Контейнеризация в системах обработки данных

Селезнёв Александр Игоревич, студент магистратуры;
Селезнёв Игорь Львович, кандидат технических наук, доцент
Белорусский государственный университет информатики и радиоэлектроники (г. Минск, Беларусь)

В статье рассматривается использование технологии контейнеризации в процессах обработки данных. Обосновывается необходимость выбора этой технологии, проводится анализ актуального программного обеспечения и сравнение с другими инструментами обработки данных, а также приводятся достоинства и недостатки технологии контейнеризации.

Ключевые слова: *контейнер, операционная система, Docker, виртуальная машина, программное обеспечение.*

С каждым годом объем информации в мире стремительно увеличивается, поэтому возникает необходимость в эффективной и быстрой обработке данных. Обработка данных — это систематизированная последовательность операций над исходными данными для получения новой информации путем различных вычислений и преобразований. Для реализации этой процедуры используются сложные компьютеризированные системы, включающие в себя аппаратное и программное обеспечение. Аппаратное обеспечение, как правило, представляет собой компьютерное оборудование. К программному обеспечению относятся следующие типы компонентов:

1. Аппаратно-программное обеспечение, в качестве которого выступает «базовая система ввода/вывода» (BIOS).
2. Операционная система (ОС) — программный комплекс, предназначенный для организации взаимодействия с пользователем.
3. Прикладное программное обеспечение — программы, предназначенные для выполнения определенных задач пользователя, часто в интерактивном режиме [1].

Для организации эффективной системы обработки данных классическим методом необходимо предварительно пройти различные этапы построения инфраструктуры: выбор места для размещения физического оборудования, закупка, установка и настройка необходимого программного обеспечения, а также постоянный мониторинг и обслуживание. С другой стороны, существует иной метод, который позволяет перенаправить большую часть ресурсов и затрат с построения предвари-

тельной инфраструктуры непосредственно на задачи, связанные с обработкой данных — использование виртуализации. Виртуализация — это процесс, при котором один системный ресурс, такой как оперативная память, центральный процессор, диск или сеть, может быть виртуализирован и представлен в виде множества ресурсов. С помощью данной технологии можно добиться построения гибких, легко изменяемых и высокопроизводительных систем обработки данных благодаря возможности динамического распределения ресурсов и минимизации затрат на обслуживание оборудования. Классическим типом виртуализации являются виртуальные машины. Однако при обработке возрастающих потоков данных проявляются недостатки классической виртуализации: многообразие установленного прикладного обеспечения вносит сложность в обновление и бесконфликтную работу в отдельно взятой виртуализированной среде. Решением данной проблемы является изоляция таких программных комплексов — но с использованием классической виртуализации это означает создание многих виртуализированных сред со своими ОС, что в конечном счете потребует организации сложной системы взаимодействия. Как один из вариантов решения этой проблемы на практике получил распространение другой тип виртуализации — контейнеры.

Контейнер — это альтернативный тип виртуализации, использующий изолированную среду для исполнения приложений. Сборка контейнеров осуществляется в ограниченном пространстве среды исполнения, которая содержит все необходимые компоненты — от системных инструментов до библиотек. Эти ресурсы называются виртуализированными именами. Все компоненты контейнеров взаимосвязаны, но выход за пределы их сред ограничен [2]. Технология контейнеризации — это метод, с помощью которого программный код упаковывается в единый исполняемый файл вместе с библиотеками и зависимостями, чтобы обеспечить его корректный запуск. Структура контейнера представлена на рисунке 1.

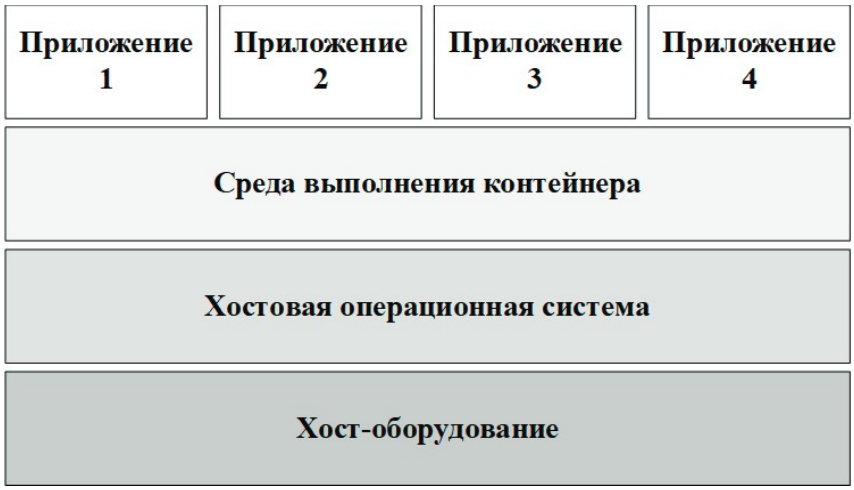

Рис. 1. **Структура контейнера**

Контейнеры обладают рядом преимуществ:

1. Высокая скорость выполнения. Контейнеры не перегружены излишними накладными расходами, потому что совместно используют ОС основного хоста. Эта облегченная структура обеспечивает более высокую эффективность сервера и сокращает время запуска. Более высокая скорость исполнения снижает нагрузку на сервер.

2. Локализация ошибок. Каждое контейнерное приложение работает независимо, что упрощает выявление различных сбоев. Пока происходит устранение возникшей проблемы, остальные контейнеры могут работать без простоев.

3. Высокая эффективность. Контейнеры имеют меньший размер чем виртуальные машины, быстро загружаются и обладают большей вычислительной мощностью, ввиду того что они предназначены для выполнения узкого круга задач. Эти характеристики делают контейнеры более эффективными, особенно при обработке ресурсов, а также снижают затраты на сервер и лицензирование.

4. Безопасность. Контейнерные приложения изолированы, а значит, если вредоносный код проникнет в одно из приложений, система управления средой не позволит распространиться вирусу за пределы текущего контейнера. Разработчики могут определять требования безопасности, которые контролируют доступ и обмен данными между контейнерами.

5. Перераспределение ресурсов. Технология контейнеризации позволяет ограничивать различные ресурсы: память, локальное хранилище, центральный процессор и другие. Каждый раз при завершении выполнения отдельных функций освободившиеся ресурсы распределяются на другие контейнеры в зависимости от рабочей нагрузки и установленных ограничений [3].

В контейнерах хранятся компоненты, необходимые для запуска требуемого программного обеспечения. К таким компонентам относятся файлы, переменные окружения, зависимости и библиотеки. ОС хоста ограничивает доступ контейнера к физическим ресурсам, таким как процессор, хранилище и память, поэтому один контейнер не может использовать ресурсов больше, чем ему выделено изначально.

Контейнеры по своей технологии схожи с виртуальными машинами. Виртуальные машины — это сложные

программные пакеты, которые обеспечивают полную эмуляцию низкоуровневых аппаратных устройств, таких как центральный процессор, дисковые и сетевые устройства. Виртуальная машина также может включать дополнительный программный стек для запуска на эмулируемых аппаратных средствах. Такие пакеты аппаратных и программных средств представляют собой полнофункциональный снимок вычислительной системы. Основное различие контейнеров и виртуальных машин заключается в том, что виртуальные машины могут виртуализировать весь компьютер вплоть до аппаратных уровней, а контейнеры — только программные уровни выше уровня ОС [4].

Файлы образов контейнеров — это полные, статические и исполняемые версии приложений или сервисов, которые различаются в зависимости от технологии. Одним из наиболее популярных решений технологии контейнеризации является платформа Docker, работающая в ОС Linux [5].

Docker — это платформа контейнеризации с открытым исходным кодом, с помощью которой можно автоматизировать создание и управление приложениями. Эта платформа ускоряет тестирование и создание приложений, а также позволяет запускать на одной машине требуемое количество контейнеров. В основе Docker лежит идея: разработанное приложение на одной машине должно полноценно запускаться и работать на любой другой машине. Для этого необходимо упаковывать настройки окружения вместе с приложением [6]. Образы Docker состоят из нескольких слоев, что представлено в примере на рисунке 2.

На рисунке 2 показано, что слои 1–3 представляют собой статический образ, данные из которого могут быть прочитаны, но не могут быть изменены. Слой 4 представляет собой образ записи/чтения для последнего используемого контейнера, в который происходит запись файлов из слоя 2. Далее с помощью специальных возможностей Docker происходит создание Docker тома, который может монтироваться как обычная файловая система с возможностью чтения и изменения данных как приложениями, которые работают вне контейнера, так и другими контейнерами. Поскольку каждый контейнер имеет свой собственный слой, содержащий описание конфигурации, базовые слои образов могут быть сохранены и повторно использованы в нескольких контейнерах. Аналогичным образом в контейнере могут одновременно работать несколько экземпляров образа, а новые экземпляры могут заменять неправильно работающие образы без нарушения работы всего приложения. Ввиду популярности контейнеров на сегодняшний день существуют и другие решения, например, такие как Podman, Buildah и Buildkit.

