Инженерный вестник Дона, №9 (2021)
ivdon.ru/ru/magazine/archive/n9y2021/7189

Экспериментальная реализация гибридной модели облачного сервиса на принципах экономики совместного потребления

А.М. Воробьёв, Ю.В. Боганюк, М.С. Воробьева, А.О. Козлова

Тюменский Государственный Университет, Тюмень

Аннотация: В статье рассматривается экспериментальная реализация облачного центра обработки данных на основе принципов экономики совместного потребления. Приведен алгоритм и реализация управления ресурсами акторов совместного потребления, проведена качественная оценка работоспособности системы, выполнены базовые тесты производительности веб-приложений, развернутых по предложенной модели.
Ключевые слова: 10 облачные технологии, веб-приложение, экономика общественного потребления, виртуализация, виртуальная машина, системное администрирование, облачный кластер, передача данных, облачный сервис, IaaS.

Введение

Облачные вычисления уверенно вошли в современную отрасль информационных технологий и прочно заняли определенное место, обоснованное экономическими и техническими выгодами. Действительно, целые классы приложений и инсталляций приобретают повышенную экономическую эффективность в случае их переноса из корпоративной on-premise инфраструктуры в инфраструктуру облачного провайдера различного уровня, будь то IaaS (Infrastructure-as-a-Service – инфраструктура как услуга), PaaS (Platform-as-a-Service – платформа как услуга) или комбинации подобных подходов [1].

Основная особенность рынка облачных вычислений заключается в достаточно высокой планке входа на данный рынок: построение современной платформы облачных вычислений требует высоких капитальных затрат еще на этапе проектирования центра обработки данных, многократного резервирования каналов передачи данных, построение дублирующих систем вычисления и хранения данных, наличия высокоскоростных каналов связи.

Принципы экономики совместного потребления (sharing economy) демонстрируют возможность снижения начальных затрат при построении бизнеса определенного типа [2]. Выделяют следующие направления работы данного подхода:

1) стартапы;
2) дополнительные направления бизнеса крупных компаний;
3) компании, создающие прибавочную стоимость на транзакциях между клиентами;
4) прямое взаимодействие клиентов друг с другом на некоторой площадке.

Принципы sharing economy создают преимущества для всех участников процесса: поставщиков, потребителей, организаторов взаимодействия. Отмечается вовлеченность в экономику совместного потребления как крупных поставщиков, так и обычных физических лиц, в зависимости от гибкости модели. [3]

Основным сдерживающим фактором прямого взаимодействия двух субъектов является отсутствие доверия, уверенности в добросовестности участника сделки. Роль промежуточной стороны состоит в создании механизмов укрепления доверия и/или страхования рисков [4]. В исследуемом вопросе построения экспериментального прототипа на принципах совместного потребления именно доверие собственника программного кода к конечным исполнителям может стать основным сдерживающим фактором популярности платформы.

Таким образом, на данном этапе работ необходимо подтвердить возможность построения прототипа в условиях взаимодействия акторов экономики совместного потребления. Вопросы построения цепочки и механизмов доверия следует решать на основе устойчивого технического

решения для такого взаимодействия, когда ясны и понятны возможные ограничения и вызовы различного характера.

Исходя из вышеизложенного, модель прототипа выглядит следующим образом: проект концентрируется на организации экосистемы облачных вычислений, где непосредственные вычислительные мощности (ресурс, товар), находятся во владении других лиц (различного типа), а роль проекта состоит в распределении задач, агрегации результатов и построении доверия между поставщиками и потребителями подобного решения.

Цель исследования – экспериментальная проверка возможности построения платформы для выполнения облачных вычислений на основе принципов экономики совместного потребления.

Методология построения облачной платформы

Начальный облик прототипа облачной платформы с точки зрения типов хостов и их назначения выглядит следующим образом:

1) ядро системы – надежно реализованный программный сервис, отвечающий за ввод\вывод вычислительных мощностей в эксплуатацию, распределение задач и агрегацию результатов;

2) хост выполнения – предоставленный хост для выполнения вычислений, получающий указания от ядра.

