ОБ АВТОРЕ

Баланов Антон Николаевич имеет большой опыт руководства и консультирования в сфере ИТ-технологий. Работал топ-менеджером в крупных компаниях — таких, как Industrial and Commercial Bank of China (КНР), Caravan portal (ОАЭ), Банк ВТБ, Сбербанк России, VK; руководил разработками сервиса Gosuslugi.ru. Имеет степень MBA IT (CIA) и сертификации Microsoft, CompTIA, ISACA, PMI, SHRM, IIBA, HRCI, ISO, Six Sigma (Master Black Belt). Преподавал в следующих вузах и учебных центрах: Российском университете дружбы народов, СберУниверситете, Институте бизнеса и делового администрирования и Центре подготовки руководителей и команд цифровой трансформации (на базе Высшей школы государственного управления РАНХиГС). Автор десятков книг и научно-практических публикаций в профессиональных изданиях. Является советником Российской академии естественных наук.

Широкая эрудиция и глубокие профессиональные компетенции автора в сфере ИТ-технологий позволили ему создать книжную серию «Айтишный университет», один из выпусков которой находится перед вами.

ОГЛАВЛЕНИЕ

ГЛАВА 1

ВВЕДЕНИЕ В МИКРОСЕРВИСЫ И ВЫСОКОНАГРУЖЕННЫЕ ПРИЛОЖЕНИЯ

ВВЕДЕНИЕ

Глава 1 посвящена введению в микросервисы и высоконагруженные приложения. Мы определим понятия микросервисной архитектуры и высоконагруженных приложений, рассмотрим их преимущества и вызовы, а также изучим основные концепции и принципы, лежащие в основе этих подходов.

Микросервисная архитектура — это подход к разработке приложений, в котором приложение состоит из небольших автономных сервисов, каждый из которых выполняет свою специфическую функцию. В высоконагруженных приложениях, которые обрабатывают большие объемы запросов и данных, требуется особая организация и архитектура для обеспечения масштабируемости, отказоустойчивости и производительности.

В первой части главы мы определим понятия микросервисной архитектуры и высоконагруженных приложений. Мы разберемся в том, как эти подходы отличаются от традиционных монолитных приложений и как они взаимодействуют совместно для достижения высокой производительности и масштабируемости.

Во второй части главы мы рассмотрим преимущества и вызовы, с которыми сталкиваются разработчики при использовании микросервисов и разработке высоконагруженных приложений. Мы изучим преимущества, такие как гибкость, независимость развертывания и масштабирования, а также вызовы, связанные с управлением распределенными системами, поддержкой сетевой связности и обеспечением безопасности и целостности данных.

В третьей части главы мы обратим внимание на основные концепции и принципы, лежащие в основе микросервисов и высоконагруженных приложений. Мы изучим архитектурные принципы, такие как разделение обязанностей, горизонтальное масштабирование и управление данными. Мы также рассмотрим инструменты и платформы, которые помогают реализовать эти концепции, такие как контейнеризация и оркестрация.

Изучение микросервисов и высоконагруженных приложений позволит нам понять, как эти подходы могут помочь разработчикам создавать масштабируемые, отказоустойчивые и производительные приложения. Мы узнаем о преимуществах и вызовах, связанных с этими подходами, и о том, какие концепции и принципы следует учитывать при работе с ними.

ОПРЕДЕЛЕНИЕ ПОНЯТИЙ МИКРОСЕРВИСНОЙ АРХИТЕКТУРЫ И ВЫСОКОНАГРУЖЕННЫХ ПРИЛОЖЕНИЙ

Микросервисная архитектура

Микросервисная архитектура — это подход к разработке программного обеспечения, при котором приложение разбивается на набор небольших и независимых компонентов, называемых микросервисами. Каждый микросервис выполняет специфическую функцию и может разрабатываться, развертываться и масштабироваться отдельно от других микросервисов. Микросервисы взаимодействуют между собой через легковесные протоколы, такие как HTTP или сообщения.

Высоконагруженные приложения

Высоконагруженные приложения — это приложения, которые обрабатывают большие объемы данных или запросов с высокой производительностью и отзывчивостью. Они часто встречаются в сферах электронной коммерции, социальных сетей, финансовых услуг и других областях, где требуется обработка и хранение большого количества информации и обеспечение быстрого ответа на запросы пользователей.

Таблица 1.1

Понятие	***Описание***
Микросервисная архитектура	Подход к разработке ПО, при котором приложение разбивается на небольшие независимые компоненты.
Высоконагруженные приложения	Приложения, обрабатывающие большие объемы данных или запросов с высокой производительностью.