В современном мире требуется все больше взаимосвязанной и безопасной обработки различных типов информации, поэтому возникла необходимость в создании чего-то большего, чем простые интерфейсы, связывающие отдельно взятые контейнеры — средства оркестрации контейнеров. Оркестрация контейнеров — это программная технология, позволяющая автоматически контролировать запуск, остановку и масштабирование контейнеров. Она позволяет управлять множеством контейнеров, работающих на одном или нескольких хостах, как единой системой, примерами которой являются Kubernetes

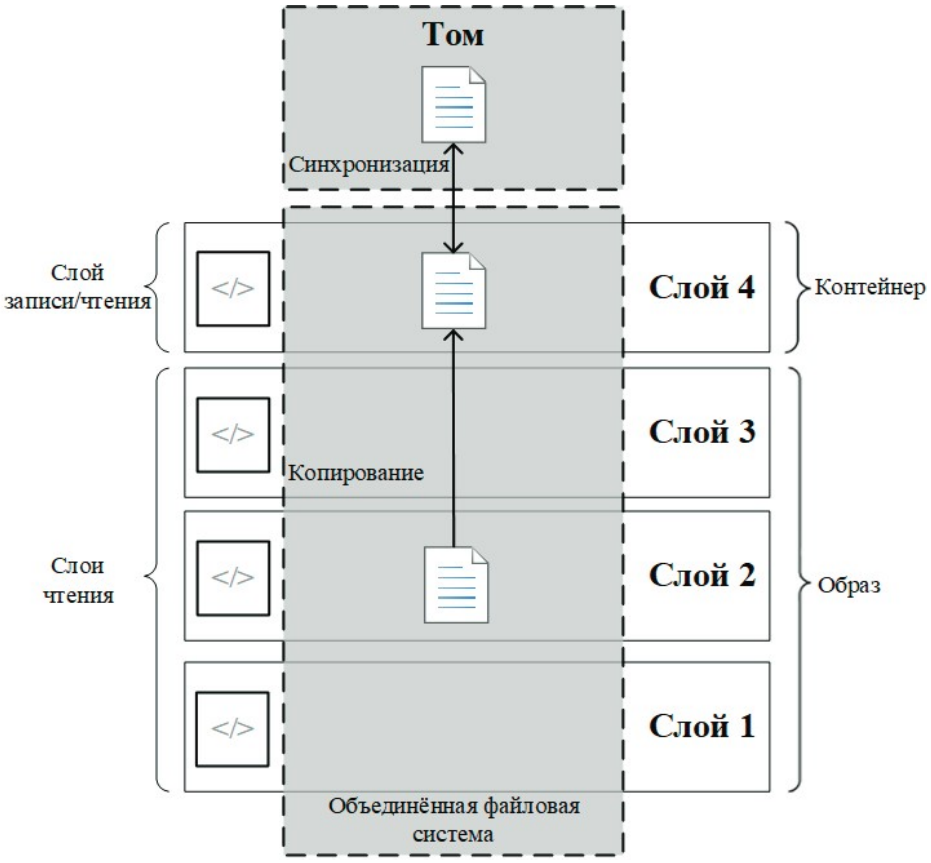

Рис. 2. **Пример образа Docker**

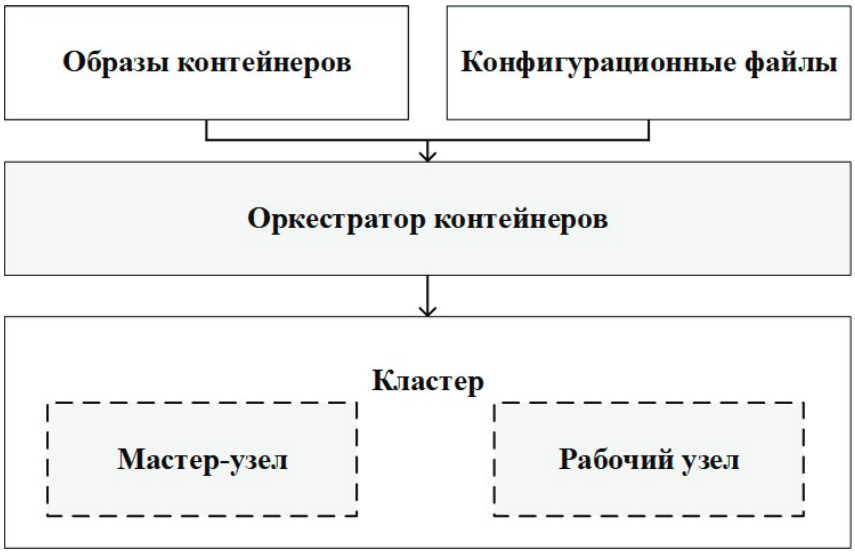

Рис. 3. **Система оркестрации контейнеров Kubernetes**

и Docker Swarm. На рисунке 3 представлена структурная схема системы Kubernetes.

