# ИСПОЛЬЗОВАНИЕ СИСТЕМ УПРАВЛЕНИЯ КОНТЕЙНЕРАМИ ДЛЯ ПОСТРОЕНИЯ РАСПРЕДЕЛЕННЫХ ОБЛАЧНЫХ ИНФОРМАЦИОННЫХ СИСТЕМ С МИКРОСЕРВИСНОЙ АРХИТЕКТУРОЙ

**А.С. Бондаренко,** *аспирант*
**К.С. Зайцев,** *д-р тех. наук, доцент*
**Национальный исследовательский ядерный университет «МИФИ» (НИЯУ МИФИ) (Россия, г. Москва)**

*DOI:10.24412/2500-1000-2022-1-1-62-65*

***Аннотация.*** *В статье рассматривается микросервисная архитектура с использованием систем управления контейнерами. Микросервисная архитектура в основном применяется для распределенных приложений в промышленных системах и применение в системе управления контейнерами в настоящее время приобретает все большую актуальность. Решения, построенные с использованием контейнеров и вышеупомянутой архитектуры позволяют эффективно распределять ресурсы, при снижении стоимости разработки, ведь контейнеры – содержат все необходимые зависимости, что позволяет запускать написанный однажды код программы без изменения конфигурации в различных средах. Микросервисная архитектура позволяет выборочно развертывать и масштабировать компоненты приложений. Данный подход является весомой альтернативой масштабированию монолитных приложений, обусловленному чрезмерной нагрузкой на один компонент.*

***Ключевые слова:*** *распределенные системы, контейнеры, микросервисы, архитектура систем, облачная информационная система.*

В настоящее время распределенные информационные системы получили широкое распространение благодаря развитию компьютерных сетей и росту производства компьютерной техники. В то же время конфиденциальная информация, такая как персональные данные, должна обрабатываться, что выдвигает более высокие требования к оценке безопасности распределенных информационных систем. В настоящее время проводится анализ безопасности протокола обмена данными. Актуальность данной работы обусловлена, с одной стороны, необходимостью обеспечения информационной безопасности при распределенном обмене информацией, а с другой стороны, высокой вычислительной сложностью изучения безопасности информационных систем на основе разработанных протоколов. Контроль доступа гарантирует, что только авторизованные пользователи имеют доступ к данным и услугам. Эта проблема становится сложной в распределенных системах, где координация деятельности центрального органа может оказаться невозможным или может быть требовательна к ресурсам.

В распределенных системах существуют современные проблемы доступа, такие как мобильные сети, транспортные сети, интеллектуальные сети и "облачные вычисления". Каждое из этих приложений имеет различные ограничения и требования. Термин "облачные вычисления" широко используется, но различные облачные модели отличаются друг от друга с точки зрения безопасности. Поэтому важно, чтобы организации не применяли общий подход ко всем моделям. Облачные модели можно разделить на:

- программное обеспечение как услуга (SoftwareasaService, SaaS);
- платформы как услуга (PlatformasaService, PaaS);
- инфраструктура как услуга (интеграция как услуга, IaaS) [1].

Архитектура микросервисов – это подход к разработке приложений в виде набора небольших независимых сервисов, каждый из которых выполняется как отдельный независимый процесс. Службы взаи-

модействуют друг с другом с помощью простых механизмов, обычно по протоколу HTTP, что позволяет реализовать каждую из служб с использованием другого набора библиотек и языков программирования.

Подходы, используемые для разработки микросервисов, очень близки философии Unix. Они сформулированы в форме четырех основных правил [2]:

1. сервисы малы и достаточны для реализации одной функции системы;

2. в разработке поощряется автоматизация развертывания и тестирования;

3. отказы систем и оборудования учитываются в дизайне систем как регулярное событие жизненного цикла;

4. все сервисы масштабируемы и сочетаемы.

Одним из перспективных подходов к разработке распределенных систем является их организация в виде комплекса независимо развертываемых модулей, обменивающихся данными путем передачи сообщений, – микросервисов. Как правило, каждый отдельный микросервис реализует некоторую функцию, осмысленную с точки зрения пользователя (бизнес-функцию), полностью скрывая детали реализации, в том числе, способ сохранения данных – в таблицах реляционной базы данных, в документах документно-ориентированной базы данных, и т.п. В результате при развертывании каждого микросервиса выполняется развертывание соответствующей системы хранения данных, используемой только этим микросервисом. Такой подход позволяет снизить взаимное влияние модулей системы и обеспечить возможность их независимого масштабирования и управления доступностью. В то же время микросервисы являются модулями одной целостной системы, и распределение данных между ними должно осуществляться с учетом глобальных критериев.

Развертывание приложений очень часто является трудной работой для разработчиков. Например, ранее можно было столкнутся с «DLL Hell» – тупиковая ситуация, связанная с управлением динамическими библиотеками DLL в операционной системе Microsoft Windows. Несмотря на то, что прошло много времени, ежедневно растущий поток новых технологий зачастую создает путаницу и неуверенность [3].

Практически во всех случаях разработки ПО среда разработки значительно отличается от окружения, в котором приложение реально будет работать. Тот факт, что различные компьютеры будут сконфигурированы по-разному – очевиден, но при этом различное поведение приложения на этих компьютерах недопустимо.

С учетом описанных трудностей, контейнерные технологии фактически стали революцией в деплойменте, а разработки, такие как Docker, привели к быстрому росту популярности.

