ПРОТОТИП СИСТЕМЫ УПРАВЛЕНИЯ ОБЛАЧНЫМИ РЕСУРСАМИ ДЛЯ ЦЕНТРОВ ОБРАБОТКИ ДАННЫХ, ПОСТРОЕННЫХ НА БАЗЕ ПРОГРАММНО-КОНФИГУРИРУЕМЫХ СЕТЕЙ

Полежаев П.Н.
Оренбургский государственный университет

В настоящее время облачные решения получили широкое распространение в сфере информационных технологий. С точки зрения провайдера весьма актуальным является предоставление облачных услуг максимально эффективным образом. В рамках данной работы было предложено соединить облачные технологии с программно-конфигурируемыми сетями (Software Defined Networking, SDN) и с технологией виртуализации сетевых функций (Network Function Virtualization, NFV). В результате были разработаны алгоритмы [1, 2], обеспечивающие эффективное планирование виртуальных машин, назначение на них контейнеров облачных приложений, также были созданы алгоритмы проактивной и реактивной маршрутизации потоков данных для программно-конфигурируемых сетей.

Данные алгоритмы были реализованы в виде отдельных модулей прототипа системы управления облачными ресурсами (см. рисунок 1).

Сервер контроллера SDN (Software Defined Networking – программно-конфигурируемой сети) представляет собой контроллер Ryu. Для него были разработаны следующие программные модули:

а) Модуль межсетевого экрана для облачных систем – с помощью OpenFlow реализует блокировку трафика на уровнях L2-L4 с поддержкой контроля состояния для протоколов TCP и UDP. Алгоритм работы данного модуля описан в [3].

б) Модуль сбора информации о состоянии программно-конфигурируемой сети облачной системы – реализует алгоритм сбора информации о топологии и состоянии сети [4], который основывается на сочетании использования протоколов LLDP, SNMP и OpenFlow в качестве источников информации для получения сведений о состоянии сети. Собираемая информация активно применяется другими модулями прототипа.

в) Модуль построения коммуникационных схем сервисов на основе статистики их взаимодействия – реализует алгоритм [1], который анализирует файл Docker Compose облачного приложения с целью выявления его отдельных микросервисов и собирает статистику об их сетевой активности с помощью счетчиков OpenFlow.

г) Модуль реактивной и проактивной маршрутизации потоков данных – реализует генетический алгоритм маршрутизации потоков данных для реактив-

ного случая и алгоритм Дейкстры для проактивного [2]. В первом случае проактивная маршрутизация выполняется до запуска облачного приложения после определения назначенных виртуальных машин. Генетический алгоритм решает оптимизационную задачу по максимизации оценки степени соблюдения требований к QoS – максимальной гарантированной задержки и минимальной гарантированной пропускной способности.

