DOI 10.37882/2223–2966.2021.10.24

ПРИМЕНЕНИЕ АЛГОРИТМОВ БАЛАНСИРОВКИ СЕТЕВОЙ НАГРУЗКИ В ВИРТУАЛЬНОЙ ИНФРАСТРУКТУРЕ ПРОГРАММНО-ОРИЕНТИРОВАННОГО ВЫЧИСЛИТЕЛЬНОГО КЛАСТЕРА

APPLYING NETWORK LOAD BALANCING ALGORITHMS IN THE VIRTUAL INFRASTRUCTURE OF A SOFTWARE-DEFINED COMPUTE CLUSTER

A. Sagalaeva
Yu. Sagalaev
O. Romashkova

Summary. The article is devoted to the study of the possibility of balancing traffic between different parts of a virtualized application running in a private computing cluster. The possibilities of software-configurable networks and their elements in reducing network delays and evenly distributing the load between the system nodes are considered. Received performance results using different traffic distribution algorithms depending on the selected load. The relevance of building services using balancing is justified. Provided an example of using the Openstack Octavia project to recreate the required working environment.

Keywords: cloud computing, load balancing, IT infrastructure, openstack, jmeter.

Сагалаева Анна Игоревна
Аспирант, ГАОУ ВО «Московский городской педагогический университет (МГПУ)» г. Москва
omegaanya@gmail.com
Сагалаев Юрий Романович
Аспирант, ГАОУ ВО «Московский городской педагогический университет (МГПУ)» г. Москва
yrok472@mail.ru
Ромашкова Оксана Николаевна
Д.т.н., профессор, Российская академия народного хозяйства и государственной службы при Президенте РФ (РАНХиГС)» г. Москва
ox-rom@yandex.ru

Аннотация. Статья посвящена исследованию возможности балансировки трафика между различными частями виртуализированного приложения, выполняющегося в частном вычислительном кластере. Рассмотрены возможности программно-конфигурируемых сетей и их элементов в снижении сетевых задержек и равномерному распределению нагрузки между узлами системы. Получены результаты производительности с использованием различных алгоритмов распределения трафика в зависимости от выбранной нагрузки. Обоснована актуальность построения сервисов с использованием балансировки. Приведен пример использования проекта Openstack Octavia для воссоздания требуемого рабочего окружения.

Ключевые слова: облачные вычисления, балансировка нагрузки, ИТ-инфраструктура, openstack, jmeter.

Введение

Существующие подходы в разработке облачных сервисов и приложений для нужд бизнеса подразумевают короткий цикл разработки и тестирования, минимальное время простоя и развертывания решений для конечного пользователя. Такие запросы ставят серьезные требования к среде выполнения разработанных решений. В качестве пространства для размещения приложений с микросервисной архитектурой все чаще используются технологии гипервизорной и контейнерной виртуализации, например на базе платформ KVM/OpenVZ/Docker [1]. Модули разрабатываемой системы запаковываются в самостоятельные контейнеры или образа виртуальных машин и запускаются при помощи системы оркестрации, расположенной на вычислительном кластере. Каждый запущенный модуль, например СУБД, требует наличия уникального ip адреса и других атрибутов для взаимодействия с другими сервисами. Для обеспечения сетевой доступности всех модулей, а также возможности масштабирования, применяется подход с использованием программно-конфигурируемых сетей (Software Defined Networking, SDN).