Микросервисная архитектура представляет собой подход к разработке программного обеспечения, который способствует гибкости, масштабируемости и независимости компонентов приложения. Вместо того, чтобы создавать монолитное приложение, микросервисы разрабатываются и развертываются отдельно, позволяя независимую разработку и масштабирование каждого компонента. Такая архитектура обеспечивает лучшую отдельность ответственности, упрощает разработку и поддержку, а также улучшает масштабируемость и гибкость системы в целом.

Высоконагруженные приложения, с другой стороны, характеризуются способностью обрабатывать большие объемы данных или запросов с высокой производительностью. Они требуют оптимизации и масштабирования, чтобы обеспечить быстрый и отзывчивый ответ на запросы пользователей. Высоконагруженные приложения широко используются в таких отраслях, как электронная коммерция, социальные сети и финансовые услуги, где требуется обработка большого объема данных в реальном времени.

Примеры микросервисной архитектуры и высоконагруженных приложений могут включать следующие.

1. *Netflix*. Netflix — популярная платформа для потокового видео, которая использует микросервисную архитектуру для обеспечения масштабируемости и гибкости. Они разделяют свое приложение на набор микросервисов, таких как сервисы для управления пользователями, поиском, рекомендациями

и т.д. Каждый микросервис имеет свою собственную базу данных и взаимодействует с другими микросервисами через API. Это позволяет Netflix масштабироваться, обновлять и внедрять новые функции независимо от других сервисов.

2. *Uber*. Uber — международный сервис такси и доставки, который также использует микросервисную архитектуру для управления своей платформой. У них есть различные микросервисы для управления пользователями, обработки платежей, маршрутизации и т.д. Каждый микросервис имеет свою собственную функциональность и может масштабироваться отдельно от других сервисов.

3. *Ампер Электро.* Ампер Электро — сеть профильных магазинов электротоваров с более чем 100 000 позиций, доступных к заказу, а также собственным производством. Использует микросервисную архитектуру для распараллеливания процессов обработки данных. Микросервис 1 отвечает за хранение и управление информацией о товарах. Микросервис 2 обеспечивает более быстрый и точный поиск на сайте. Микросервис 3 позволяет обрабатывать и оптимизировать изображения. Система позволяет легко масштабироваться горизонтально и обрабатывать большие объемы данных и запросов.

4. *Airbnb*. Airbnb — платформа для аренды жилья, которая также использует микросервисную архитектуру для обеспечения гибкости и масштабируемости. У них есть различные микросервисы для управления листингами жилья, обработки платежей, обратной связи и т.д. Это позволяет Airbnb добавлять новые функции и поддерживать высокую производительность при обработке большого числа бронирований и запросов от пользователей.

5. *Amazon*. Amazon — крупнейшая электронная коммерция и облачный провайдер, использует микросервисную архитектуру для управления своей инфраструктурой и различными сервисами. У них есть множество микросервисов, которые обрабатывают функции, такие как управление заказами, платежами, хранение, доставка и т.д. Микросервисная архитектура помогает Amazon масштабироваться горизонтально и обеспечивает отказоустойчивость и высокую производительность.

В целом, комбинация микросервисной архитектуры и высоконагруженных приложений позволяет создавать гибкие, масштабируемые и производительные системы, способные эффективно обрабатывать большие объемы данных и запросов, и является ключевым фактором в развитии современных приложений и сервисов.

ПРЕИМУЩЕСТВА И ВЫЗОВЫ ПРИ ИСПОЛЬЗОВАНИИ МИКРОСЕРВИСОВ И РАЗРАБОТКЕ ВЫСОКОНАГРУЖЕННЫХ ПРИЛОЖЕНИЙ

Микросервисная архитектура и разработка высоконагруженных приложений представляют собой современные подходы, которые позволяют создавать гибкие и масштабируемые системы. Рассмотрим преимущества и вызовы, связанные с использованием микросервисов и разработкой высоконагруженных приложений.

1. Преимущества микросервисов

Микросервисная архитектура предлагает ряд преимуществ по сравнению с традиционными монолитными приложениями.

а) *Гибкость и масштабируемость*. Микросервисы разделены на небольшие и автономные службы, что позволяет гибко масштабировать их индивидуально в зависимости от нагрузки. Таким образом, ресурсы могут быть эффективно использованы, а система может быть легко масштабирована при необходимости.

