ВЕБ-УРОВЕНЬ БЕЗ СОХРАНЕНИЯ СОСТОЯНИЯ

Пришло время поговорить о горизонтальном масштабировании веб-уровня. Для этого нужно вынести из него состояние (например, информацию о пользовательских сеансах). Данные сеансов рекомендуется записывать в постоянные хранилища, такие как реляционные БД или NoSQL. Каждый веб-сервер в кластере может запросить состояние из базы данных. Таким образом получается веб-уровень без сохранения состояния.

Архитектура с сохранением состояния

От того, хранит сервер состояние или нет, зависит, будет ли он «помнить» данные клиента (состояние) между разными запросами.

На рис. 1.12 показан пример архитектуры с сохранением состояния.

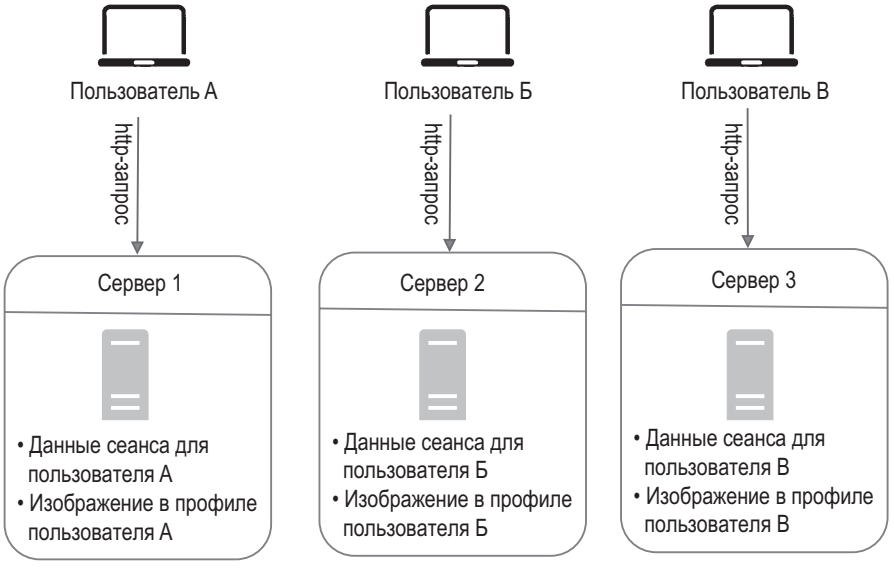

Рис. 1.12

На рис. 1.12 данные сеанса и изображение в профиле пользователя А хранятся на сервере 1. Чтобы аутентифицировать пользователя А, HTTP-

запрос должен быть направлен к этому серверу. Если отправить этот запрос, к примеру, серверу 2, аутентификация не пройдет, так как на втором сервере нет данных соответствующего сеанса. Точно так же все HTTP-запросы пользователя Б должны направляться к серверу 2, а запросы пользователя В — к серверу 3.

Проблема в том, что каждый запрос с отдельно взятого клиента необходимо оправлять на соответствующий сервер. В большинстве балансировщиков нагрузки для этого предусмотрены липкие сеансы [10], но такой подход увеличивает накладные расходы. Из-за него добавление и удаление серверов дается с трудом. Также возникают проблемы, если сервер выходит из строя.

Архитектура без сохранения состояния

На рис. 1.13 показана архитектура без сохранения состояния.