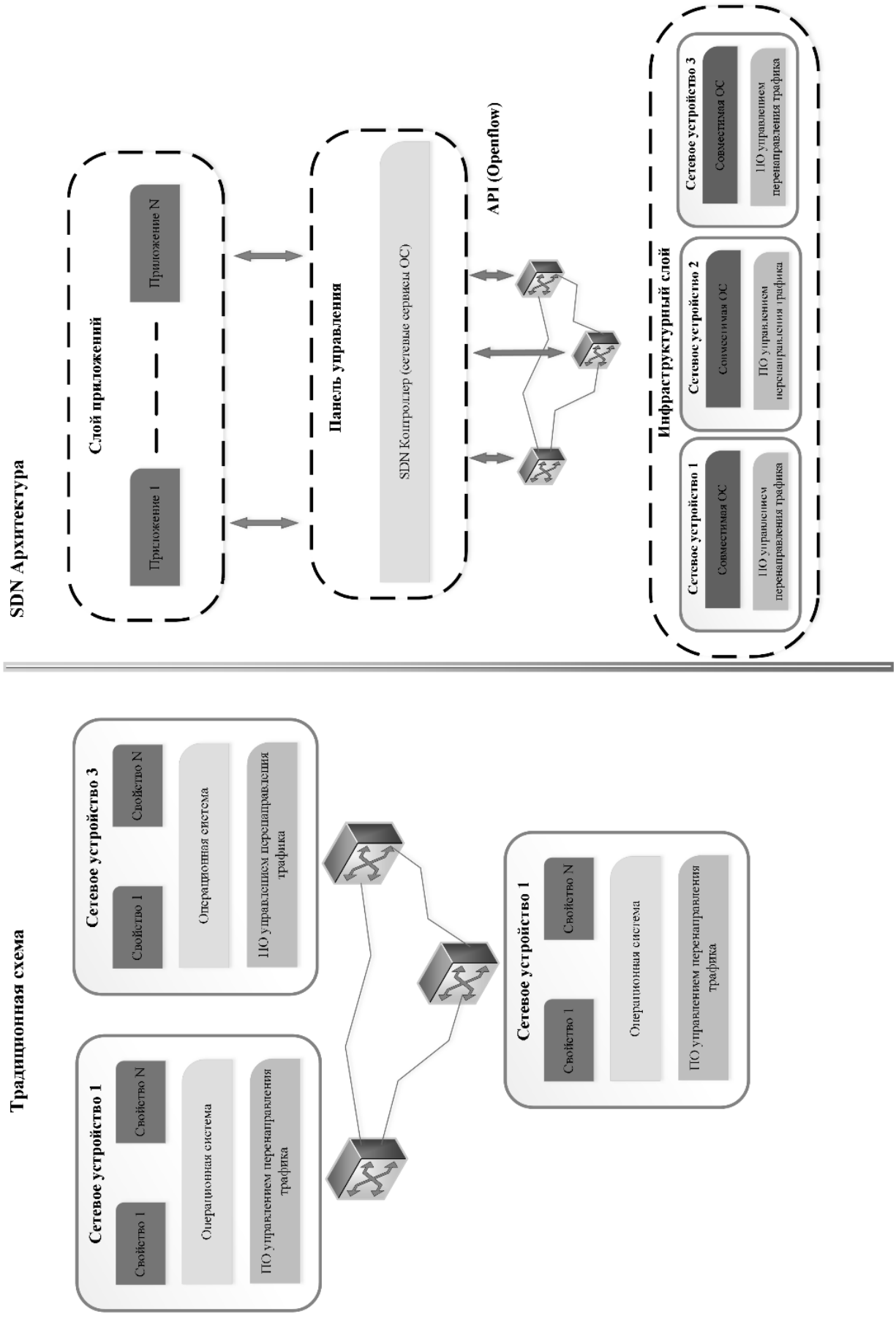

Рис. 1. Сравнение традиционного и SDN подхода при построении сетевой инфраструктуры