В качестве целевого назначения прототипа выбирается реализация веб-приложений с высокой доступностью и балансировкой нагрузки:

1) система принимает приложения в качестве контейнеров Docker;

2) внутри контейнера имеется простейший веб-сайт со всем необходимым для работы набором служб – веб-сервер, сервера базы данных различных типов, средства аутентификации и т.п.;

3) для обеспечения балансировки нагрузки использует веб-сервер NGINX в бесплатной редакции – применение пассивных проверок работоспособности.

Особенностью sharing economy подхода к построению бизнес-модели является наличие широкого разнообразия ресурса, представляемого акторами. Разнообразие заключается в количественных и качественных характеристиках, степени износа ресурса и уровне его доступности.

Акторы предоставляют не полностью свободные аппаратные мощности, а некоторую долю времени работы существующих вычислительных систем, что затрагивает вопрос резервирования (разной степени жесткости) ресурсов для интересов облачной системы достаточно остро.

Базовым подходом управления ресурсами в современных ИТ-системах является виртуализация. Современные системы виртуализации поддерживают коллективный доступ виртуальных машин не только к процессору и памяти, но и к компонентам сложной периферии [5].

Виртуализация является основным механизмом управления ресурсами в системах облачных вычислений. Более того, виртуализация как концепт прочно закрепилась в облачных технологиях в том числе и как программно-определяемая сеть [6]. Суть данной концепции состоит в абстракции сетевой инфраструктуры и предоставлении пользователю виртуальной сетевой архитектуры, обычно в рамках Virtual Private Cloud Network или подобного подхода. Аналогично этому виртуализация систем хранения данных привела к построению идеологии программно-определяемого хранилища данных, абстрагирующее физическую и программную архитектуру в пользу предоставления конкретных услуг по хранению данных в рамках различных моделей облачных сервисов [7].

Следует отметить, что при использовании одного хоста облачного центра обработки данных (ЦОД), представленного в виде виртуальной машины на мощностях актора экономики общественного потребления, для хостинга нескольких приложений возникает вопрос распределения ресурсов между ними в соответствии с выбранной стратегией. В качестве дополнительной меры управления ресурсами уже внутри виртуальной машины применяются встроенные возможности систем контейнеризации в области управление ресурсами, реализованного в виде квот и лимитов [8].

Выбранный в рамках экспериментального прототипа подход «Виртуальная машина – контейнер с приложением» подразумевает наличие виртуальных машин с определенным количеством заявленных виртуальных ресурсов.

С учетом описанного, требуется реализовать способ создания идентичных (с точки зрения службы управления) виртуальных машин в условиях большого количества комбинаций аппаратного и программного обеспечения акторов. Следует начать с оценки существующих подходов к унификации как самого образа и конфигурации ВМ, так и с механизмов ввода и вывода ВМ из эксплуатации.

Задача унификации конфигурации ВМ решена в рамках инициативы Open Virtualization Format и имплементирована в большинство современных систем виртуализации. Существующие механизмы допускают конфигурацию параметров ВМ с целью более точного соответствия особенностям конкретного развертывания. Данный стандарт широко применяется в задачах унификации доступа к средам различных поставщиков, использующих отличающийся технологический стек [9].

Наиболее развитым средством унификации доступа к инфраструктуре виртуализации на сегодняшний день является решение libvirt, API виртуализации [10]. Libvirt поддерживает управление решениями

виртуализаций следующих поставщиков: Xen, Virtuozzo, VMWare ESXi, LCX, BHyve и Hyper-V. Все основные операции работы с ВМ унифицированы в рамках данного решения. Сама служба libvirt, по заявлению разработчиков, может выполнять на операционных системах семейства Linux, Windows, Mac OS X и других.

