УДК 004.93’1-021.131

СИСТЕМА РАСПОЗНАВАНИЯ ОБЪЕКТОВ В ВИДЕОПОТОКЕ НА ОСНОВЕ ТЕХНОЛОГИЙ ВИРТУАЛИЗАЦИИ И ОБЛАЧНЫХ ВЫЧИСЛЕНИЙ

Е.А. Пац
Магистрант БГУИР, Cloud Operations Engineer Netcracker Technology

Е.В. Насуро
Доцент кафедры электронных вычислительных машин БГУИР, кандидат технических наук

Белорусский государственный университет информатики и радиоэлектроники, Республика Беларусь
E-mail: egor.pats@gmail.com

Пац Е.А.
Окончил Белорусский государственный университет информатики и радиоэлектроники. Магистрант БГУИР. Работает в Netcracker Technology в должности Cloud Operations Engineer. Проводит научные исследования в области облачных технологий и DevOps методологий.

Е.В. Насуро
Доцент кафедры электронных вычислительных машин БГУИР, основные направления деятельности: проектирование взаимодействия пользователя и сложных программно-аппаратных продуктов, научное руководство магистрантами.

Аннотация. Облачные технологии – это перспективный и широко применяемый способ обработки и хранения данных. Использование технологий виртуализации позволяет повысить отказоустойчивость предоставляемых сервисов, уменьшить время доставки кода и улучшить утилизацию вычислительных ресурсов. Таким образом возникает потребность использования облачных технологий для решения различных задач. Одной из таких задач является обработка видеопотока. В работе предложен способ построения архитектуры облачной платформы для обработки большого количеств потоков в режиме реального времени. Так же рассмотрены инструменты для конфигурации и развертывания системы.

Ключевые слова: Облачные технологии, виртуализация, IaaS, Terraform, IaC, configuration management systems, Chef, Puppet, Ansible, Salt, PaaS, Kubernetes.

Введение. В современном мире наблюдается очень быстрый рост технологий. Это связано с увеличением потребностей человека и желанием автоматизировать рутинные процессы. Вследствие чего, увеличивается количество сфер человеческой деятельности, где есть возможность применения информационных технологий.

На сегодняшний день одним из перспективных направлений информационных технологий являются облачные вычисления и предоставляемые ими современные сервисы создания, хранения, обработки и поиска данных.

Современные облачные технологии способны существенно сократить расходы, и сегодня многие компании все чаще переносят свои корпоративные системы и бизнес-приложения в облако. Данные системы позволяют в значительной мере снизить нагрузку на инфраструктуру, оптимизировать утилизацию вычислительных ресурсов, повысить доступность критически важных данных и сервисов, улучшить информационную

безопасность компонентов и систем. Перечисленные преимущества делают очевидным выбор облачных технологий большинством компаний.

Одним из наиболее удачных примеров использования облачных технологий является использование микросервисной архитектуры [1] программного обеспечения. Данная архитектура подразумевает использование отдельных сервисов для решения определенных задач. Таким образом убирается единая точка отказа приложения, что увеличивает доступность сервиса даже при отсутствии части функционала.

Зачастую при употреблении термина «видеопоток» подразумевается наличие видеокамеры, которая используется для его записи. Таким образом возникает ряд проблем, которые необходимо решить во время проектирования архитектуры системы. Проблема масштабируемости системы, необходимо предусмотреть возможность многократного увеличения обрабатываемых видеопотоков. Вторая, немаловажная проблема – это проблема изолированности данных. Под данным термином понимается, недоступность данных видеопотока сторонним процессам системы, в том числе и процессам, обрабатывающим соседние потоки данных.

Задача обнаружения и мониторинга объектов в видеопотоке на данный момент, не имеет общепринятых решений. Для анализа видеопоследовательности зачастую используются высококвалифицированные сотрудники, но с развитием технологий появляется возможность оптимизировать данный процесс. Этап обнаружения и распознавания объектов в видеопотоке решается с помощью автономных систем искусственного интеллекта.

При расширении количества обрабатываемых потоков данных необходимо позаботиться о распределении нагрузки, масштабируемости и отказоустойчивости всех ключевых компонентов сервиса, данные задачи позволяет решить разработанная облачная система [2-3]. Зачастую, при развертывании программного обеспечения, отсутствует возможность запуска каждого экземпляра на отдельном, изолированном сервере. Одной из особенностей сложных систем выступает утилизация ресурсов. На данный момент, разработка подобных систем очень актуальна. Большинство компаний, которые предоставляют программное обеспечение как сервис, хотят получить максимальную доступность и минимальное время отклика предоставляемого сервиса. Таким образом, система, разрабатываемая в рамках диссертации, находится на пике популярности, так как отвечает всем основным принципам высоконагруженных, отказоустойчивых, масштабируемых систем.