По моему мнению, контейнеры – то недостающее звено, которое вывело микросервисы в мейнстрим. Контейнеры обеспечивают простое развертывание. В результате сборки получается контейнер, который можно развернуть практически на любой машине, не требуя дополнительных пререквизитов. Хотя, конечно, есть особенности использования контейнеров под ОС Windows и Linux, но в пределах разных ОС Linux можно развернуть контейнер на любой Linux-машине. Необходимо настроить правило параметризации этого контейнера (например, определенным контейнерам нужно настроить параметры доступа к файловому хранилищу, проставить переменные окружения), и контейнер готов к запуску. Это позволяет развернуть контейнер в нескольких экземплярах, на одной машине или на десятке серверов.

Основная особенность контейнеров – это то, что они являются неизменяемыми (immutable). Если вы создали контейнер один раз, он сохраняет полный слепок файловой системы [4]. В любой момент, независимо от среды запуска, это будет полностью идентичная копия сервиса, включая ОС и все необходимые dll-библиотеки. Такой контейнер будет работать везде одинаково: на машине разработчика, в тестовом окружении или в production.

Очень часто контейнеры пытаются сравнивать с виртуальными машинами. Надо понимать, что между ними суще-

ствует принципиальное различие. Принцип работы виртуальных машин: есть некая хост операционная система (далее – гипервизор), которая абстрагирует хост ОС от виртуальных машин, и образы, в каждом из которых работает сама ОС, а в ней файловая подсистема и ваше приложение [5]. Обычно ОС на порядок больше по размерам и по потреблению ресурсов, чем непосредственно приложение, которое там работает. Использовать такую архитектуру для микросервисов – очень избыточно. Когда Azure только начинал развиваться, у них было решение PaaS Application Services. Запуская на этой платформе небольшой веб-сайт в трех экземплярах, необходимо было физически запустить три виртуальные машины, на которых процессор был загружен на 1 %, а 60 % оперативной памяти забирала на себя ОС.

**Заключение**

Основным преимуществом контейнеров, особенно в сравнении с виртуальными машинами, является уровень абстракции, который обеспечивает их простоту и переносимость.

Простота: контейнеры используют общие ресурсы ядра ОС хоста, что позволяет отказаться от полного экземпляра ОС для каждого приложения, а также уменьшить размер файлов контейнеров. Меньший размер контейнеров, особенно в сравнении с виртуальными машинами, означает возможность быстрого запуска и лучшую поддержку облачных приложений с возможностью горизонтального масштабирования.

Переносимость и платформонезависимость: контейнеры содержат все необходимые зависимости, что позволяет запускать написанный однажды код программы без изменения конфигурации на портативных компьютерах, в облачных и локальных вычислительных средах.

Поддержка современных архитектур и подходов к разработке: ввиду удачного сочетания совместимости на разных платформах и небольшого размера контейнеры являются идеальным выбором для современных инструментов и паттернов разработки, таких как DevOps, бессерверные вычисления и микросервисы, которые предусматривают регулярное развертывание кода небольшими фрагментами.

Более эффективное использование: как и виртуальные машины ранее, контейнеры позволяют разработчикам и операторам улучшить использование ресурсов процессора и памяти в физических системах. Поддержка микросервисных архитектур как дополнительное преимущество контейнеров позволяет выборочно развертывать и масштабировать компоненты приложений. Это очень заманчивая альтернатива масштабированию монолитных приложений, обусловленному чрезмерной нагрузкой на один компонент.

**Библиографический список**

1. Грушин Д.А., Кузюрин Н.Н. О задаче эффективного управления вычислительной инфраструктурой // Труды ИСП РАН. – 2018. – Т. 30, Вып. 6. – С. 123-142. DOI: 10.15514/ISPRAS-2018-30(6)-7

2. Шабанов Б.М., Самоваров О.И. Принципы построения межведомственного центра коллективного пользования общего назначения в модели программно-определяемого ЦОД // Труды ИСП РАН. – 2018. – Т. 30, Вып. 6. – С. 7-24. DOI: 10.15514/ISPRAS-2018-30(6)-1

3 Воеводин Вл.В., Попова Н.Н. Инфраструктура суперкомпьютерных технологий // Программирование. – 2019. – Т. 45, №3. – С. 6-13.

4. Крюков А.П., Демичев А.П. Децентрализованные хранилища данных: технологии построения // Программирование. – 2018. – Т. 44, №5. – С. 12-30.

5. Hoff T. Amazon Architecture. High Scalability, 2007. [Online]. Available: http://highscalability.com/blog/2007/9/18/amazon-architecture.html. [Accessed: 07-Nov-2018].

# USING CONTAINER MANAGEMENT SYSTEMS TO BUILD DISTRIBUTED CLOUD INFORMATION SYSTEMS WITH MICROSERVICE ARCHITECTURE

**A.S. Bondarenko,** *Postgraduate Student*
**K.S. Zaytsev,** *Doctor of Technical Sciences, Associate Professor*
**National Research Nuclear University MEPhI (Moscow Engineering Physics Institute) (NRNU MEPhI)**
**(Russia, Moscow)**

***Abstract.*** *The article discusses microservice architecture using container management systems Microservice architecture is mainly used for distributed applications in industrial systems and application in the container management system is currently becoming increasingly relevant. Solutions built using containers and the aforementioned architecture allow efficient allocation of resources, while reducing development costs, because containers contain all the necessary dependencies, which allows you to run once written program code without changing the configuration in different environments. Microservice architecture allows you to selectively deploy and scale application components. This approach is a powerful alternative to scaling monolithic applications due to excessive load on a single component.*

***Keywords:*** *distributed systems, containers, microservices, system architecture, cloud information system.*