относительно неизменном уровне.

Проводя анализ зависимости з\с и времени ответа от количества пользователей (выборка из БД) для Spring приложения видно, как до 1000 пользователей время отклика остается низким, а производительность растет. Но после этого значения время отклика резко возрастает, а производительность растет. Производительность перестает расти после 2000.

Сравнивая результаты для Java EE и Spring при записи данных, можно сделать вывод, что для времени отклика оптимизированный Spring сработал более продуктивно. Но не на всей области сравнения, а только после 100-250 пользователей. При выборке данных Java EE дольше показывал стабильные результаты для времени отклика. Для производительности (з\с) Spring практически сразу стал показывать более высокие результаты.

В результате выполнения данной работы были разработаны веб-приложения в соответствии со спецификацией Java EE и основанное на технологии Spring и библиотеке Hibernate. Создана MySQL база данных, в которую веб-приложения производили запись и выборку данных. При создании приложения в связке Spring и Hibernate были предложены способы оптимизации работы с данными. Был проведен ряд тестов для определения производительности веб-приложений. Была определена точка насыщения – ключевое значение, при котором рост числа клиентов не повышает общей производительности приложения.

Список использованных источников:

1. Тестирование и анализ производительности с помощью сервера приложений WebSphere [Электронный ресурс]. Дата публикации: 13.06.2013. URL: http://www.ibm.com/developerworks/ru/library/wes-1208_hare/
2. Оптимизация и масштабирование Web приложений [Электронный ресурс]. Дата обновления: 04.10.2015. URL: http://ruhighload.com/
3. Spring MVC 3, Аннотации Hibernate, MySQL. Хабрахабр [Электронный ресурс]. Дата публикации: 23.01.2015. URL: http://habrahabr.ru/post/248541/

МИКРОСЕРВИСНАЯ АРХИТЕКТУРА ВЕБ-ПРИЛОЖЕНИЙ

Белорусский государственный университет информатики и радиоэлектроники
г. Минск, Республика Беларусь

Грученков В. В.

Новиков В. И. – канд. техн. наук, доцент

Растущий спрос на гибкость и масштабируемость для удовлетворения быстро растущих потребностей бизнеса создает потребность в применении новых подходов к проектированию архитектуры веб-приложений. В докладе рассматривается микросервисная архитектура, анализируются ее достоинства и недостатки.

Большинство современных Web-приложений включают три основные части: пользовательский интерфейс, база данных и сервер. Серверная часть обрабатывает HTTP запросы, запрашивает и обновляет данные в БД, заполняет HTML страницы, которые затем отправляются браузеру клиента. В подобных системах вся логика по обработке запросов выполняется в единственном процессе, любое изменение в системе приводит к пересборке и развертыванию новой версии серверной части приложения, а масштабирование осуществляется путем запуска дополнительных серверов, либо обновлением конфигурации существующих.

Решением вышеописанных проблем является использование микросервисной архитектуры. Микросервисная архитектура – это стиль построения сложных систем, предлагающих разбиение приложения на мелкие сервисы, каждый из которых работает в собственном процессе и взаимодействует с остальными используя легковесные механизмы, как правило протокол HTTP. Идея подобной архитектуры, однако, не является революционной - ее корни уходят далеко в прошлое, как минимум к принципам проектирования, использованным в системах Unix.

На рисунке 1 приведена приложение с микросервисной архитектурой:

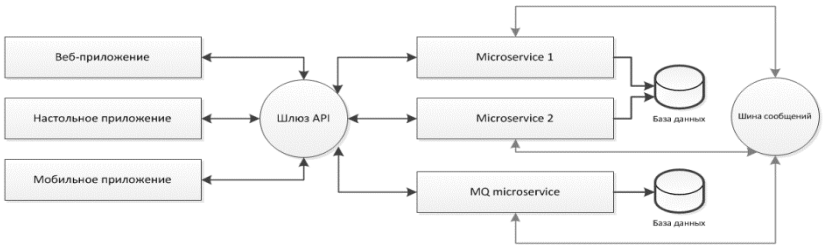

Рис. 1. Структурная схема приложения с микросервисной архитектурой

Преимущества микросервисной архитектуры:

– Техническое разнообразие - то, что каждый компонент изолирован от остальных, дает