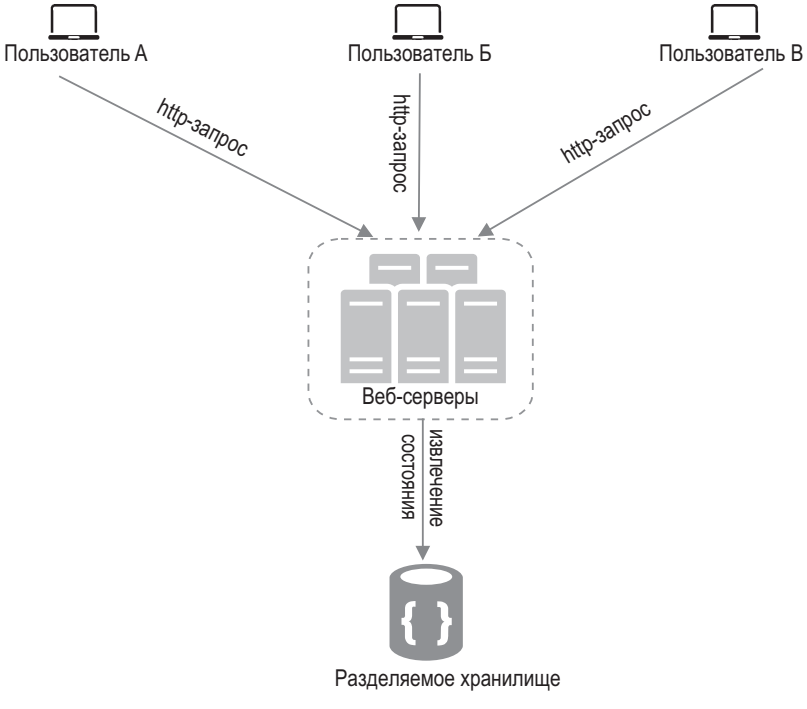

Рис. 1.13

В этой не хранящей состояние архитектуре пользовательские HTTP-запросы могут быть направлены любым веб-серверам, которые извлекают данные о состоянии из общего хранилища. Хранилище отделено от веб-серверов. Отсутствие состояния делает систему более простой, надежной и масштабируемой.

На рис. 1.14 показана обновленная конфигурация с веб-уровнем, не хранящим состояние.

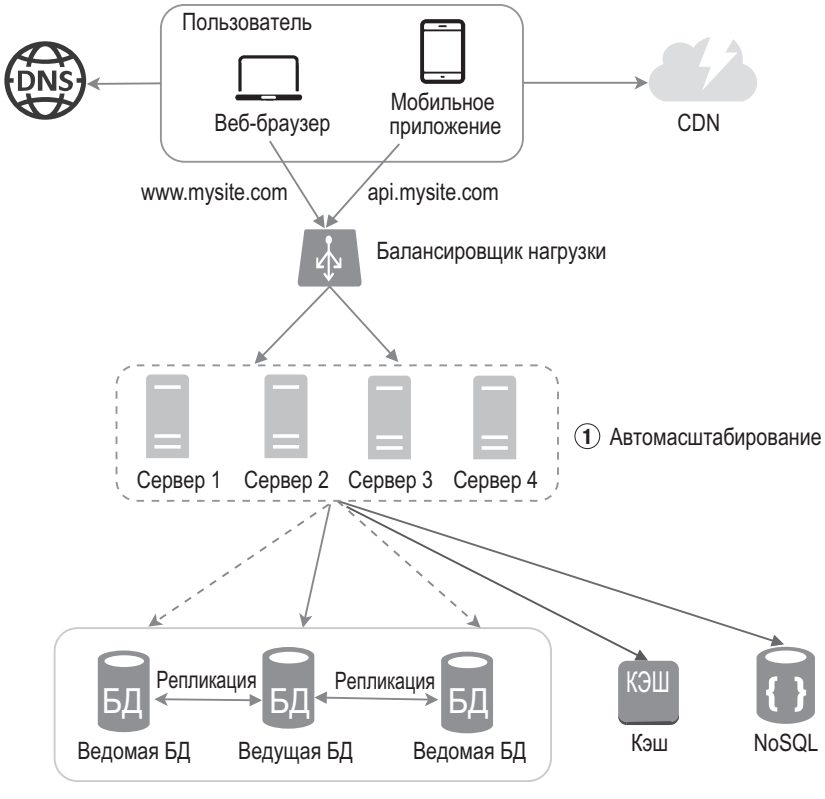

Рис. 1.14

На рис. 1.14 данные сеанса вынесены из веб-уровня и теперь находятся в постоянном хранилище, роль которого могут играть реляционные базы данных: Memcached/Redis, NoSQL и т. д. Здесь хранилище NoSQL выбрано в связи с простотой его масштабирования. Автомасштабирование означает, что добавление и удаление веб-серверов происходит автоматически в за-

висимости от объемов трафика. После того как данные о состоянии вынесены в отдельное хранилище, автомасштабирование веб-уровня легко достигается за счет добавления и удаления серверов с учетом нагрузки.