Стандартные способы предоставления приложений являются дорогостоящими, так же для предоставления доступа к ресурсу в формате 24/7 необходимо экспретное сопровождение серверов, на которых размещено приложение. Данный подход постепенно перестает использоваться большинством компаний из-за его сложности и дороговизны. В качестве решения данной проблемы были предложенны технологии виртуализации, которые позволяют собрать необходимую виртуальную инфраструктуру, которая будет соответствовать бизнес требованиям компаний. Такой подход к созданию и использованию инфраструктуры получил название – облачные технологии.

Так как востребованность облачных технологий значительно выросла за последние годы, большинство крупных компаний разработали методы предоставления виртуальной инфраструктуры как сервиса (IaaS) [4]. Таким образом появилась возможность переноса критически важных инфраструктурных решений в облачную платформу с гарантированной доступностью более 99,9% [5].

Материалы и методы. Большинство современных облачных провайдеров имеют более одного центра обработки данных. Самыми крупными компаниями предоставляющими виртуальную инфраструктуру являются Google (GCP), Amazon (AWS), Microsoft (Azure). Данные компании имеют центры обработки данных, расположенные на разных континентах и в разных странах. Каждый провайдер облачных технологий стремится увеличить время доступности инфраструктуры для конечного потребителя. Таким образом для сравнения лидеров, предоставляющих возможность создания и поддержания виртуальной инфраструктуры, был создан рейтинг, для которого используются две линейные прогрессивные экспертные шкалы: полнота видения (completeness of vision) и способность реализации (ability to execute). На сегодняшний день компания Amazon с облачным решением под названием Amazon Web Services занимает лидирующую позицию по доступности инфраструктурных решений (рисунок 1).

Рисунок *1.* – Оценка качества предоставляемых облачных услуг
на основе доступности инфраструктурных решений

Большинство современных приложений и бизнес ресурсов имеют особые требования к компьютерной безопасности. У многих компаний есть желание перенести свои инфраструктурные решения в облачную платформу, но по причинам безопасности, либо иным бизнес требованиям, нет желания использовать публичные облака и переводить данные и сервисы на сервера облачных провайдеров. Для таких случаев были разработаны облачные платформы для приватного использования. Наиболее популярной приватной облачной платформой является OpenStack.

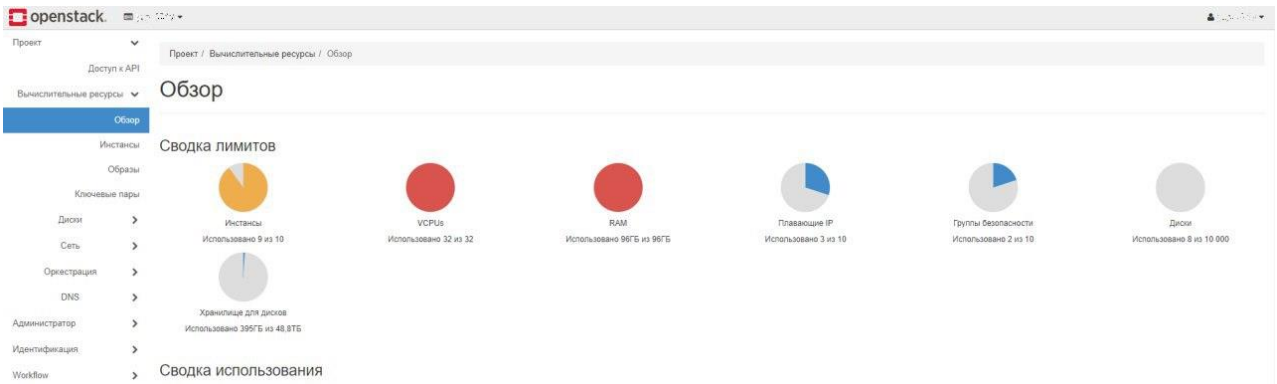

Рисунок *2.* – Веб-интерфейс облачной платформы OpenStack

Облачные платформы позволяют объединять сервера с большими вычислительными мощностями в кластер, что означает, предоставление мощностей нескольких серверов, как единого пула ресурсов. Таким образом, предоставляется возможность, спроектировать архитектуру оптимальным образом, что значительно снижает расходы на аренду или обслуживание железных серверов. Использование данной технологии позволяет использовать один сервер с множеством ролей. Количество ролей зависит от количества виртуальных машин, запущенных на данном сервере.