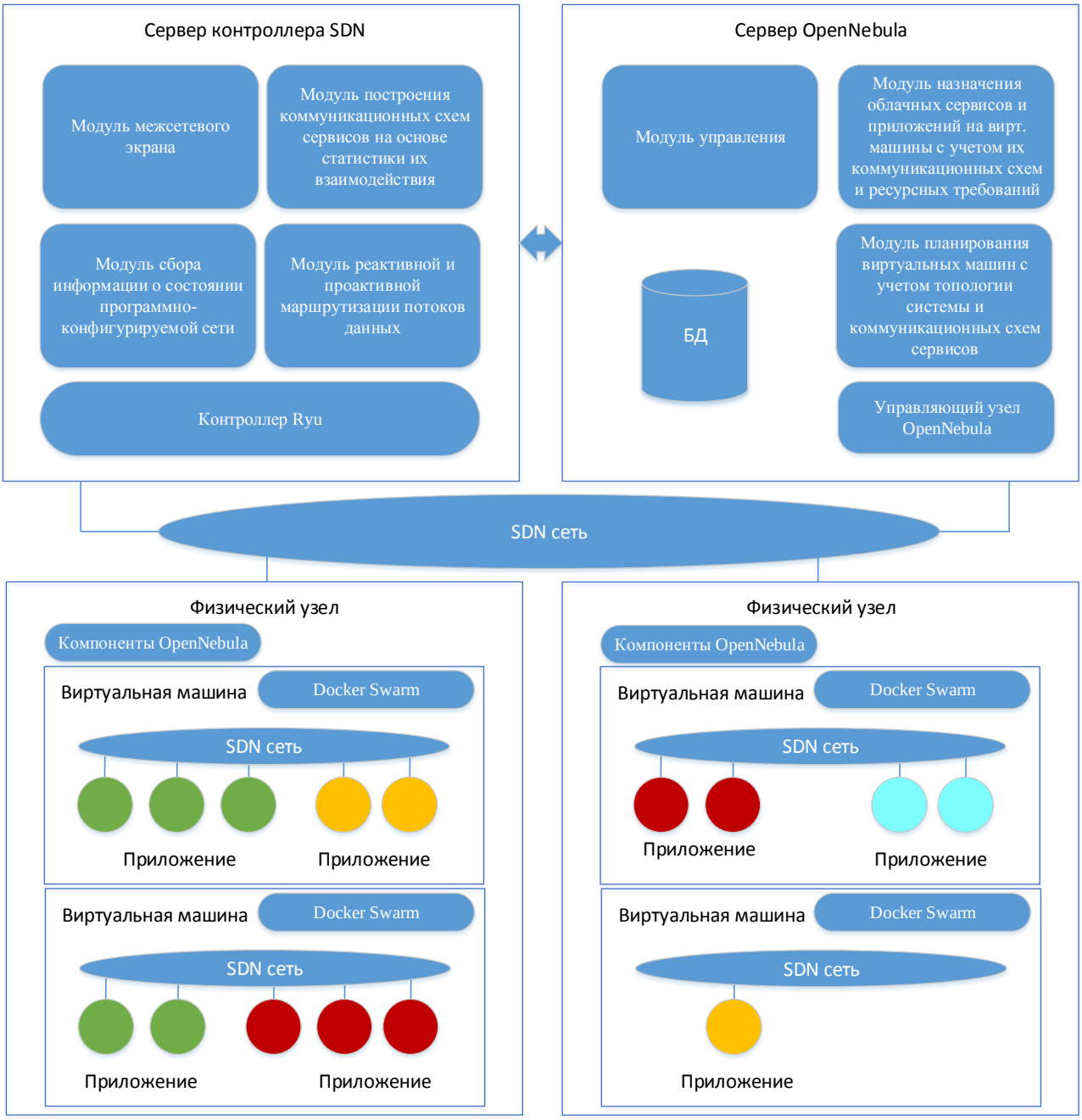

Рисунок 1 – Логическая архитектура прототипа системы управления

Во втором случае реактивная маршрутизация выполняется для динамически возникающего трафика, о котором нет информации заранее. Здесь было предложено использовать модифицированный вариант алгоритма Дейкстры [2].

Сервер OpenNebula представляет собой управляющий узел для системы управления облачной системы OpenNebula, которая была выбрана в качестве основы для реализации данного прототипа. Он включает в себя следующие модули:

а) Модуль управления – предоставляет простой Web-интерфейс для работы конечных пользователей с прототипом. С его помощью можно запустить облачное приложение и управлять им.

б) БД – база данных прототипа, содержащая информацию остальных модулей. Реализована с помощью СУБД PostgreSQL.

в) Модуль назначения облачных сервисов и приложений на виртуальные машины с учетом их коммуникационных схем и ресурсных требований. Он реализует разработанный генетический алгоритм, оптимизирующий целевую функцию – линейную свертку оценок использования вычислительных и коммуникационных ресурсов [2]. Алгоритм оптимизирует всю целевую функцию путем подбора оптимального отображения микросервисов облачного приложения на существующие и новые запускаемые виртуальные машины (за счет свободных ресурсов физических серверов). При этом для получения коммуникационной оценки каждый раз вызывается модуль проактивной маршрутизации потоков данных, который прокладывает все необходимы маршруты с учетом требований к QoS.

г) Модуль планирования виртуальных машин с учетом топологии системы и коммуникационных схем сервисов, запускаемых внутри виртуальных машин. Данный модуль заменяет модуль OpenNebula и решает две задачи: запуск группы виртуальных машин для назначения компонентов запускаемого облачного приложения (с помощью генетического алгоритма) [2], запуск одиночной виртуальной машины для размещения компоненты (микросервиса) облачного приложения (с помощью жадного алгоритма) [1].

На каждом физическом узле, используемом для развертывания облачных приложений и сервисов, были установлены обслуживающие компоненты OpenNebula. Через них происходит запуск виртуальных машин, в которых функционирует Docker Swarm. С помощью Docker Swarm происходит развертывания облачных приложений, состоящих из компонент (микросервисов), описываемых в виде файлов Docker Compose.

