Подробнее о Stateless и Stateful архитектуры для масштабирования

Stateless (Без сохранения состояния)

В stateless архитектуре каждый запрос от клиента к серверу является независимым. Сервер не хранит информацию о предыдущих запросах. Это значит, что каждый запрос должен содержать всю информацию, необходимую для его обработки.

Пример:

HTTP и REST являются хорошими примерами stateless протоколов и архитектур. Каждый HTTP-запрос содержит всю информацию, необходимую для его обработки.

Когда используется:

1. Когда нужно масштабировать приложение горизонтально.
2. Когда операции являются независимыми друг от друга.
3. В микросервисных архитектурах для упрощения управления и развертывания.

Реализация:

1. **Без сохранения состояния на сервере**: Вся информация для обработки запроса поступает с самим запросом. Например, если пользователь запрашивает доступ к каким-то данным, токен авторизации будет отправлен в каждом запросе.
2. **Кеширование на клиенте**: Часто используется кеширование на стороне клиента для улучшения производительности.
3. **Балансировка нагрузки**: Так как каждый сервер является независимым, можно легко распределить нагрузку, добавляя или убирая серверы.

Stateful (С сохранением состояния)

В stateful архитектуре сервер хранит информацию о предыдущих запросах. Это может быть полезно, например, для отслеживания состояния пользователя в сессии.

Пример:

FTP (File Transfer Protocol) — пример stateful протокола. Сервер помнит предыдущие действия пользователя, такие как аутентификация и текущая директория, чтобы можно было выполнять последующие операции.

Когда используется:

1. В реальном времени и интерактивных приложениях, где важен контекст (например, чаты, онлайн-игры).
2. В транзакционных системах, где нужно сохранить состояние между несколькими шагами.

Реализация:

1. **Сессии**: Состояние пользователя обычно хранится в сессиях на сервере.
2. **Базы данных**: Состояние может храниться и в базе данных для более долгосрочного хранения.
3. **In-memory хранилища**: Для быстрого доступа к состоянию могут использоваться in-memory хранилища типа Redis.
4. **Sticky Sessions в балансировщиках нагрузки**: Если приложение stateful, балансировщик нагрузки должен направлять пользователя на тот сервер, на котором хранится его сессия.

Сравнение подходов:

1. **Масштабируемость**: Stateless приложения проще масштабировать, так как каждый запрос можно обработать независимо.
2. **Сложность**: Stateful приложения часто сложнее в управлении и требуют дополнительных ресурсов для хранения состояния.
3. **Производительность**: Stateful приложения могут быть быстрее на определенных этапах операции, так как они "помнят" предыдущие действия, но они также могут требовать больше ресурсов для хранения этих данных.
4. **Надежность**: В stateful архитектуре больше рисков, связанных с потерей данных, так как состояние хранится на сервере.

В зависимости от требований вашего приложения, можно использовать один из этих подходов или комбинировать их для достижения оптимальной производительности и надежности. Но в основном сейчас используются Stateless подходы в крупных системах - так как они нуждаются в горизонтальном масштабировании.