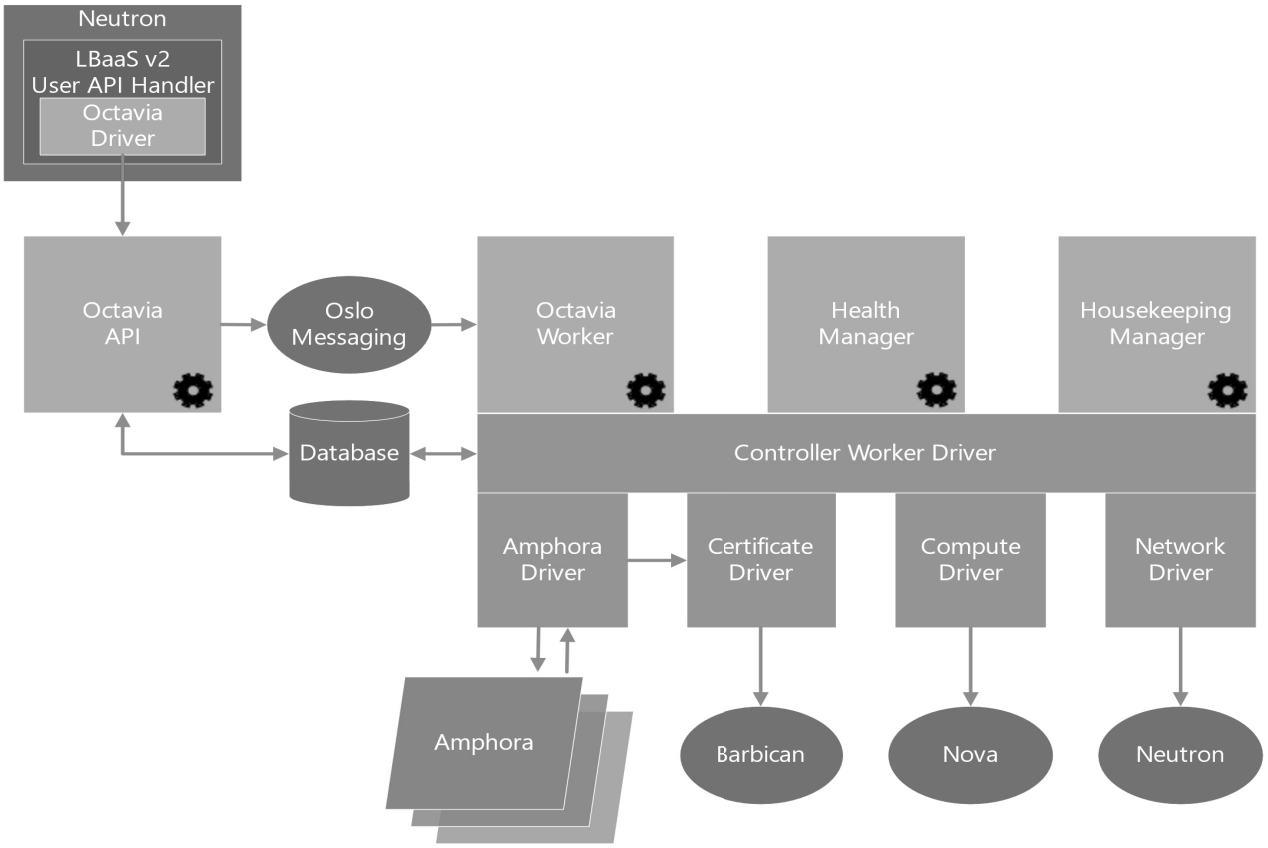

Рис. 2. Взаимодействие Octavia с другими компонентами openstack в кластере

SDN позволяет задавать и перепрограммировать частные сети не прибегая к переконфигурации топологии физических сетей, обслуживающих вычислительный кластер (рисунок 1). Он определяет сетевую архитектуру, в которой состояние продвижения трафика на уровне данных находится под контролем полностью независимого уровня управления [2]. Один и тот же физический канал может обслуживать множество виртуальных сетей, при этом соблюдается полноценная изоляция трафика при помощи использования VXLAN. Технология VXLAN предназначена для формирования больших изолированных сетей L2 в виртуализированных средах с множеством групп пользователей [3] Виртуализация сетевых функций, таких как фильтрация, балансировка трафика, брандмауэра дает простоту масштабирования и отказоустойчивости дает значительное преимущество относительно традиционного подхода в построении сетей.

Алгоритмы балансировки трафика в SDN

Балансировка трафика, согласно сетевой модели OSI, в SDN происходит на прикладном уровне. Существующие применяемые алгоритмы направлены на решении различных задач, такие как сокращение времени отклика, равномерное распределение нагрузки, повышение предсказуемости и достижения масштабируемости системы. В зависимости от выбранного алгоритма, балансировщик работает в режиме «прокси» (proxy), в котором происходит анализ входящих пакетов и дальнейшее перераспределение на конечный ресурс.

Алгоритм кругового обслуживания (Round-Robin) является одним из самых простых и популярных способов распределения запросов, представляющий из себя передачу пакетов между всеми участниками по кругу [4] Он не требует дополнительной настройки со стороны протоколов вышестоящих уровней, а также работает в полном отсутствии сетевой связанности между участниками балансировщика.