возможность использования наиболее подходящий для каждой задачи язык программирования;
- Горизонтальное масштабирование и отказоустойчивость – выполнение каждого сервиса в отдельном процессе позволяет увеличить производительность системы путем разнесение сервисов по отдельным физическим машинам, без необходимости внесения изменений в приложение. Отказ какого-либо из сервисов не ведет к отказу всей системы;
- Простота развертывания – в отличие от монолитных приложений, внесение изменений в которое требует перезапуска всей программы, внесение изменений в микросервисные приложения требуют перезапуска только изменившихся компонентов;
- Возможность повторного использования функциональности каждого из сервисов сторонними системами;
- Распределение работы между командам – поскольку каждый сервис представляет собой отдельный проект, работа над ними может быть легко разделена между разработчиками;
- Несмотря на очевидные достоинства, микросервисная архитектура не лишена недостатков, основными из которых являются:
- Увеличение сложности системы в целом;
- Дополнительные накладные расходы на передачу данных между микросервисами, сериализацию и десериализацию;
- Безопасность транзакций. Поддерживать безопасность транзакций при работе с независимыми процессами иногда оказывается нетривиальной задачей.

Несмотря на весь положительный опыт, нельзя однозначно утверждать, что микросервисы — это будущее проектирования ПО. Однако, уже сейчас ясно, что микросервисная архитектура обладает большим потенциалои и предлагает серьезные преимущества для разработки и реализации корпоративных приложений.

Список использованных источников:

1. Sam Newman. Building Microservices. Designing Fine-Grained Systems / Sam Newman //. –: O'Reilly Media, 2015. – 280 p.
2. Martin Fowler. Microservices. A definition of this new architectural term [Электронный ресурс] – Режим доступа: http://martinfowler.com/articles/microservices.html

ОБРАБОТКА И АНАЛИЗ ИЗОБРАЖЕНИЯ ОТПЕЧАТКА ПАЛЬЦА ВЫДЕЛЕНИЕ ОСОБЫХ ТОЧЕК ОТПЕЧАТКА

Белорусский государственный университет информатики и радиоэлектроники
г. Минск, Республика Беларусь

Козак А. В.

Теслюк В.Н. – канд. физ-мат. наук, доцент

Каждый человек наделен особой меткой, которая отличает его от других людей, - отпечатками пальцев. Отпечатки пальцев делают нас особенными, именно с помощью них (и не только) можно идентифицировать человека. Так дактилоскопия – это наука, изучающая способы распознания человека по отпечаткам ладони и пальцев рук, основанная на неповторимости папиллярных линий. Впервые дактилоскопию применили для распознания преступника в Великобритании в 1902г.

В каждом отпечатке можно выделить две группы признаков: локальные и глобальные. Глобальные признаки – те, которые можно увидеть невооруженным глазом. К ним относится, например, тип папиллярного узора («петля», «дуга» и «спираль»). Локальные признаки – минуции (особые точки). Минуции – это уникальные для каждого отпечатка признаки, определяющие пункты изменения структуры папиллярных линий (окончание, раздвоение, разрыв и др.), ориентацию папиллярных линий и координаты в этих пунктах. Каждый отпечаток может содержать от 70 минуций.

Не трудно понять, что глобальные признаки некоторых людей одинаковы, но совершенно невозможно наличие одинаковых микроузоров минуций. Поэтому глобальные признаки используют для разделения базы данных на классы и на этапе аутентификации. На втором этапе распознавания используют уже локальные признаки.

Выделяют несколько подходов к сравнению отпечатков пальцев.

Наиболее популярный подход основан на локальных признаках. Этот подход основан на выделение минуций и сопоставления их с другими образцами. Для определения соответствия входного отпечатка со сравниваемым образцом используется формула: $K = \frac{D^2}{pq} \cdot 100\%$, где D – количество совпавших минуций, p – количество минуций эталона, q – количество минуций опознаваемого отпечатка. В случае, если результат превышает 65 %, отпечатки считаются идентичными (порог может быть понижен выставлением другого уровня бдительности).

Рассмотрим механизм и алгоритмы обработки, анализа и выделения минуций отпечатка.

Существуют стандарты для изображения опечатка. Так считается, что для лучшей и качественной