Обзор распределенных систем

Распределенные системы представляют собой фундаментальную эволюцию архитектуры, которая происходит естественным образом по мере того, как программные системы выходят за пределы возможностей одной машины. Современные приложения SaaS неизбежно становятся распределенными системами, когда они реализуют CDN, развертывают базы данных в нескольких зонах доступности или используют реплики для чтения для оптимизации производительности. Эти архитектурные решения превращают простые приложения в сложные распределенные среды, требующие тщательной координации между компонентами.

Распределенные системы — это не просто академические конструкции, а практические решения, отвечающие реальным требованиям к масштабируемости, надежности и производительности, которые не могут быть удовлетворены монолитными архитектурами.

Неизбежный путь от простых к распределенным

Системы обычно начинаются с простых архитектур: одного сервера, одной базы данных и, возможно, балансировщика нагрузки. Однако рост неизбежно приводит к усложнению, которое стимулирует распределение архитектуры.

Требования к производительности со стороны географически распределенных пользователей требуют внедрения CDN, что приводит к проблемам с обновлением кэша в нескольких периферийных точках. Ограничения производительности баз данных в периоды пиковой нагрузки требуют использования реплик для чтения, что в конечном итоге приводит к необходимости управления согласованностью между основными хранилищами данных и хранилищами реплик.

Этот эволюционный процесс превращает простые приложения в сложные распределенные системы, состоящие из десятков сервисов, взаимодействующих между несколькими регионами, с кэшами, очередями, базами данных и сторонними API, требующими координации и синхронизации.

Заблуждения о распределенных вычислениях

Распределенные системы фундаментально нарушают допущения, лежащие в основе разработки программного обеспечения для одной машины. Питер Дойч и его коллеги определили эти заблуждения как «заблуждения о распределенных вычислениях», которые при игнорировании приводят к дорогостоящим архитектурным ошибкам:

Сеть надежна

В сетях происходят потери пакетов, таймауты соединений и полные отключения центров обработки данных.

Задержка равна нулю

Каждый сетевой вызов влечет за собой измеримую задержку, часто в миллисекундах или секундах.

Пропускная способность бесконечна

Ограничения пропускной способности сети в конечном итоге сдерживают высокопроизводительные приложения.

Сеть безопасна

Весь сетевой трафик потенциально может быть перехвачен без надлежащего шифрования.

Топология не меняется

Балансировщики нагрузки перезапускаются, службы мигрируют, а конфигурации DNS постоянно развиваются.

Есть один администратор

Распределенные системы требуют координации между несколькими командами и административными доменами.

Транспортные расходы равны нулю

Передача данных влечет за собой реальные финансовые затраты, особенно между регионами или поставщиками.

Сеть однородна

Разные службы работают на различной инфраструктуре с разными характеристиками производительности.

Каждое заблуждение представляет собой модель сбоя, которая проявляется в производственных условиях, часто в критические периоды эксплуатации.

Основные проблемы распределенных систем

Основная сложность распределенных систем связана с требованиями к координации, а не с отдельными технологическими компонентами. Одномашинные системы используют преимущества общей памяти, атомарных операций и гарантий согласованности операционной системы. Распределенные архитектуры устраняют эти основополагающие допущения, вводя новые категории сложности.

Частичные сбои представляют собой наиболее серьезную проблему. Одномашинные системы демонстрируют двоичные режимы сбоев — полный успех или полный сбой. В распределенных системах происходят частичные сбои, при которых отдельные компоненты работают или выходят из строя независимо друг от друга, создавая несогласованные состояния. Успешная обработка платежа в сочетании с сбоем уведомления по электронной почте приводит к тому, что клиенты получают счета без подтверждения, что требует сложных механизмов обработки ошибок и восстановления.

Временная неопределенность усложняет упорядочение событий между распределенными компонентами. Упорядочение на основе временных меток становится ненадежным, когда события происходят на машинах с независимыми часами. Определение последовательности операций, таких как обновление профиля и изменение пароля, зависит от синхронизации часов и протоколов координации, а не от простого сравнения временных меток.

Протоколы консенсуса налагают значительную нагрузку на распределенное согласование. Достижение консенсуса по выбору лидера, упорядочению операций или завершению транзакций требует нескольких сетевых циклов и тщательной реализации протоколов, что существенно ограничивает производительность системы и увеличивает ее сложность.

Что такое протоколы консенсуса и зачем они нужны

В распределённых системах данные хранятся и обрабатываются сразу на нескольких серверах (узлах). Чтобы система работала корректно, все узлы должны **сойтись на одном и том же состоянии данных** — например, какой блок данных записан последним или какой сервер сейчас считается лидером.

Для этого используются **протоколы консенсуса** — наборы правил взаимодействия между узлами, которые помогают им согласованно принимать решения, даже если связь нестабильна или некоторые узлы выходят из строя.

Протоколы консенсуса обеспечивают:

- **надёжность** — система продолжает работу даже при сбоях отдельных узлов
- **согласованность данных** — все узлы видят одинаковый результат
- **устойчивость к ошибкам и сетевым задержкам**

Современные решения и инструменты

В облачной экосистеме разработаны сложные решения для управления сложностью распределенных систем. Kubernetes предоставляет механизмы обнаружения служб и восстановления после сбоев. Сервис-меши, такие как Istio, обеспечивают автоматическое шифрование, балансировку нагрузки и возможности наблюдаемости. Управляемые службы баз данных обрабатывают распределенный консенсус и сложность репликации.

Kubernetes

Механизмы обнаружения служб и восстановления после сбоев

Сервис-меши

Автоматическое шифрование, балансировка нагрузки и наблюдаемость

Управляемые БД

Обработка распределенного консенсуса и репликации

Kafka

Слабо связанные, в конечном итоге согласованные архитектуры

Платформы потоковой передачи событий, такие как Kafka, позволяют создавать слабо связанные, в конечном итоге согласованные архитектуры, которые хорошо соответствуют реалиям распределенных систем. Инструменты распределенного трассирования, такие как Jaeger, обеспечивают видимость потоков запросов между несколькими сервисами, что позволяет отлаживать и оптимизировать производительность.

В последующих главах рассматриваются концепции распределенных систем с точки зрения как теоретических основ, так и практической реализации.