Следует отметить, что службы типа libvirt имеют серверное назначение и их применение предусматривается в составе интегрированных программных комплексов управления виртуализацией, что накладывает высокие требования к поддержке решений на базе libvirt. Подобное допустимо применять в решениях, использующих специальные кластеры виртуализации, т.е. на базе акторов, имеющих специальное оборудование и компетенции по его менеджменту.

В качестве альтернативы системы управления виртуализацией для решений на базе ПК следует рассмотреть свободные и бесплатные пользовательские решения по работе с виртуальными машинами. Наиболее популярным, бесплатным и свободным решением является Oracle VirtualBox.

Результаты

Проектирование

Реализация подразумевала применение системы frontend-backend, где начальной точкой входа в приложение является веб-сервер с функциями распределения запросов между серверами второго эшелона.

Серверы второго эшелона выполняют непосредственную полезную работу и возвращают результат frontend серверу, который направляет результат потребителю сервиса.

Исходя из изложенной схемы, предложена типизация хостов вычислений:

1) хост вычисления «Front»:

- наличие публичного IP-адреса;
- высокие требования к надежности работы;

2) хост вычисления «Back»:

- не требуется публичный IP-адрес;
- наличие достижимости до «Front».

Хост вычисления типа «Front» выступает точкой входа пользователя в приложение, что подразумевает повышенные требования к резервированию и отказоустойчивости. Несмотря на то, что хост типа «Front» не выполняет полезную работу самостоятельно, следует отметить, что необходимость обслуживать запросы большого количества клиентов повышает требования к вычислительной мощности хоста типа «Front».

Хост вычисления типа «Back» выступает непосредственным исполнителем полезной нагрузки веб-приложения. В данном экспериментальном прототипе веб-приложение является самодостаточной средой, не использующей сторонние службы и сервисы.

В рамках построения экспериментального решения вводятся используется термин «инсталляция» – набор, состоящий из хостов типа «Front» и «Back», объединенных в сеть с применением зашифрованных туннелей, используемый для доставки пользовательского приложения. Пользователь загружает файл приложения, указывает желаемое количество хостов типа «Front» и «Back», ядро системы формирует инсталляцию и запускает доставку приложения.

Общий формат работы предлагаемого облачного решения выглядит следующим образом (рис. 1):

1) для работы веб-приложения ядром формируются группа хостов типа «Front» и «Back», занятая в обслуживании одного приложения;

2) доменное имя приложения разрешается в адреса хостов типа «Front» – реализация балансировки и высокой доступности между хостами типа «Front»;

3) ядром формируются связи хостов типа «Front» и типа «Back»: каждый хост типа «Front» имеет надежный защищенный канал связи до каждого из хостов типа «Back»;

4) хосты типа «Back» получают на исполнение контейнер Docker с веб-приложением, следуют инструкции и пробрасывают внешние порты хоста во внутренние порты контейнера;

5) хосты типа «Front» получают конфигурацию сервера NGINX, обеспечивающую работу приложения с учетом балансировки нагрузки между всеми подключенными хостами типа «Back».

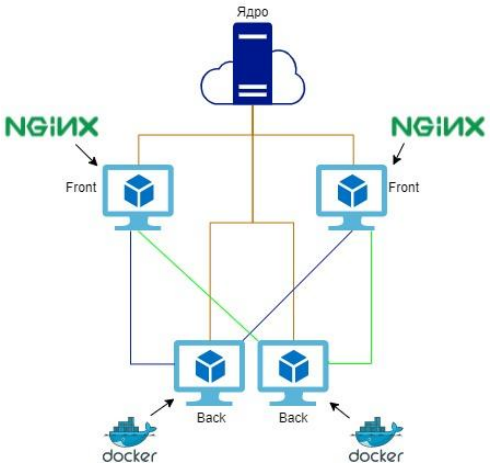

Рис. 1. – Схема взаимодействия прототипа

Технологии реализации ядра системы

Ядро системы представлено веб-службой на основе фреймворка FLASK, сервера баз данных PostgreSQL и языка Python. Обслуживание веб-приложения реализовано с помощью сервера NGINX. Ядро устанавливает и поддерживает туннель до каждого хоста обоих типов. Реализация туннелей выполнена на основе технологии WireGuard.