На рисунке 1 различные облачные приложения изображены в виде кругов внутри виртуальных машин одного цвета. Облачные приложения могут быть распределены по нескольким виртуальным машинам, которые в общем случае могут выполняться на нескольких узлах.

Связность приложений и инфраструктурных компонент обеспечивается с помощью SDN.

Прототип был развернут на базе оборудования Оренбургского государственного университета, его физическая архитектура изображена на рисунке 2.

Экспериментальный сегмент облачной системы располагается в телекоммуникационной стойке и включает в себя: сетевое хранилище данных NetGear ReadyNAS, два OpenFlow коммутатора HP 3500, два OpenFlow коммутатора NetGear 7224, сервер OpenNebula Aquarius Server T50, сервер контроллера SDN, (Intel Xeon, 4 ядра, 32 Гб оперативной памяти), 8 вычислительных узлов и шлюз для подключения к сети Интернет.

Коммутаторы NetGear 7224 и H33500 соединены в топологию кольца. Достоинством такой структуры является то, что каждый узел физически соединен со всеми остальными, что обеспечивает высокую степень избыточности. Если какой-либо канал выходит из строя, то существует резервный маршрут, позволяющий передать данные в пункт назначения.

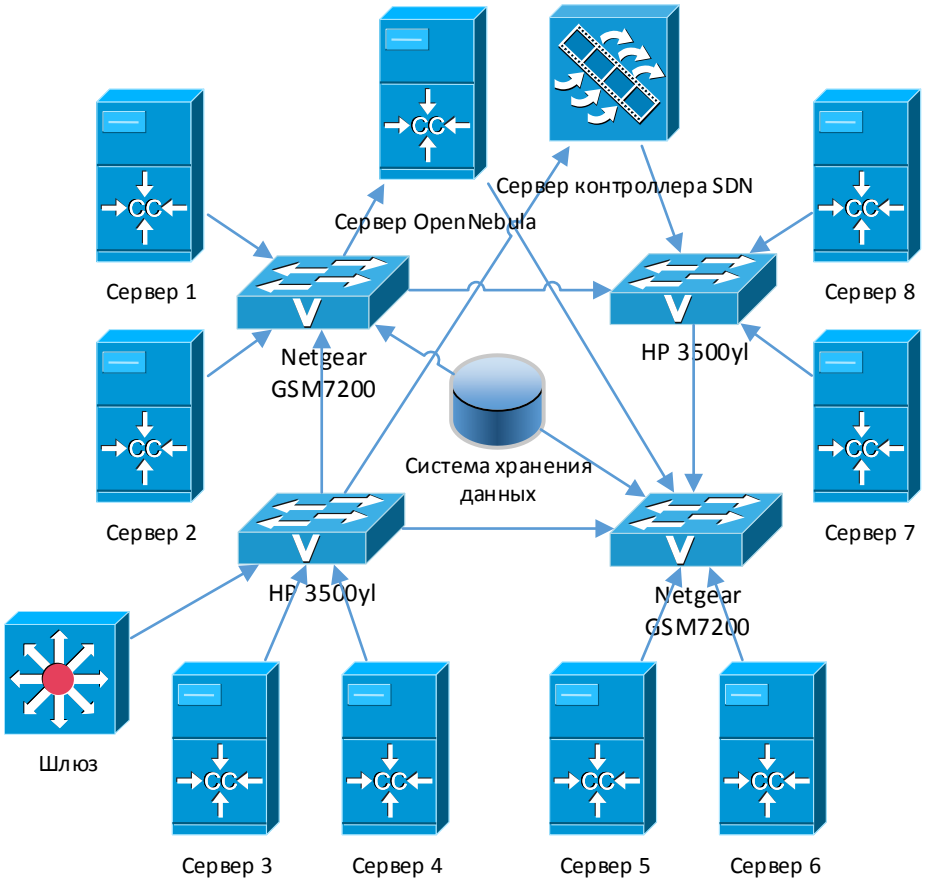

Рисунок 2 – Физическая архитектура прототипа системы управления

Сервера OpenNebula и контроллера OpenFlow подключены к двум разным коммутаторам с целью обеспечения избыточности и надежности. Серверы 1-8 используются для развертывания виртуальных машин и облачных приложений.

В экспериментальном исследовании участвовали облачные приложения следующих типов:

а) Neural Network – нейросетевое приложение в процессе обучения на наборе данных CIFAR-10 [5].

б) QSS – приложение имитационного моделирования.

в) Media Converter – приложение автоматической конвертации мультимедийных файлов.