В качестве продвинутой альтернативы существует алгоритм, учитывающий текущую загруженность участников пула и распределяющий новые поступающие запросы на менее загруженные узлы (Least connection). Предполагается, что каждое новое установленное соединение создает примерно одинаковую загрузку, что

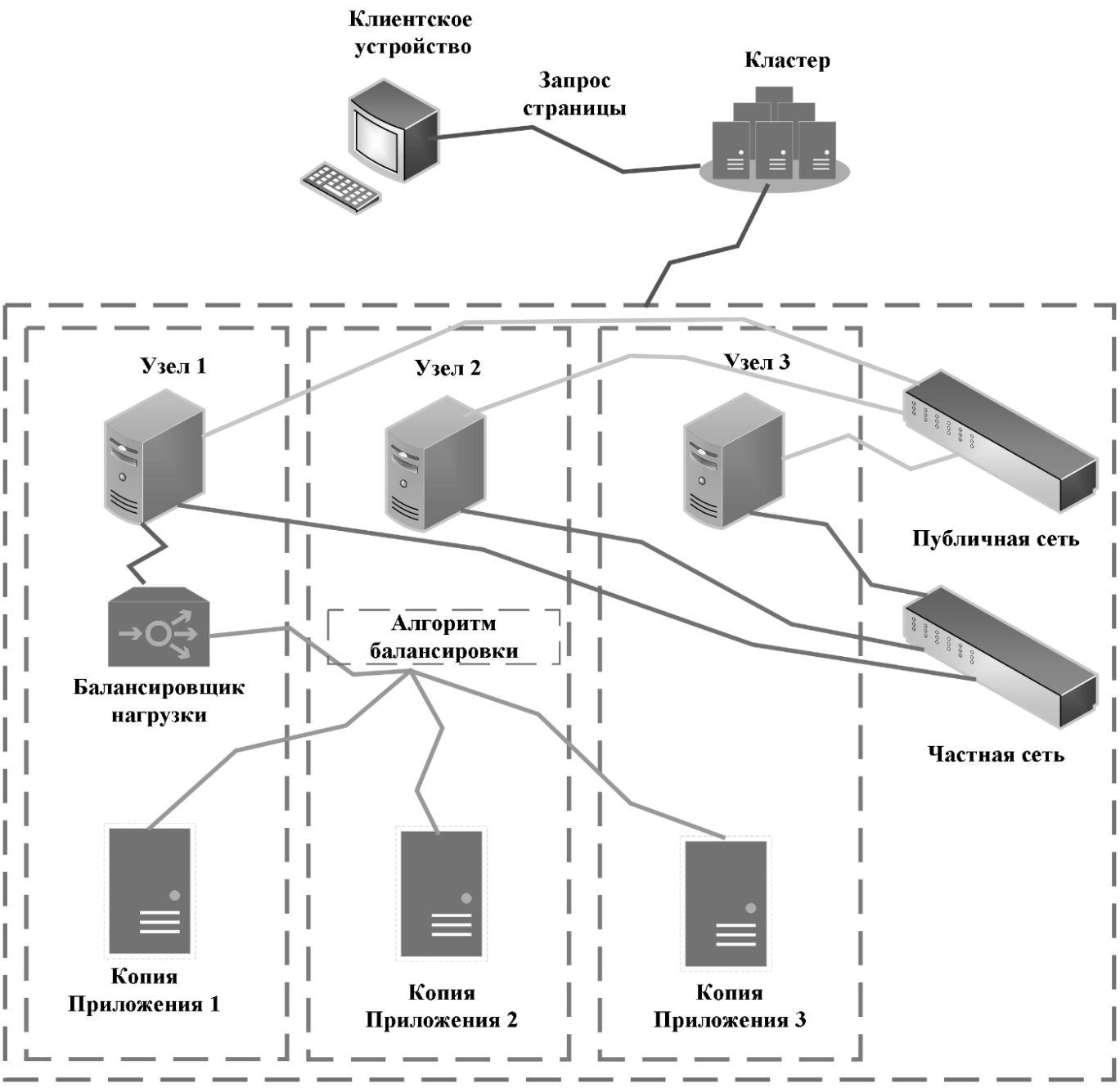

Рис. 3. Прохождение трафика до конечного экземпляра приложения через балансировку

позволяет прогнозировать занятость ресурсов при увеличении количества запросов.

Еще одним из способов уменьшения времени выполнения запросов является алгоритм, привязывающий определенного клиента к конкретному экземпляру приложения в случае многократного обращения (Source IP). Технология кеширования и выбор одного и того же пути до приложения позволяют пользователю не тратить время на «холодный старт».