Служба реализует следующий функционал:

1) ввод в эксплуатацию ресурсов акторов: служба осуществляет прием регистрационной информации от скрипта настройки, запущенного в ВМ на стороне актора, генерацию ключей WireGuard, установку туннеля и осуществление тестового доступа к ВМ на стороне актора;

2) вывод ресурса из эксплуатации: служба осуществляет отключение ресурса, удаление развернутого экземпляра приложения, перенастройку конфигурации инсталляции с учетом уменьшения количества рабочих хостов типа «Back» или «Front», установленные туннели удаляются, ключи выводятся из эксплуатации;

3) загрузка контейнеров приложений в хранилище: прием от клиента системы контейнера с приложением, сохранение его в собственном хранилище виртуальной машины ядра;

4) создание инсталляции – комплексная процедура развертывания приложения, на вход принимается название приложения для развертывания (должно присутствовать в хранилище), количество хостов типа «Front» и «Back»:

- выбор подходящих под заявленные характеристики хостов типа «Front» и «Back»;
- формируются туннели: каждый хост типа «Front» поддерживает туннель с каждым хостом типа «Back»;
- хосты типа «Back» загружают контейнер и запускают экземпляр приложения с пробросом портов 5000-6000/tcp;
- хосты типа «Front» получают конфигурацию веб-сервера NGINX, направленную на балансировку нагрузки между хостами типа «Back»:

  i. количество ошибок, ведущее к временной остановке работы с backend – 1;

ii. время неактивности backend в случае ошибки – 20 секунд;

iii. балансировка осуществляет равномерно, вес каждого бэкенда принят как 1;

5) удаление инсталляции – вывод приложения из эксплуатации. Удаление приложения с хостов типа «Back», удаление конфигурации веб-сервера с хостов типа «Front», отключение туннелей между хостами типа «Front» и «Back».

Служба ядра снабжена упрощенным веб-интерфейсом для выполнения основных задач (рис. 2).

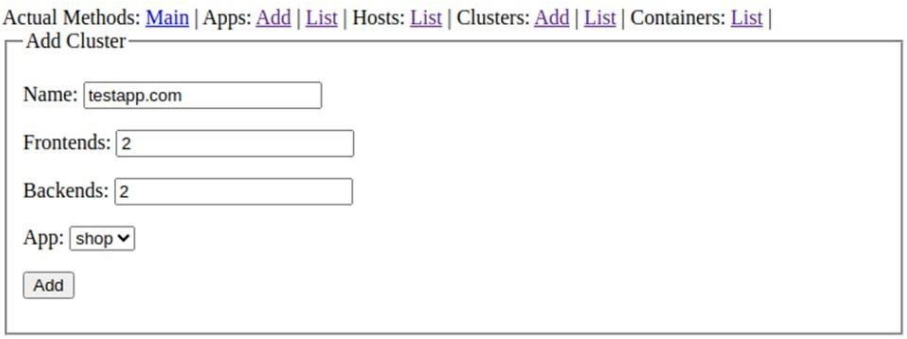

Рис. 2. – Веб-интерфейс службы ядра

Технологии реализации виртуальных машин

В качестве виртуальной машины (ВМ), используемой как среда для выполнения контейнеров, выбран образ на основе CentOS 8 актуальной версии. ВМ сформирована в минимальном варианте с применением динамического диска размером в 50 Гб, из которых в интересы ЦОД передан 1 Гб. Виртуальные машины, используемые в эксперименте, обладают 2 виртуальными ядрами центрального процессора и 512 мегабайтами оперативной памяти без разделения на память контейнера и память, используемую другими приложениями.