Все входные данные, необходимые для работы данных приложений, размещались во внешних docker volume, которые располагались в системе хранения данных.

Экспериментальное исследование сервисов производилось в смешанном режиме. Для всех типов облачных приложений генерировался собственный экспоненциальный поток заявок на их запуск с типовыми значениями интенсивности $\lambda_{NN} = 0.01 \cdot L$, $\lambda_{QSS} = 0.1 \cdot L$ и $\lambda_{Media} = 0.02 \cdot L$ для соответствующих серви-

сов. Здесь $\cdot L \in [1,100]$ – коэффициент загрузки, позволяющий исследовать эффект масштабирования. С целью автоматической генерации потока заявок было написано отдельное вспомогательное приложение.

Экспериментальное исследование проводилось для различных значений коэффициента загрузки L с определенным дискретным шагом. Для каждого конкретного значения L проводилось по 20 повторений эксперимента, получаемые значения метрик усреднялись.

Сравнение производилось между разработанным прототипом и обычной облачной системы. В качестве последней использовался прототип, запущенный в минимальном режиме – стандартный планировщик виртуальных машин OpenNebula, стандартный модуль назначения, алгоритм маршрутизации OSPF, SDN не используется.

На рисунке 3 представлены графики зависимости средней загруженности вычислительных ядер серверов от коэффициента загрузки. Прототип демонстрирует увеличение средней загруженности вычислительных ядер на 3-5 % в зависимости от коэффициента загрузки.

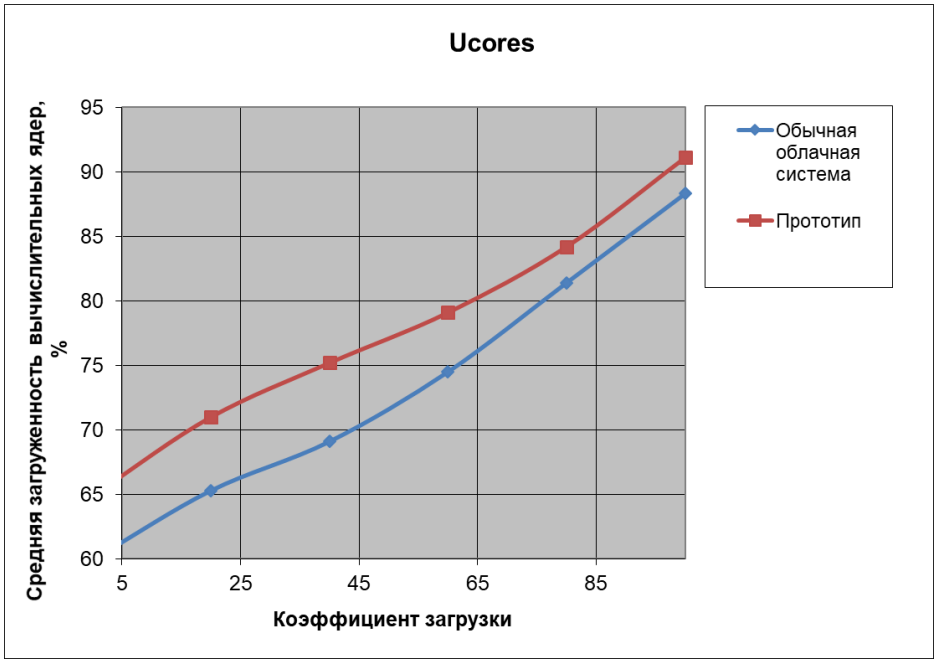

Рисунок 3 – Графики зависимости средней загруженности вычислительных ядер серверов от коэффициента загрузки

Данные значения согласуются с результатами, полученными ранее с помощью симулятора облачной системы и программно-конфигурируемой сети – с увеличением коэффициента загрузки растет значение показателя, также демон-

стрируется улучшение в сравнении со стандартными решениями. Показатели улучшения того же порядка, что и на симуляторе.

На рисунке 2 представлены графики зависимости процента нарушений требований QoS от коэффициента загрузки. Здесь видно, что прототип показывает меньше потерь, что также согласуется с результатами, полученными с помощью симулятора.