Использование решения Openstack Octavia при балансировке нагрузки между виртуальными машинами вычислительного кластера

Openstack Octavia является программным решением с открытым исходным кодом, разработанное для работы с Openstack [5]. Octavia выполняет задачу по балансировке нагрузки, управляя массивом виртуальных машин, контейнеров или физических узлов (так называемыми «амфорами»), которые запускаются по мере

Таблица 1. Описание характеристик нагрузочного тестирования

Параметр	Значение
Количество пользователей	500
Время выхода на максимальную мощность	300 секунд
Время проведения теста	1 час

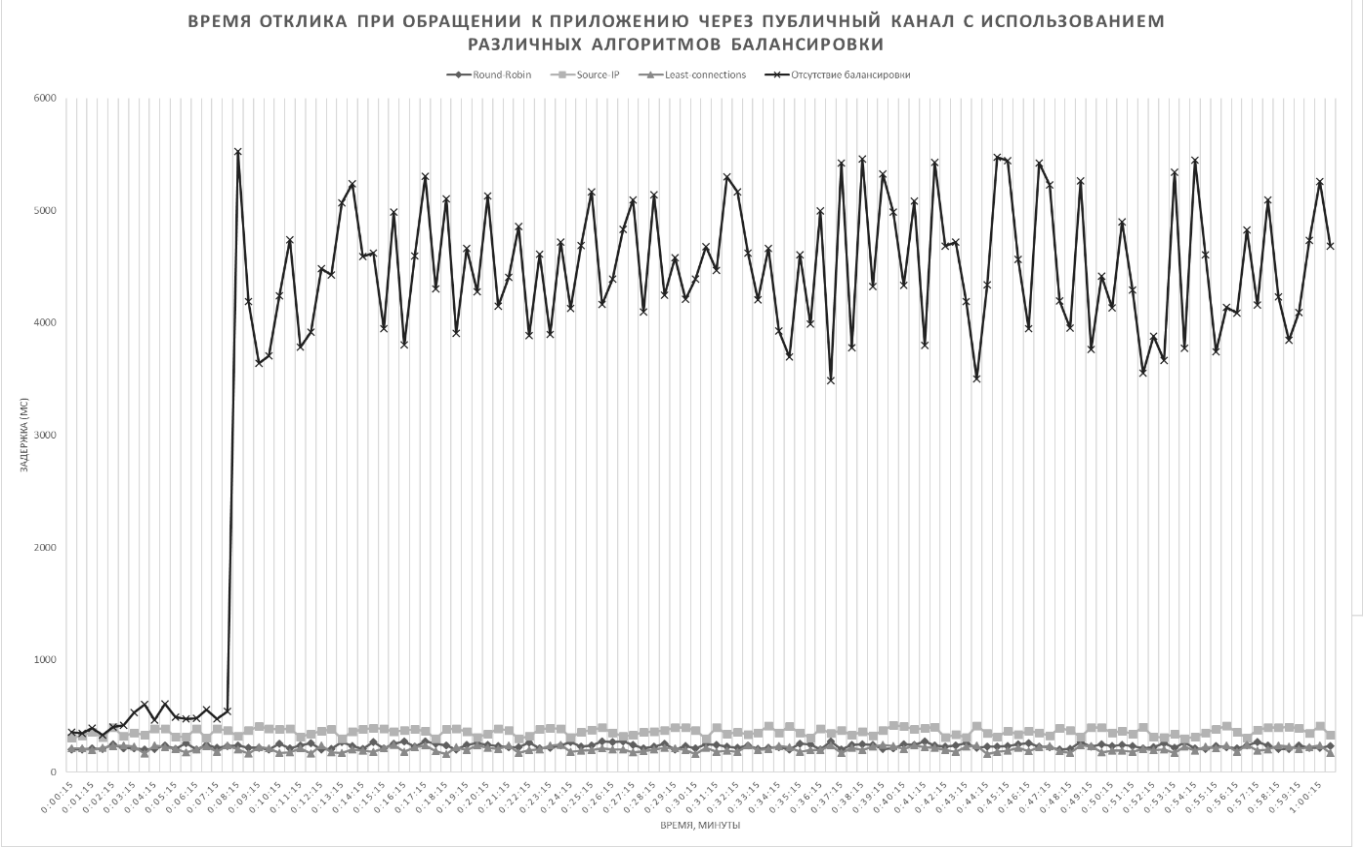

Рис. 4. Время отклика через публичный канал с использованием различных алгоритмов балансировки

необходимости. Функция горизонтального масштабирования по требованию является одной из ключевых среди остальных доступных решений, тем самым делая технологию наиболее подходящей для использования в частном или публичном вычислительном кластере. Octavia тесно взаимодействует с другими системами openstack (рисунок 2).