ВМ реализована в формате VirtualBox и в рамках эксперимента применяется на хостах типа «Back». Хосты типа «Front» представлены типовыми ВМ Google Cloud Platform. Характеристики данного типа ВМ:

1) 1 виртуальный процессор;
2) 3.75 Гб оперативной памяти
3) 20 Гб хранилища.
4) временный публичный IP-адрес.

Подключение ВМ к сервису организовано при помощи Python-скрипта «connect.py», совершающего запросы к API сервиса. В качестве параметров скрипт принимает на вход:

- имя узла;
- тип узла;
- IP-адрес для хостов типа «Front»;
- количество vCPU для хостов типа «Back»:
- объем оперативной памяти для хостов типа «Back»;
- объем дискового пространства для хостов типа «Back».

Процесс подключения происходит в соответствии с рисунком 3.

Помимо вышеописанного скрипт принимает флаг «--config». В случае его использования вместе с типом узла произойдет установка необходимых пакетов и конфигурация узла.

1) Общая конфигурация хостов включает:
   a. установку WireGuard;
   b. открытие порта, используемого WireGuard (51820/udp);
   c. создание пользователя «devops» с беспарольным sudo для взаимодействия с узлом при помощи Ansible;

d. запись открытого ssh-ключа в файл «authorized_keys» пользователя «devops».

2) Конфигурация для «Back» состоит из:

a. установки Docker (docker-ce);

b. открытия диапазона портов, используемых для доступа к контейнерам (5000-6000/tcp).

3) Конфигурация для «Front» состоит из:

a. установки Nginx;

b. открытия порта 80/tcp.

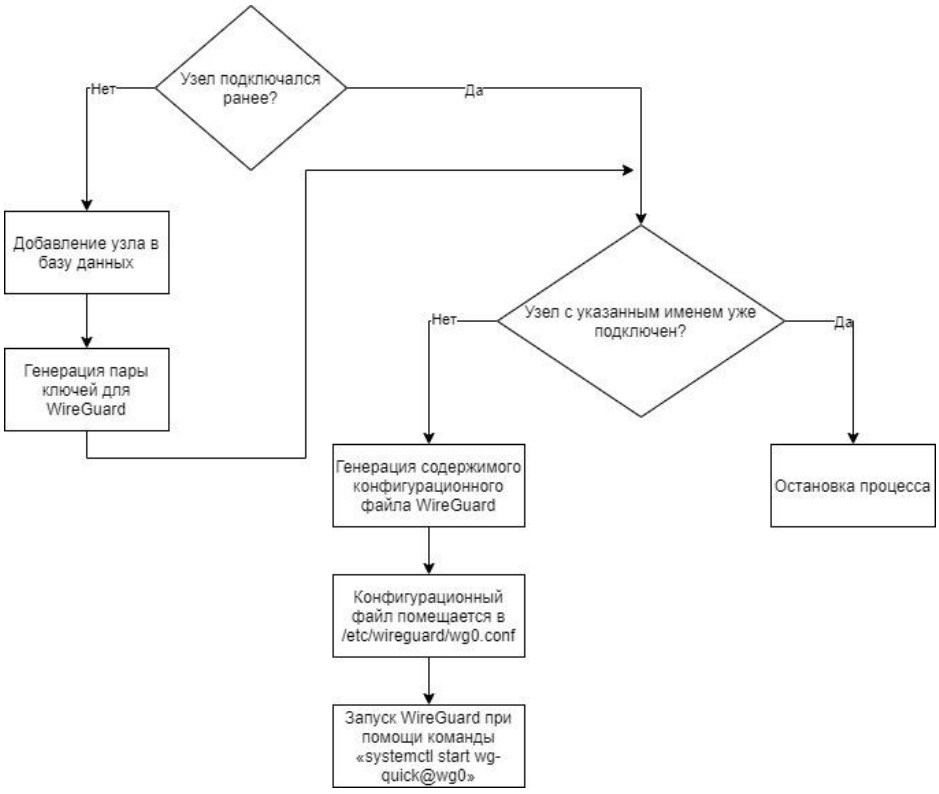

Рис. 3. – Схема подключения ВМ к сервису

Настройка SELinux в рамках прототипа не рассматривается, хотя она понадобится. Например, для использования Nginx в качестве reverse-proxy необходимо установить переменную «httpd_can_network_relay» в значение «1». Сейчас же для удобства отключим его командой «setenforce 0».