б) *Разработка и развертывание*. Микросервисы могут быть разработаны и развернуты независимо друг от друга. Это упрощает процесс разработки, тестирования и обновления приложения. Команды могут работать параллельно над разными сервисами, что ускоряет разработку и обеспечивает независимость между службами.

в) *Устойчивость и отказоустойчивость*. Если один микросервис выходит из строя, остальные сервисы продолжают работать нормально. Это позволяет создавать более устойчивые системы и изолировать проблемы, минимизируя влияние отказов.

Таблица 1.2

Преимущества микросервисов

Преимущество	***Описание***
Гибкость и масштабируемость	Возможность гибкого масштабирования отдельных сервисов в зависимости от нагрузки
Независимая разработка и развертывание	Возможность независимой разработки и развертывания микросервисов
Устойчивость и отказоустойчивость	Способность системы продолжать работать при отказе отдельных микросервисов

2. Вызовы при использовании микросервисов

Хотя микросервисы предлагают ряд преимуществ, существуют и некоторые вызовы, с которыми сталкиваются разработчики при работе с ними.

а) *Сложность управления*. С ростом количества микросервисов возрастает сложность управления системой в целом. Контроль версий, мониторинг, отладка и координация между сервисами могут быть сложными задачами, требующими специальных инструментов и процессов.

б) *Сетевая сложность*. В микросервисной архитектуре взаимодействие между сервисами осуществляется через сеть. Это может приводить к проблемам с производительностью, задержками и отказами сети. Надежное и эффективное управление сетью становится важным аспектом для обеспечения стабильной работы системы.

в) *Мониторинг и отладка*. В микросервисной архитектуре отслеживание и отладка проблем становятся сложнее из-за распределенности сервисов и увеличенного числа компонентов. Необходимо использовать инструменты мониторинга, журналирования и трассировки, чтобы эффективно обнаруживать и устранять проблемы *(см. Табл. 1.3)*.

Примеры

1. *Netflix*. Netflix является примером компании, которая успешно применила микросервисную архитектуру. Они раз-

Таблица 1.3

Вызовы при использовании микросервисов

Вызов	***Описание***
Сложность управления	Управление системой, включая контроль версий, мониторинг и координацию, может стать сложной задачей
Сетевая сложность	Взаимодействие через сеть может вызывать проблемы с производительностью и надежностью сети
Мониторинг и отладка	Отслеживание и отладка распределенных сервисов становятся сложными задачами, требующими специализированных инструментов

били свою платформу на множество микросервисов, что позволило им гибко масштабировать систему и быстро вносить изменения в отдельные компоненты.

2. *Uber*. Uber также использует микросервисную архитектуру для своей платформы. Каждая функциональность, такая как заказ такси, платежи и управление водителями, представлена отдельным микросервисом. Это позволяет им гибко масштабировать каждый компонент системы и управлять ими независимо.

Микросервисная архитектура предоставляет гибкость и масштабируемость, позволяет независимо разрабатывать и развертывать сервисы, а также обеспечивает устойчивость и отказоустойчивость системы. Однако она также представляет вызовы, связанные с управлением сложной системой и обеспечением стабильной работы сети.

ОСНОВНЫЕ КОНЦЕПЦИИ И ПРИНЦИПЫ, ЛЕЖАЩИЕ В ОСНОВЕ УКАЗАННЫХ ПОДХОДОВ

При использовании микросервисной архитектуры и разработке высоконагруженных приложений следует придерживаться ряда основных концепций и принципов, которые помогут

обеспечить эффективность, масштабируемость и надежность системы.

1. Разделение по границам предметной области

Каждый микросервис должен отвечать только за определенную часть функциональности или предметную область приложения. Это позволяет достичь лучшей изоляции, модульности и возможности развития каждого сервиса независимо от других. Разделение по границам предметной области облегчает разработку, поддержку и масштабирование приложения.

Таблица 1.4

Пример для разделения по границам предметной области

Микросервис	***Предметная область***
Аутентификация	Управление доступом
Управление заказами	Управление заказами
Отправка уведомлений	Уведомления

2. Автономность и независимость сервисов

Каждый микросервис должен быть автономным и независимым от других сервисов. Это означает, что каждый сервис должен иметь свою собственную базу данных, конфигурацию и управлять своим состоянием. Это обеспечивает изоляцию и позволяет легко масштабировать и развертывать отдельные сервисы.

Таблица 1.5

Пример для автономности и независимости сервисов

Микросервис	***База данных***	***Конфигурация***
Аутентификация	auth_db	auth_config
Управление заказами	orders_db	orders_config