Ваш веб-сайт стремительно развивается, привлекая множество пользователей со всего мира. Для улучшения доступности и UX в различных регионах крайне необходима поддержка нескольких центров обработки данных.

ЦЕНТРЫ ОБРАБОТКИ ДАННЫХ

На рис. 1.15 показана демонстрационная конфигурация с двумя центрами обработки данных (ЦОД). В нормальных условиях пользователи, скажем,

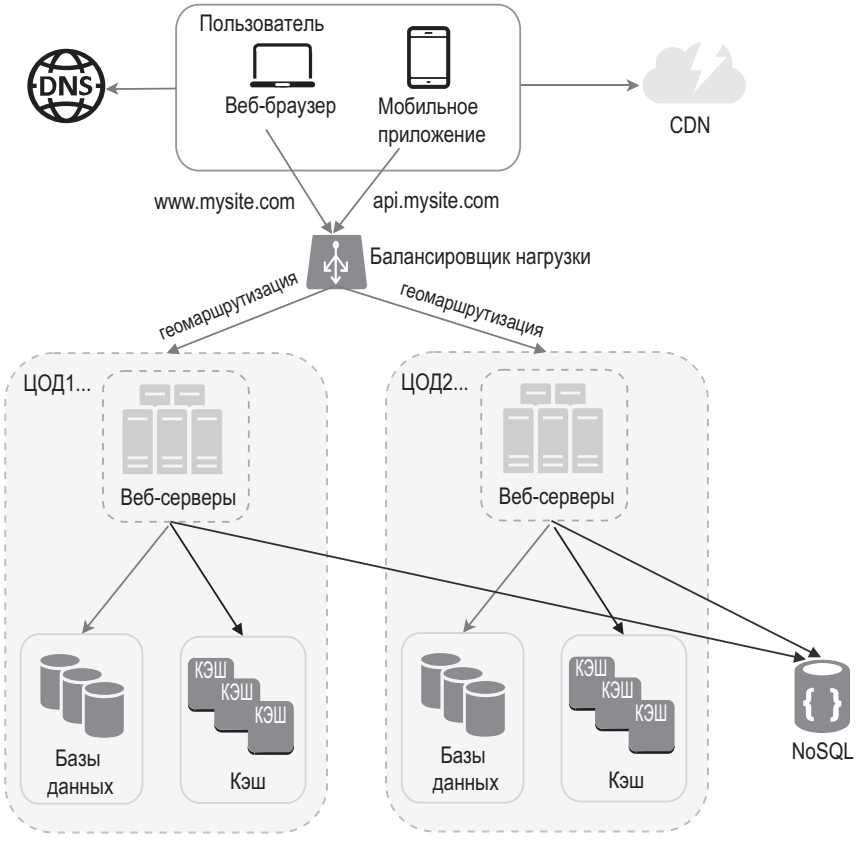

Рис. 1.15

из США с помощью geoDNS направляются к ближайшему центру обработки данных с разделением трафика между регионами US-East и US-West в пропорции x % к (100 – x) %. Это называется географической маршрутизацией. geoDNS — это сервис, который сопоставляет доменные имена с IP-адресами в зависимости от местонахождения пользователя.

В случае любого серьезного нарушения работы одного из центров обработки данных мы перенаправляем весь трафик к исправному ЦОД. На рис. 1.16 ЦОД2 (US-West) недоступен, поэтому 100 % трафика направляется к ЦОД1 (US-East).