Для демонстрации работоспособности балансировки при помощи Octavia, а также для получения сравнительного анализа производительности была произведена полноценная установка openstack проекта на физическом кластере, состоящим из пяти узлов под управлением ОС Centos 7, объединенных отдельной физической сетью со скоростью передачи 25 Гбит/с для транспортировки трафика частных сетей. Каждый узел имеет физический канал для выхода в сеть, который также реализует публичный доступ к приложению посредством второго интерфейса. Для измерений были выбраны узлы со следующими характеристиками:

1. Процессор Intel CPU E3–1230 3,50 ГГЦ;
2. SSD накопитель Samsung PM983 (линейная чтения до 3200 Мб/с);
3. Оперативная память MTA18ADF2G72AZ-2G6E1DDR4 2666 МГц.

В качестве средства управления приложением был развернут оркестратор kubernetes, состоящий из виртуальных машин, располагающихся на физическом кластере, пять из которых служат местом исполнения контейнеров с полезной нагрузкой. Kubernetes — открытое программное обеспечение для автоматизации

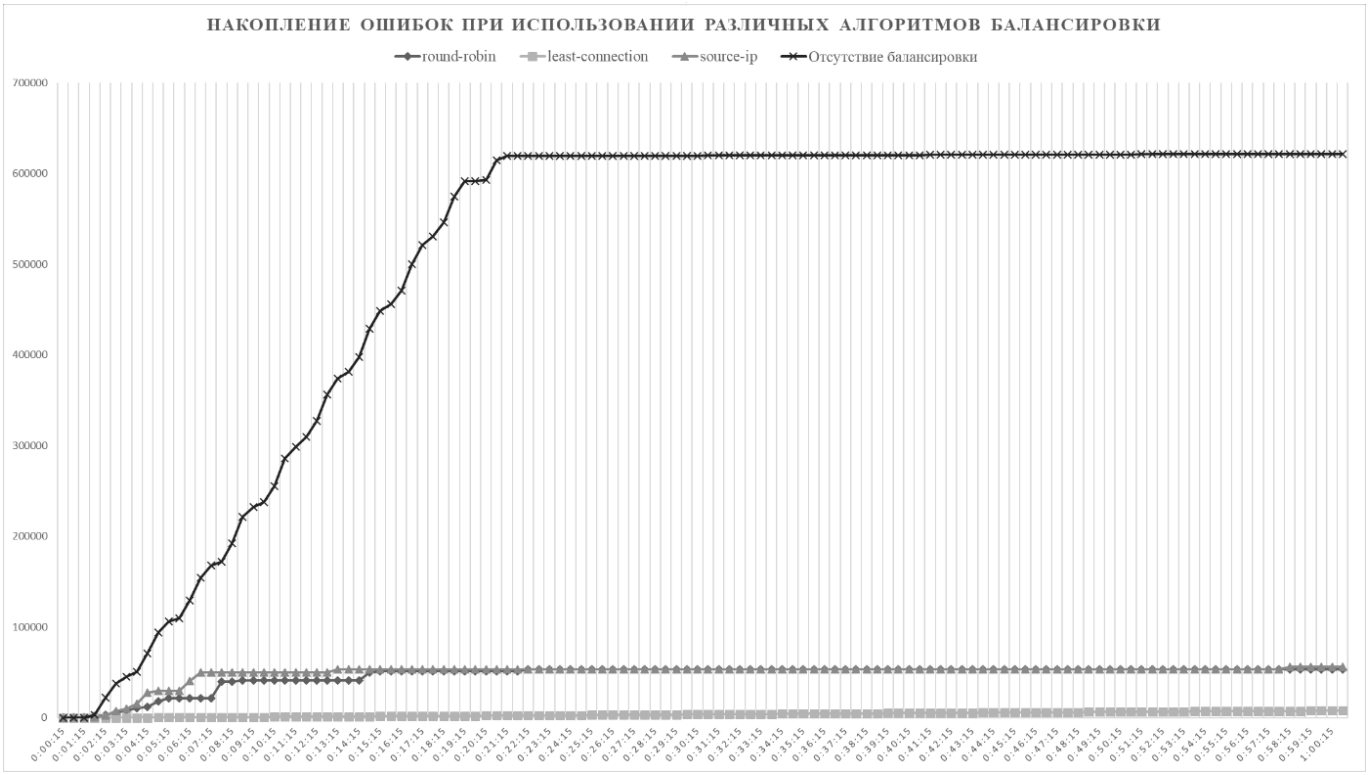

Рис. 5. Тенденция накопления ошибок с использованием различных алгоритмов балансировки

развёртывания, масштабирования и управления контейнеризированными приложениями [6]. Каждая рабочая виртуальная машина-«работник» (worker) исполняется на отдельном физическом узле.