Так как все вычислительные ресурсы в облаке составляют единый пул, это позволяет гибко масштабировать инфраструктуру. Таким образом, если планируется временная нагрузка на систему, существует возможность добавить к кластеру еще несколько вычислительных узлов для распределения нагрузки. В то же время при выходе одного из серверов из строя, виртуальные машины будут автоматически перенесены на свободные вычислительные узлы. Что позволяет быть уверенными в доступности инфраструктуры.

Большое количество облачных провайдеров создает конкуренцию в предоставлении данной услуги. Каждая из облачных платформ имеет отдельный интерфейс пользователя. Используя данный интерфейс пользователю предоставляется возможность в создании и контролировании инфраструктурных решений в режиме реального времени. Но, зачастую, архитектура инфраструктуры разрабатывается заранее, просчитывается количество необходимых узлов, процессорного времени, оперативной памяти, дискового пространства и так далее. Для конфигурации определенной инфраструктуры, использование пользовательского интерфейса, значительно замедляет время развертывания и настройки основных компонентов. Таким образом большинство провайдеров виртуальной инфраструктуры используют API (Application Program Interface).

Использование API облачных платформ значительно ускоряют создание необходимых компонентов инфраструктуры. Для оптимизации данного процесса был разработан инструмент конфигурации под названием Terraform (рисунок 2). Terraform позиционируется как IaS (Infrastructure as code), позволяя описывать виртуальную инфраструктуру кодом. Так же, используя данный инструмент, появляется возможность поддержания и конфигурации инфраструктуры. Terraform является одним из ключевых инструментов для эффективного развертывания и обновления инфраструктурных решений. Таким образом, Terraform отлично подходит для реализации DevOps методологий.

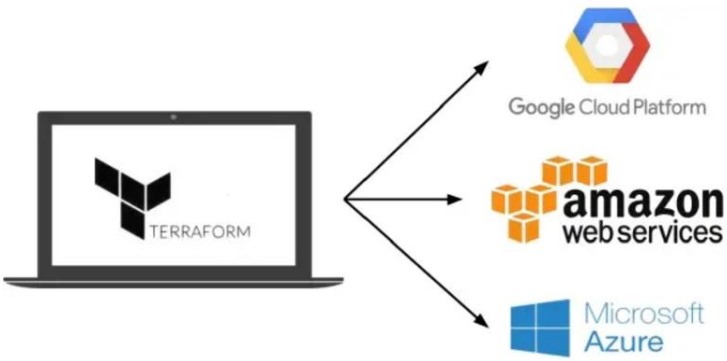

Рисунок *3*. – Возможность использования Terraform, для разворачивания виртуальной инфраструктуры, используя различные облачные провайдеры

При развертывании инфраструктуры, используя Terraform (рисунок 3), появляется возможность сохранения состояния ее компонентов. Таким образом, при изменении одного или нескольких компонентов, инструмент конфигурации применит необходимые настройки только для определенных структурных частей инфраструктуры.

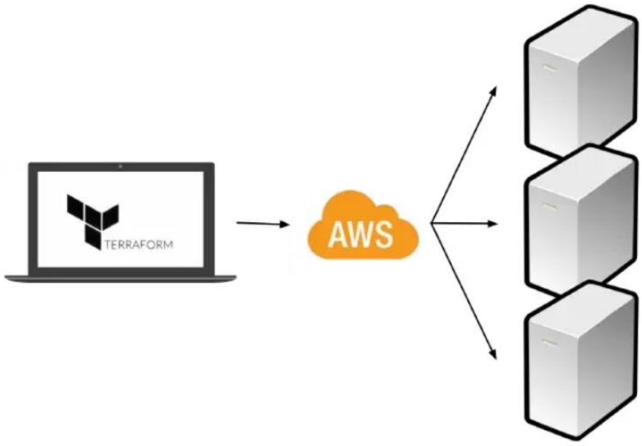

Рисунок *4.* – Получение виртуальной инфраструктуры используя Terraform с API облачного провайдера AWS

В рамках научной работы был разработан Terraform модуль, позволяющий развернуть инфраструктуру для дальнейшей установки системы оркестрации контейнеров под названием Kubernetes.

