УДК 004.85

Макаров Д.А.
студент
факультет «Информатика и системы управления»
Московский государственный технический
университет имени Н.Э.Баумана
Россия, г. Москва
Шибанова А.Д.
студент
факультет «Робототехника и комплексная автоматизация»
Московский государственный технический
университет имени Н.Э.Баумана
Россия, г. Москва

МАСШТАБИРОВАНИЕ ВЕБ-ПРИЛОЖЕНИЙ

Аннотация: в данной статье рассмотрены механизмы масштабирования веб-приложений. Были описаны причины возникновения проблем с доступом к приложению при росте нагрузки. Проведен анализ возможных узких мест при разработке приложений. Даны рекомендации по их устранению.

Ключевые слова: архитектура, масштабирование, нагрузка, база данных, кеширование.

Makarov D.A.
student
faculty of informatics and management systems
Moscow State Technical University named after N.E. Bauman
Russia, Moscow
Shibanova A.D.
student
faculty of robotics and complex automation
Moscow State Technical University named after N.E. Bauman
Russia, Moscow

SCALING WEB APPLICATIONS

Abstract: This article discusses mechanisms for scaling web applications. The reasons for the problems with access to the application when the load increases were described. The analysis of possible bottlenecks in application development is carried out. Recommendations for their elimination are given.

Keywords: architecture, scaling, load, database, caching.

Когда вы создаете отличный продукт или приложение, рано или поздно оно будет привлекать внимание все большего и большего числа пользователей, которые будут ожидать безупречного, совершенного приложения. Если не быть к этому готовым, производительность приложения начнет падать, и вы потеряете свою аудиторию и бизнес. В этой статье будет показано, на что следует обратить внимание при создании масштабируемого приложения.

Что такое масштабируемость приложения?

Масштабируемость приложения - это способность приложения расти со временем, способность эффективно обрабатывать все больше и больше запросов в минуту (RPM). Это не просто настройка, которую можно включить/выключить; это длительный процесс, который затрагивает почти каждый элемент в вашем стеке, включая как аппаратную, так и программную части системы. [1]

В случае возникновения проблем вы можете продолжать добавлять новые процессоры или увеличивать память, но тем самым вы просто увеличиваете пропускную способность, а не производительность приложения. Это не тот путь, которого вы должны придерживаться, когда видите, что у вашего приложения возникают проблемы с эффективностью. Масштабирование приложения - непростая задача, поэтому вы должны очень хорошо знать свое приложение, прежде чем начинать думать о том, как и когда его масштабировать.

Проблемы с масштабированием приложений

Довольно часто, когда ваше приложение становится слишком большим, вы видите множество проблем с масштабируемостью, но возможность масштабирования приложения больше касается всей системной архитектуры.

Даже если у вас нет проблем с производительностью или масштабируемостью, таких как у Twitter или Shopify, правильное планирование и разработка приложения бесценны. Когда дело доходит до масштабирования, вы можете столкнуться с десятками различных проблем. Несколько общих источников ваших проблем могут быть связаны с:

- Ограниченные физические ресурсы, такие как память, процессоры и т.д.
- Неправильное управление памятью
- Неэффективный движок базы данных
- Сложная схема базы данных, плохая индексация
- Плохо выполняемые запросы к базе данных
- Неправильная конфигурация сервера
- Ограничения сервера приложений
- Общее качество кода
- Неэффективное кеширование
- Отсутствие инструментов мониторинга
- Слишком много внешних зависимостей

- Неправильный дизайн фоновых заданий [2]

Как сделать эффективное масштабирование?

Это кажется очевидным, но мы просто не можем не указать на это здесь. Написание правильного и продуктивного кода - ключ к масштабируемости вашего приложения.

Чтобы система могла расти плавно, вы должны позаботиться о базе данных. Выбрав подходящий механизм БД и спроектировав возможную надежную схему, вы сможете эффективно обрабатывать увеличивающееся количество транзакций в секунду.

Каждое отдельное веб-приложение выполняет огромное количество запросов к базе данных, даже если у вас есть прекрасно спроектированные отношения с правильной индексацией, работающие на лучшем движке на рынке, не забудьте эффективно запрашивать информацию у своей базы данных, избегая запросов N + 1, ненужной загрузки, и т.д.,

Мы должны помнить, что масштабируемость - это не только ваш код. Очень важно иметь правильную серверную инфраструктуру и конфигурацию. Вы можете сэкономить много времени, просто выбрав соответствующие инструменты и поставщиков. Например, Amazon EC2 предоставляет функции автоматического масштабирования, Docker предлагает масштабирование для создания контейнеров и т. д. [3]

Нам не нужен специальный компьютер для кэширования при запуске проекта, но это определенно хорошая практика для реализации, отслеживания и изменения ваших потребностей в кешировании с течением времени. Если мы не используем «готовую к работе» инфраструктуру, такую как AWS, помните, что однажды вы должны подготовиться к переходу от односерверной архитектуры к многосерверной с балансировкой нагрузки и обратным проксированием.

Если у нас будет возможность переместить код во внешний интерфейс, сделайте это. Ваш бэкэнд станет менее перегруженным, так как все больше и больше вещей будет вычисляться на стороне клиента.

Даже если наша кодовая база чистая и легко обслуживается, нам нужны инструменты для ее мониторинга и выявления проблем как можно скорее.

Выявление проблем - это одно, но мы должны воспользоваться имеющимися у нас инструментами и оптимизировать наш код и конфигурацию, чтобы минимизировать узкие места в вашем приложении.

Это также часть чистоты кода - старайтесь не смешивать слишком много частей вашей системы в одном месте. Разделяйте внешние и внутренние уровни, отделяйте фоновые задания от основной системы, разумно используйте шаблоны проектирования.

Держите все в актуальном состоянии, чтобы избежать блокировок из-за устаревших частей вашей системы (например, старой версии Ruby).

Эти несложные рекомендации позволяют эффективно масштабировать нагрузку на веб-приложение.

Использованные источники:

1. Ли Атчисон. Масштабирование приложений. Выращивание сложных систем.–Санкт-Петербург: Питер, 2018. – 256 с.
2. Масштабирование нагрузки web-приложений. URL: https://habr.com/ru/post/113992/ (дата обращения: 04.01.2021).
3. Архитектура высоких нагрузок. URL: https://ruhighload.com/Архитектура+высоких+нагрузок (дата обращения: 04.01.2021).