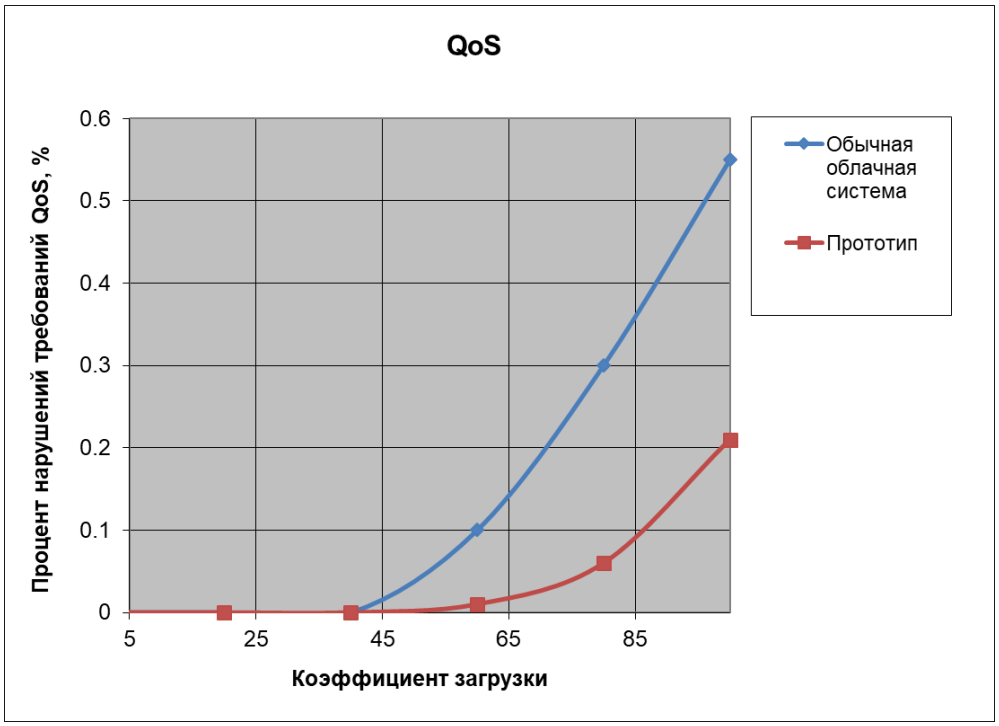

Рисунок 4 – Графики зависимости процента нарушений требований QoS от коэффициента загрузки

Разработан прототип системы управления облачными ресурсами для центров обработки данных, построенных на базе программно-конфигурируемых сетей. Описаны его логическая и физическая архитектуры. Проведенные экспериментальные исследования продемонстрировали его эффективность.

Созданный прототип может быть использован владельцами публичных и приватных облачных ЦОД, а также компаниями-разработчиками облачных ЦОД под ключ. Использование данного прототипа, включающего все разрабатываемые алгоритмические решения, позволит компаниям увеличить эффективность использования физических ресурсов, а также повысить качество предоставляемых услуг. Последний фактор очень важен для конечных пользователей облачных сервисов.

Работа выполнена при поддержке Президента Российской Федерации, стипендия для молодых ученых и аспирантов (СП-2179.2015.5).

Список литературы

1 Полежаев П.Н. Решение задач планирования виртуальных машин и сбора статистики о работе облачных приложений // Университетский комплекс как региональный центр образования, науки и культуры [Электронный ресурс]: материалы Всероссийской научно-методической конференции; Оренбург. гос. ун-т. - Электрон. дан. - Оренбург: ОГУ, 2017. – С. 3178-3184

2 Полежаев П.Н. Создание эффективных алгоритмов функционирования облачных систем // Университетский комплекс как региональный центр образования, науки и культуры [Электронный ресурс]: материалы Всероссийской научно-методической конференции; Оренбург. гос. ун-т. - Электрон. дан. - Оренбург: ОГУ, 2017. – С. 3185-3193

3 Bakhareva N.F., Polezhaev P.N., Ushakov Yu.A., Shukhman A.E. SDN-based firewall implementation for large corporate networks // Proceedings of 2017 IEEE 11th International Conference on Application of Information and Communication Technologies (AICT 2017), 2017. – P. 313-318.

4 Ушаков Ю.А., Полежаев П.Н., Бахарева Н.Ф., Коннов А.Л. Сбор и обобщение информации с сетевых устройств и виртуальных сетевых модулей в рамках сегмента сети NFV // Интеллект. Инновации. Инвестиции, 2017. - №10. – С 82-87.

5 Порохненко Ю.С., Полежаев П.Н. Классификация изображений набора данных CIFAR-10 с помощью нейронных сетей // Компьютерная интеграция производства и ИПИ-технологии: материалы VIII Всероссийской научно-практической конференции. – Оренбург, 2017. – С. 299-303.