Приложением выступает веб-страница, берущая содержимое для наполнения из базы данных. Приложение состоит из следующих компонентов:

1. Сервис, отвечающий за внешний пользовательский интерфейс (frontend). Отвечает за вывод веб станицы при обращении к нему. За наполнением таблицы с информацией обращается к сервису базы данных (backend).
2. Сервис, отвечающий за выполнение базы данных MongoDB. Сохраняет поступающую информацию от сервиса frontend, а также отдает информацию при запросе. Использование MongoDB обуславливается возможностью хранить бесструктурные данные и достигать высокую скорость доступа к ним [7].

Worker-узлы ограничены в ресурсах, кластер выделяет по 4 Гбайт оперативной памяти и 4 виртуальных ЦПУ для каждого такого узла. Сервис обеспечивающий вывод интерфейса развернут в количестве пяти экземпляров, один на каждый worker-узел, что означает наличие такого сервиса на каждом узле вычислительного кластера.

Для организации внешнего доступа к созданным ресурсам выстраивается цепочка из виртуального маршрутизатора, объединяющий собой публичную и частную сеть, балансировщик нагрузки, использующий полученный маршрутизатор для подключения (рисунок 3). Балансировщику присваивается ip адрес из публичной сети (Floating ip), а также сообщается «пул балансировки» — свойство, содержащее в себе знания, о том, при помощи какого алгоритма и между какими виртуальными машинами производить балансировку трафика. Для сравнительного анализа производительности были выбраны доступные алгоритмы балансировки — циклический (Round Robin), least connections и source IP, а также непосредственно прямой доступ к одному из узлов с приложения без использования какой-либо балансировкой [8].

В качестве инструмента для измерения скорости была использована кроссплатформенная программа для нагрузочного тестирования Apache Jmeter. Измеряемыми параметрами в ходе проведения эксперимента являются время отклика, количество обработанных запросов за минуту работы, количество отказов. В ходе работы Jmeter создает необходимое количество подключений, тем самым повторяя реальную распределенную работу пользователей. Он может использоваться для моделирования большой нагрузки на сервере, группе серверов, сети, чтобы протести-

ровать их максимальную нагрузочную способность или разложить общую производительность под различными типами загрузки [9]. Входные параметры для прохождения тестирования представлены в таблице 1.

В результате измерения производительности были получены замеры, представленные на рисунке 4.

Анализ данных эксперимента показывает, что правильная маршрутизация сетевого трафика с использованием балансировщика в SDN более эффективна для подхода без использования балансировки, в независимости от выбранного алгоритма. Использование прямого подключения только к одному экземпляру приложения при резком увеличении количества запросов утилизирует все мощности конкретной виртуальной машины, что приводит к резкому увеличению времени отклика и увеличенное количество получаемых ошибок при обращении (рисунок 5).

Применение конкретных алгоритмов дает преимущество относительно друг друга при дальнейшем увеличении нагрузки на веб-приложение в определенны сценариях. Так, использование «Source IP» алгоритма не целесообразно при построении подобных решений с высокой нагрузкой, так как не дает гарантии доступности ресурса за определенное время, в случае высокой нагрузки определенной ВМ. Использование «Round-Robin» позволяет равномерно распределить все запросы между всеми участниками пула, используя круговой перебор, однако не решает проблему повышения времени отклика и количества отказов в случае проблем с конкретным экземпляром приложения, например в случае утечки памяти или повышенного потребления ЦПУ. «Least Connections» производит проверку количества активных соединений с каждой из ВМ и устанавливает новое соединение к ВМ с минимальным количеством текущих соединений, таким образом добиваясь более эффективной утилизации ресурсов кластера и не допуская переадресацию на перегруженный экземпляр [10].

Заключение

В результате проведенного исследования можно сделать выводы о целесообразности использования балансировки сетевой нагрузки приложения на вычислительном кластере. Рассмотрен подход программно-конфигурируемых сетей. Существующие алгоритмы, в том числе использующиеся в проекте Openstack Octavia, позволяют оптимизировать распределение нагрузки и тем самым улучшить время отклика и загрузку кластера в целом, в контексте конкретного приложения. В ходе экспериментальной части исследования были получены результаты, указывающие о повышении скорости обработки запросов с использованием распределения трафика относительно традиционного прямого обращения к тестируемому сервису.