Из рисунка 3 видно, что первоначальные образы контейнеров и конфигурационные файлы поступают в оркестратор контейнеров, который формирует специальный кластер, содержащий мастер-узел и рабочий узел. Мастер-узел управляет кластером посредством команд и инструкций, отправляемых администратором. Рабочий узел предназначен для стабильной работы приложений, которые содержатся в контейнерах [7].

Проведенный анализ технологии контейнеризации показывает, что она обладает следующими достоинствами:

1. Переносимость. Вследствие того, что приложения в контейнерах работают в собственной среде, они легко переносимы между разными платформами, что облегчает разработку приложений, так как они не зависят от ОС хоста.
2. Легковесность. Контейнеры потребляют меньше ресурсов, чем виртуальные машины, и быстрее запускаются.
3. Изоляция. Контейнеры предоставляют хорошо изолированные среды, что повышает безопасность и стабильность работы приложений.
4. Масштабируемость. Контейнеры могут быть легко масштабированы для удовлетворения растущей нагрузки.
5. Управляемость. Создание и удаление контейнеров, сборка шаблона для тестирования — все эти процедуры при предварительной настройке легко контролируются системами оркестрации.

В свою очередь, технология контейнеризации обладает следующими недостатками:

1. Ограниченность. Все контейнеры на одном хосте для работы требуют одинаковую версию ОС, что может ограничить их переносимость.
2. Частичная изоляция. В отличие от виртуальных машин, где используется полная изоляция, контейнеры изолированы на уровне ОС.
3. Уязвимость. Так как ОС у контейнеров общая, то уязвимость в ядре системы несёт риски и для контейнеров.
4. Сложность. Технология контейнеризации продолжает развиваться, и для успешной реализации требуются наличие опыта и знаний [8].

Выводы

Контейнеризация — относительно новая технология, которая получила широкое распространение благодаря высокой популярности Docker в 2014 году. С тех пор интерес к использованию контейнеров при разработке своих приложений для обработки различной информации неизменно растет как у крупных корпораций, так и у небольших компаний ввиду несомненных преимуществ, которые предоставляет эта технология. Контейнеризация применяется в таких технологиях, как NGINX, Redis, MySQL, GitLab и многих других.

Будущее контейнеров тесно связано с технологиями оркестрации, в частности, с Kubernetes, и тенденции их применения проявляются в облачных сервисах и вычислениях, из чего можно сделать вывод, что они еще долго будут являться ключевым компонентом при разработке приложений обработки данных.

Литература:

1. Система обработки данных.— Текст: электронный // EUCIP: [сайт].— URL: https://eopearhiiv.edu.ee/e-kursused/eucip/arendus_vk/111____.html (дата обращения: 19.10.2023).
2. Container.— Текст: электронный // Serverspace: [сайт].— URL: https://serverspace.by/support/glossary/container/ (дата обращения: 19.10.2023).

3. Что такое контейнеризация.— Текст: электронный // Yandex Cloud: [сайт].— URL: https://cloud.yandex.ru/docs/glossary/containerization (дата обращения: 19.10.2023).
4. Сравнение контейнеров и виртуальных машин.— Текст: электронный // Atlassian: [сайт].— URL: https://www.atlassian.com/ru/microservices/cloud-computing/containers-vs-vms (дата обращения: 19.10.2023).
5. Gillis, A. S. Containers (container-based virtualization or containerization) / A. S. Gillis.— Текст: электронный // Techtarget: [сайт].— URL: https://www.techtarget.com/searchitoperations/definition/container-containerization-or-container-based-virtualization (дата обращения: 19.10.2023).
6. Как устроен Docker и почему он популярен.— Текст: электронный // Yandex Cloud: [сайт].— URL: https://cloud.yandex.ru/blog/posts/2022/03/docker-containers (дата обращения: 19.10.2023).
7. Important Kubernetes Concepts Made Easy.— Текст: электронный // In28minutes Cloud: [сайт].— URL: https://cloud.in28minutes.com/kubernetes-for-beginners-important-concepts-simplified (дата обращения: 19.10.2023).
8. Чем отличается виртуализация от контейнеризации.— Текст: электронный // Cloud4y: [сайт].— URL: https://www.cloud4y.ru/blog/containerization-vs-virtualization/ (дата обращения: 19.10.2023).