После разворачивания виртуальной инфраструктуры, появляется необходимость в ее настройке. Одной из основных концепций DevOps является разделение инфраструктуры на «домашних животных» и «крупного рогатого скота». Разница этих подходов в моделях обслуживания. В случаи с «домашними животными» каждый сервер критически важен, и отказ любого из серверов может вывести из строя всю систему. Обычно размер такой инфраструктуры достаточно мал, что позволяет не прибегать к инструментам автоматизации и конфигурации. При использовании данной модели обслуживания все действия выполняются вручную. Совершенно другой подход к обслуживанию инфраструктуры применяется при модели «крупного рогатого скота». В такой модели размер кластера может доходить до сотен вычислительных узлов, и отказ одного или нескольких серверов никак не отразятся на состоянии предоставляемого сервиса. Но настройка таких систем вручную невозможна, таким образом возникает потребность применения систем управления конфигурациями.

Системы управления конфигурациями представляют из себя программное обеспечение, позволяющее получить доступ одновременно к большому количеству вычислительных узлов, с целью применения определенных настроек. Так как с развитием облачных технологий применение данных систем становится все более актуальным, количество данных систем так же увеличивается. Каждая из систем управления конфигурациями разработана для определенных задач и имеет различный порог вхождения и поддержания разработанных решений. К наиболее актуальным можно отнести четыре системы, такие как Chef, Puppet, Ansible, Salt.

Chef – система управления конфигурациями, клиентская часть написана на языке программирования Ruby, серверная часть разработана на Erlang. Система используется для упрощения и автоматизации процессов настройки и конфигурации большого количества серверов (рисунок 4), имеет возможность интеграции с различными облачными решениями. Для использования Chef на вычислительном узле должен быть установлен клиентская часть данной системы с необходимыми правами, что означает, что система не имеет возможности работать в Push-режиме, то есть управлять конфигурациями неподготовленных серверов. Код, написанный для данной системы, называется «рецепт». Рецепты представляют из себя описание состояния вычислительного узла, включая установленные пакеты, запущенные службы, созданные файлы. Chef проверяет каждый из ресурсов инфраструктуры и поддерживает их в эталонном состоянии.

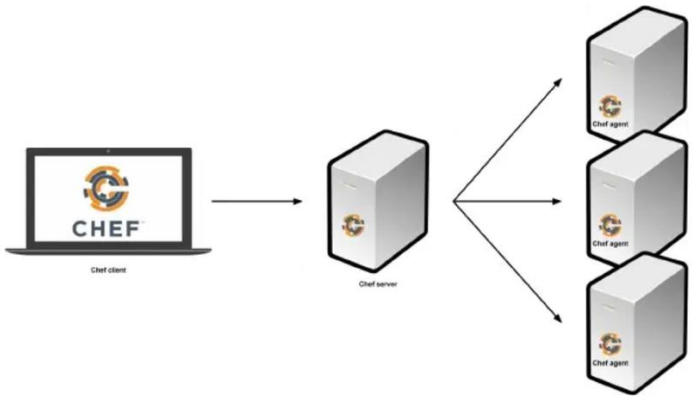

Рисунок 5. – Схема доставки конфигураций системой Chef

Puppet – кроссплатформенная система управления конфигурациями, работающая в режиме клиент-сервер. Система позволяет централизованно управлять настройками операционных систем большого количества вычислительных узлов, включая запущенное на них программное обеспечение. Puppet постоянно опрашивает узлы кластера и поддерживает их в актуальном состоянии. На ровне с Chef, Puppet получил большое распространение и используется большинством компаний на поддержания инфраструктурных решений. Puppet так же не поддерживает Push-режим, таким образом, вычислительные узлы нуждаются в первоначальной подготовке для использования данной системы управления конфигурациями.

Ansible – система управления конфигурациями написанная на языке Python. Данная система имеет достаточно низкий порог вхождения для профессионального использования. Код конфигурации, пишется используя формат yaml, и описывает задачи, которые должны быть выполнены на удаленных серверах. Ansible единственный на данный момент инструмент конфигурации, который не требует клиентского приложения на управляемых вычислительных узлах. Это означает, что данная система работает исключительно в Push-режиме. Подключение к серверам, осуществляется используя SSH. Таким образом требуется чтобы пользователь, от имени которого идет подключения обладал необходимыми правами и состоял в нужных группах.

