docker.com/, accessed 14.05.2016.
14. Course «Cloud computing Technology». Practical work 1: The basics of cloud computing // National Open University «INTUIT». Available at: http://www.intuit.ru/studies/courses/3508/750/lecture/27413?page=4#sect12, accessed 14.05.2016. (in Russian).
15. Amazon Web Services (AWS) – cloud computing services. Available at: http://aws.amazon.com/ru/, accessed 13.05.2016. (in Russian).
16. Puppet, Chef, Orchestration and DevOps. Available at: https://aucouranton.com/2013/07/31/puppet-chef-orchestration-and-devops/, accessed 25.05.2016.
17. MAAS: Metal As A Service. Available at: https://maas.ubuntu.com/docs/, accessed 25.05.2016.
18. What is xCAT? Available at: http://xcat.org/, accessed 25.05.2016.
19. IBM Tightens Stranglehold Over Mainframe Market; Gets Hit with Antitrust Complaint in Europe. Available at: http://www.ccianet.org/2008/07/ibm-with-another-mainframe-antitrust-complaint-in-europe/, accessed 25.05.2016.
20. HPE Helion Eucalyptus. Open source hybrid cloud software for AWS users. Available at: http://www8.hp.com/us/en/cloud/helion-eucalyptus-overview.html, accessed 12.05.2016.
21. On the road with clouds. Step-by-step guide to creating the infrastructure for cloud computing based on Eucalyptus // Hacker. 2011, №11. Available at: http://xakep-archive.ru/xa/131/122/1.htm, accessed 13.05.2016. (in Russian).
22. Cloud made with your own hands, or capabilities of Eucalyptus. Available at: http://blog.openquality.ru/eucalyptus-cloud/, accessed 12.05.2016. (in Russian).
23. OpenStack Open Source Cloud Computing Software. Available at: http://www.openstack.org/, accessed 16.05.2016.
24. Cloud, open to all: OpenStack - review and first impressions. Available at: https://xakep.ru/2011/03/14/55033/, accessed 16.05.2016. (in Russian).
25. Ironic. Available at: https://wiki.openstack.org/wiki/Ironic, accessed 25.05.2016.
26. Shrivastwa A., Sarat S. Learning OpenStack. Packt Publishing, 2015.
27. Apache CloudStack. Open Source Cloud Computing. Available at: http://www.cloudstack.org/, accessed 16.05.2016.
28. Rudenko A. CloudStack 4. Create and run virtual machines. Available at: http://ivirt-it.ru/cloudstack-4-part-3-launchvm/, accessed 16.05.2016. (in Russian).

29. Sabharwal N., Shankar R. Apache CloudStack Cloud Computing. Packt Publishing, 2013.

30. OpenNebula. Flexible Enterprise Cloud Made Simple. Available at: http://opennebula.org/, accessed 16.05.2016.

31. Cradle of the clouds. Step-by-step guide to deploying IaaS service based on OpenNebula. Available at: https://xakep.ru/2013/08/28/61155/, accessed 16.05.2016. (in Russian).

32. Nimbus. Available at: http://www.nimbusproject.org/, accessed 16.05.2016.

33. openQRM – Wikipedia, the free encyclopedia. Available at: https://en.wikipedia.org/wiki/OpenQRM, accessed 16.05.2016.

34. Cloud Foundry. The Industry Standard For Cloud Applications Available at: https://www.cloudfoundry.org/, accessed 17.05.2016.

35. Pivotal Cloud Foundry Available at: http://pivotal.io/platform, accessed 17.05.2016.

36. HPE Helion Stackato Available at: http://www8.hp.com/us/en/cloud/helion-devplatform-overview.html, accessed 17.05.2016.

37. OpenShift: PaaS by Red Hat, Built on Docker and Kubernetes. Available at: https://www.openshift.com/, accessed 17.05.2016.

38. Miller A. Implementing OpenShift. Packt Publishing, 2013.

39. Pure-Play Cloud Orchestration & Automation Based on TOSCA Available at: http://getcloudify.org/, accessed 17.05.2016.

40. OASIS Topology and Orchestration Specification for Cloud Applications (TOSCA) TC. Available at: https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=tosca, accessed 17.05.2016.

41. Apache Stratos – Open Enterprise PaaS. Available at: http://stratos.apache.org/, accessed 17.05.2016.