ЛИТЕРАТУРА

1. Колясников П.В., Силаков И.Н., Ильин Д.Ю., Гусев А.А., Никульчев Е.В. Повышение эффективности виртуального рабочего окружения распределенной разработки программ // Современные информационные технологии и ИТ-образование. 2019. № 1. URL: https://cyberleninka.ru/article/n/povyshenie-effektivnosti-virtualnogo-rabochego-okruzheniya-raspredelennoy-razrabotki-programm (дата обращения: 14.05.2021).
2. Логинов С.С. Об уровнях управления в программно-конфигурируемой сети (sdn) // T-Comm. 2017. № 3. URL: https://cyberleninka.ru/article/n/ob-urovnyah-upravleniya-v-programmno-konfiguriruemoy-seti-sdn (дата обращения: 14.05.2021).
3. Ромасевич П.В., Смирнова Е.В. Решения D-Link для организации учебного процесса в области инфокоммуникаций и телекоммуникационной инфраструктуры центров обработки данных // Современные информационные технологии и ИТ-образование. 2018. № 4. URL: https://cyberleninka.ru/article/n/resheniya-d-link-dlya-organizatsii-uchebnogo-protsessa-v-oblasti-infokommunikatsiy-i-telekommunikatsionnoy-infrastruktury-tsentrov (дата обращения: 14.05.2021).
4. Ломов Э.О., Бурковский В.Л. Проблематика управления информационным обслуживанием населения в условиях гетерогенности информационных потоков // Вестник ВГТУ. 2011. № 8. URL: https://cyberleninka.ru/article/n/problematika-upravleniya-informatsionnym-obsluzhivaniem-naseleniya-v-usloviyah-geterogennosti-informatsionnyh-potokov (дата обращения: 14.05.2021).
5. Feoktistov A.G., Sidorov I.A., Sergeev V.V., Kostromin R.O., Bogdanova V.G. Virtualization of heterogeneous HPC-clusters based on OpenStack platform // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2017. № 2. URL: https://cyberleninka.ru/article/n/virtualization-of-heterogeneous-hpc-clusters-based-on-openstack-platform (дата обращения: 14.05.2021).
6. Курганова Н.В., Филин М.А., Черняев Д.С., Шаклеин А.Г., Намиот Д.Е. Внедрение цифровых двойников как одно из ключевых направлений цифровизации производства // International Journal of Open Information Technologies. 2019. № 5. URL: https://cyberleninka.ru/article/n/vnedrenie-tsifrovyh-dvoynikov-kak-odno-iz-klyuchevyh-napravleniy-tsifrovizatsii-proizvodstva (дата обращения: 14.05.2021).
7. Иванов С.И., Тарутина Н.В., Голубчиков М.А., Сафаров Р.Р. Программное обеспечение для учета и хранения клинической и социодемографической информации о больных // Программные продукты и системы. 2015. № 3 (111). URL: https://cyberleninka.ru/article/n/programmnoe-obespechenie-dlya-ucheta-i-hraneniya-klinicheskoy-i-sotsiodemograficheskoy-informatsii-o-bolnyh (дата обращения: 14.05.2021).

8. Использование консольного расширения Octavia для клиента OpenStack. URL: https://docs.openstack.org/python-octaviaclient/latest/cli/index.html#loadbalancer (дата обращения: 13.05.2021).
9. Темичев А.А., Файзрахманов Р.А. Аналитический обзор средств автоматизации тестирования производительности применительно к системам мониторинга // Вестник ПНИПУ. Электротехника, информационные технологии, системы управления. 2015. № 15. URL: https://cyberleninka.ru/article/n/analiticheskiy-obzor-sredstv-avtomatizatsii-testirovaniya-proizvoditelnosti-primenitelno-k-sistemam-monitoringa (дата обращения: 14.05.2021).
10. Алгоритмы и методы балансировки нагрузки. URL: https://kemptechnologies.com/load-balancer/load-balancing-algorithms-techniques/ (дата обращения: 13.05.2021).

© Сагалаева Анна Игоревна (omegaanya@gmail.com),
Сагалаев Юрий Романович (yrok472@mail.ru), Ромашкова Оксана Николаевна (ox-rom@yandex.ru).
Журнал «Современная наука: актуальные проблемы теории и практики»

Московский городской педагогический университет