Для получения списка узлов Ansible должен получить на вход инвентарный файл. В данном файле описывается инфраструктура и способы подключения к каждому из серверов. Так же присутствует возможность выделять группы серверов и определять переменные для определенных групп. Ansible при подключение к серверу, первоначально получает полную информацию о его состоянии, после чего переходит к конфигурации. Если определенное действие уже было выполнено, инструмент конфигурации не будет выполнять его повторно. Все эти достоинства выделяют Ansible из списка самых популярных систем управления конфигурациями.

SaltStack – система управления конфигурациями и удаленного выполнения команд. Данное программное обеспечение, так же, как и Ansible, написано на языке программирования Python. Данная система, на ровне с Chef и Puppet, требует предварительной установки агентов на вычислительные узлы. Salt Master является центральной службой к которой подключаются клиенты – Salt Minion. Модули состояния систем описываются в формате yaml.

Таблица 1. – Сравнение основных параметров систем управления конфигурациями

	Source	Cloud	Type	Infrastructure	Language	Architecture	Community	Maturity
Chef	Open	All	Config Mgmt	Mutable	Procedural	Client/Server	Large	High
Puppet	Open	All	Config Mgmt	Mutable	Declarative	Client/Server	Large	High
Ansible	Open	All	Config Mgmt	Mutable	Procedural	Client-only	Large	Medium
Salt	Open	All	Config Mgmt	Mutable	Declarative	Client/Server	Medium	Medium

Результаты. Для выбора инструмента управления конфигурациями был проведен анализ самых популярных систем. Исходя из таблицы 1 был можно сделать вывод, что оптимальным инструментом, для решения поставленной задачи, является Ansible. Данная система управления конфигурациями имеет открытый исходный код, что означает возможность ее бесплатного использования. Ansible имеет большое сообщество, это означает, что большое количество тривиальных задач уже имеют оттестированные решения. Одним из ключевых преимуществ использования Ansible является отсутствие программ-агентов. Для конфигурации узлов достаточно иметь доступ к ним. Написание кода для конфигурации системы не вызывает затруднений, так как формат yaml интуитивно понятен.

В рамках работы над магистерской диссертацией был разработан Ansible playbook для подготовки всех узлов кластера, и последующего разворачивания системы оркестрации docker контейнеров – Kubernetes.

В работе были проанализированы основные провайдеры облачной инфраструктуры, был проведен их сравнительный анализ и принято архитектурное решение. Особое внимание было уделено типам и принципам построения инфраструктурных решений, были рассмотрены основные методы и способы поддержки кластеров. Разработаны модули автоматического развертывания, согласно концепции DevOps. Был проведен разбор инструментов автоматического управления конфигурациями.

Список литературы

[1]. Ричардсон К. Микросервисы. Паттерны разработки и рефакторинга – Санкт-Петербург: Издательский дом «Питер», 2020. – 544 с.

[2]. Yevgeniy Brikman Terraform: Up & Running: Writing Infrastructure as Code – O'Reilly Media, 2019 – 319 с.

[3]. Lorin Hochstein Ansible: Up and Running – O'Reilly Media, 2015 – 299 с.

[4]. Сайфан Джиджи Осваиваем Kubernetes. Оркестрация контейнерных архитектур. – СПб.: Питер, 2019. – 400 с.

[5]. Gartner Report: Magic quadrant for Cloud Infrastructure as a Service [Электронный ресурс] – 2020. – Режим доступа: https://pages.awscloud.com/Gartner-Magic-Quadrant-for-Infrastructure-as-a-Service-Worldwide.html. – Дата доступа: 25.02.2020

SYSTEM OF RECOGNITION OF OBJECTS IN VIDEO FLOW ON THE BASIS OF VIRTUALIZATION TECHNOLOGIES AND CLOUD COMPUTING

E.A. Pats
undergraduate student of the BSUIR,
Cloud Operations Engineer Netcracker Technology

K.V. Nasuro
assistant professor at BSUIR,
Candidate of Technical Sciences

Belarusian state university of informatics and radio electronics, Republic of Belarus
E-mail: egor.pats@gmail.com

Abstract. Cloud technologies is a promising and widely used way to process and store data. Using virtualization technologies can increase the fault tolerance of the services provided, reduce the time of code delivery and improve the utilization of computing resources. Thus, there is need to use cloud technology to solve various problems. Video stream processing is one of such tasks. A method for building a cloud platform architecture for processing a large number of streams in real time is proposed. The tools for configuring and deploying the system were discussed.

Keywords. Cloud technologies, virtualization technologies, IaaS, Terraform, IaC, configuration management systems, Chef, Puppet, Ansible, Salt, PaaS, Kubernetes.