Обсуждение

В представленном виде прототип реализует функционал гибридного облачного сервиса развертывания веб-приложений на основе docker-контейнеров. Это популярный паттерн реализации веб-приложений, снимающих большое количество вопросов совместимости и версионности. Существующим коммерческим решением (с многократно большим функционалом) является Google Kubernetes Engine.

В качестве сравнительных решений был реализован Kubernetes-кластер на платформе Google Cloud Platform GKE с указанным выше приложением. Кластер состоит из 3 серверов. Измерялось среднее время выполнения 1, 10 и 25 запросов типа GET и POST. Тестирование проводилось 15 раз для каждой указанной ситуации, в таблице 1 приведен средний результат. Тесты, содержащие 25 последовательных запросов, призваны оценить влияние балансировки нагрузки на работу алгоритма. Для проведения тестирования виртуальные машины типа «Front» были взяты аналогичного с Kubernetes типа и расположены в зонах Europe-West3 (Германия) и Asia-East1 (Тайвань). Хосты типа «Back» были представлены виртуальными машинами на личных ПК разработчика в разных точках г. Тюмени с разными провайдерами.

Таблица №1

Результаты тестирования

Измерение	Google Kubernetes	Экс. Кластер (Europe-West3)	Экс. Кластер (Asia-East1)
1	2	3	4
GET, среднее на 1 запрос (сек)	0.18705	0.35105	0.65478
POST, среднее на 1 запрос (сек)	0.21208	0.40122	0.65627

1	2	3	4
GET, среднее на 10 запросов (сек)	0.94552	1.75472	3.26728
POST, среднее на 1 запрос (сек)	1.06918	1.95673	3.27777
GET, среднее на 25 запросов (сек)	4.67620	8.77647	16.36959
POST, среднее на 25 запросов (сек)	5.30200	10.03058	16.40684

Таким образом, в ходе тестового развертывания приложения средствами разрабатываемого прототипа кластера облачных вычислений на принципах экономики совместного потребления удалось добиться работоспособности приложения и базовых функций балансировщика: хост типа «Front» доставлял пользователям приложение, последовательно распределяя запросы между несколькими хостами типа «Back». Данная схема принципиально соответствует традиционной архитектуре доставки приложений с применением балансировщика и хостов-исполнителей.

Заключение

Развернутое веб-приложение предоставило пользовательский сервис, сопоставимый по качественным и количественным характеристикам с веб-приложением, развернутым на платформе облачного провайдера. Подтверждена принципиальная работоспособность схемы.

Все заявленные этапы работы выполнены, создан задел для дальнейшего развития проекта в сторону совершенствования и соответствия промышленным стандартам ЦОД общественного потребления.

Большую долю технических решений удалось сформировать на основе протестированных открытых или свободных решений, что существенно повышает надежность системы и позволяет концентрироваться на вопросах управления ресурсами и алгоритмах формирования инсталляций приложений.

Проведенный эксперимент не учитывал множество вопросов информационной безопасности, финансового обоснования и хранения данных, что делает возможным дальнейшее совершенствование проекта:

1) проработка вопросов надежности представленной акторами общественного потребления инфраструктуры, вопросы борьбы со злонамеренными акторами, вопросы информационной безопасности хостов в условиях распространения вирусов, злонамеренного ПО и прочих угроз информационной безопасности;

2) модернизация алгоритмов агрегации и распределения приложений и формирования инсталляций в сторону интеграции с существующими решениями типа Kubernetes, что позволит повысить совместимость с существующими облачными решениями и упростить миграцию приложений в обоих направлениях;