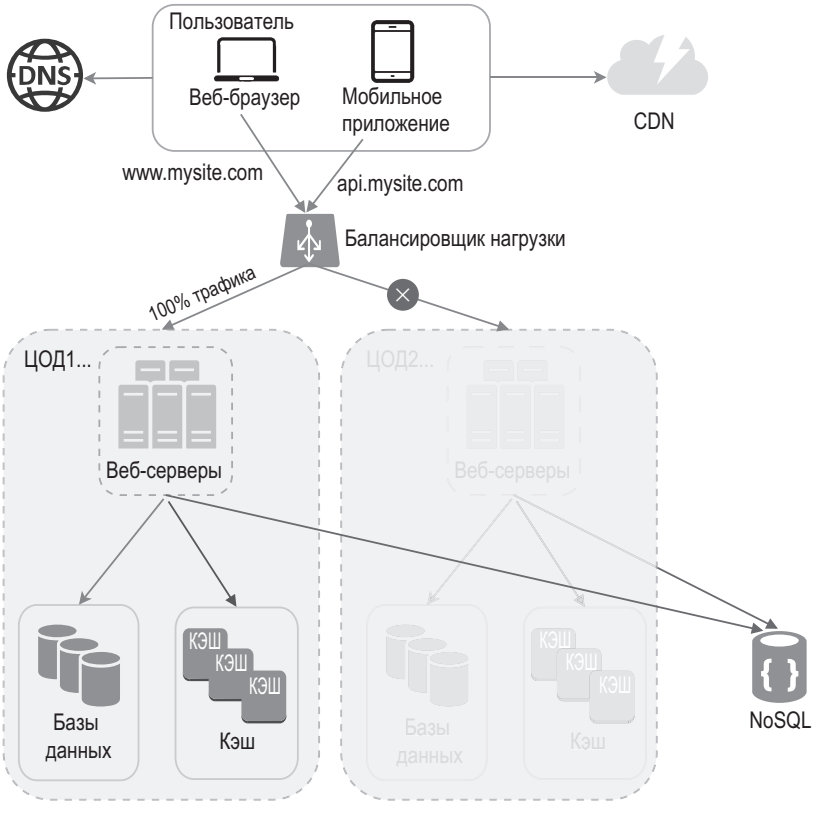

Рис. 1.16

Для реализации архитектуры с несколькими центрами обработки данных необходимо решить несколько технических вопросов.

- Перенаправление трафика. Необходимы эффективные инструменты для направления трафика к подходящему ЦОД. GeoDNS позволяет выбирать центр обработки данных, который находится ближе всего к пользователю.
- Синхронизация данных. Пользователи могут работать с разными локальными базами данных и кэшами в зависимости от региона. В случае сбоя трафик может быть перенаправлен к ЦОД, в котором нет запрашиваемых данных. Распространенным решением является репликация данных между несколькими ЦОД. В одном из исследований показано, как Netflix реализует асинхронную репликацию между разными центрами обработки данных [11].
- Тесты и развертывание. В конфигурации с несколькими ЦОД тестирование веб-сайта/приложения необходимо проводить в разных местах. Автоматические средства развертывания незаменимы в поддержании согласованности всех ЦОД [11].

Чтобы еще сильнее улучшить масштабируемость нашей системы, мы должны разделить ее компоненты; это позволит масштабировать их независимо друг от друга. Во многих реальных распределенных системах для решения этой задачи используют очереди сообщений.

ОЧЕРЕДЬ СООБЩЕНИЙ

Очередь сообщений — это устойчивый компонент, который загружается в память и поддерживает асинхронное взаимодействие. Он служит буфером и распределяет асинхронные запросы. Очередь сообщений имеет простую базовую архитектуру. Сервисы ввода, так называемые производители/издатели, создают сообщения и публикуют их в очереди. Другие сервисы или серверы, которые называют потребителями/подписчиками, подключаются к очереди и выполняют действия, определенные в сообщениях. Эта модель показана на рис. 1.17.

Благодаря разделению очередь сообщений является предпочтительной архитектурой для создания масштабируемых и надежных приложений. Производитель может публиковать сообщения в очереди, даже если потребитель не в состоянии их обработать. И наоборот — потребитель может считывать сообщения из очереди, даже если производитель недоступен.