3) оценка возможности применения указанной модели для решения задач актуальных трендов туманных и граничных вычислений, исследовать потенциал для интеграции с решениями Интернета Вещей и распределенной обработки больших данных;

4) применения модели как способ наращивания ресурсов существующей информационной системы, когда все ресурсы в рамках реализуемого ЦОД представлены одним актором и используются для массирования вычислительной мощности для отдельных проектов и мероприятий, по факту реализуя часть “облачных” возможностей на on-premises инфраструктуре предприятия;

5) модернизация алгоритма с целью реализации регионального распределения хостов типа «Front» и «Back» для получения небольших суб-инсталляций, ответственных за обслуживание своих региональных клиентов, что позволяет сократить сроки обслуживания клиентских запросов и предоставить.

Исследование выполнено при финансовой поддержке РФФИ в рамках научного проекта № 20-47-720006 «Исследование подходов и реализация технологий сбора, передачи, обработки и хранения данных для масштабируемой вычислительной облачной платформы на принципах «Экономики совместного потребления».

Литература

1. Демиденко А. А., Демиденко А. И. Облачные технологии как залог эффективности современного бизнеса // Современные проблемы и тенденции развития экономики и управления. – 2019. – С. 93-96.
2. Puschmann T., Alt R. Sharing economy // Business & Information Systems Engineering. – 2016. – Т. 58. – №. 1. – pp. 93-99.
3. Etro F. The economics of cloud computing // Cloud Technology: Concepts, Methodologies, Tools, and Applications. – IGI Global, 2015. – pp. 2135-2148.
4. Hawlitschek F., Teubner T., Weinhardt C. Trust in the sharing economy // Die Unternehmung. – 2016. – Т. 70. – №. 1. – pp. 26-44.
5. Shukur H. et al. Cloud computing virtualization of resources allocation for distributed systems // Journal of Applied Science and Technology Trends. – 2020. – Т. 1. – №. 3. – pp. 98-105.

6. Mohamed A. et al. Software-defined networks for resource allocation in cloud computing: A survey //Computer Networks. – 2021. – Т. 195. – pp. 108-151.

7. Macedo R. et al. A survey and classification of software-defined storage systems //ACM Computing Surveys (CSUR). – 2020. – Т. 53. – №. 3. – pp. 1-38.

8. Mao Y. et al. Resource management schemes for cloud-native platforms with computing containers of docker and kubernetes //arXiv preprint arXiv: 2010.10350. – 2020.

9. Raj S. et al. Virtual machine migration in heterogeneous clouds-a practical approach //2020 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT). – IEEE, 2020. – pp. 1-6.

10. Ashley W. D. Foundations of Libvirt Development. – Apress, 2019 – 416 p.

References

1. Demidenko A. A., Demidenko A. I. Sovremennye problemy i tendentsii razvitiya ekonomiki i upravleniya. 2019. pp. 93-96.

2. Puschmann T., Alt R. Business & Information Systems Engineering. 2016. T. 58. №. 1. pp. 93-99.

3. Etro F. Cloud Technology: Concepts, Methodologies, Tools, and Applications. 2015. pp. 2135-2148.

4. Hawlitschek F., Teubner T., Weinhardt C. Die Unternehmung. 2016. T. 70. №. 1. Pp. 26-44.

5. Shukur H. et al. Journal of Applied Science and Technology Trends. 2020. T. 1. №. 3. pp. 98-105.

6. Mohamed A. et al. Computer Networks. 2021. T. 195. pp. 108-151.

7. Macedo R. et al. ACM Computing Surveys (CSUR). 2020. Т. 53. №. 3. pp. 1-38.

8. Mao Y. et al. arXiv preprint arXiv: 2010.10350. 2020.

9. Raj S. et al. 2020 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT). IEEE, 2020. pp. 1-6.

10. Ashley W. D. Foundations of Libvirt Development. Apress, 2019. 416 p.