- изменение самого вида зашифрованного сообщения происходит даже при незначительном изменении ключа шифрования;
- нет понятных зависимостей между ключами, если в процессе шифрования последовательно используются несколько ключей;
- все дополнительные биты, если таковые вводятся в сообщение, хорошо скрыты в зашифрованном сообщении;
- любой ключ из множества возможных способен обеспечить надежную защиту информации.

В ходе работы было изучено множество методов шифрования, используемых как в информационных системах, так и в повседневной жизни. Также были изучены различные особенности защиты информационных систем. Был проведен анализ стойкости различных методов криптографических преобразований. Выявленные в результате данного анализа положительные и отрицательные особенности различных методов криптографических преобразований позволили выделить наиболее верные пути защиты информации. Также были выявлены особенности некоторых методов защиты информации с точки зрения удобства для рядового пользователя.

Список использованных источников:

1. Скембрей Дж. Секреты хакеров / Дж. Скембрей, Ст. Мак-Клар – Москва: «Вильямс», 2004. – 512 с.
2. Смарт Н. Cryptography: An Introduction / Н. Смарт – Москва: «Техносфера», 2006. – 528 с.
3. Сингх С. Книга шифров / С. Сингх – Москва: «Астрель», 2006. – 447 с.

СРАВНИТЕЛЬНАЯ ХАРАКТЕРИСТИКА ВЫСОКОНАГРУЖЕННЫХ ВЕБ-ПРИЛОЖЕНИЙ НА ПЛАТФОРМЕ JAVA

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

Вакульчик Е. Н.

Куликов С. С. – к.т.н., доцент

Современным веб-приложениям приходится обрабатывать сотни и даже тысячи запросов в секунду. Чтобы обеспечить необходимую производительность, необходимо определить потенциальные узкие места и предпринять меры по их устранению. Стадии оптимизации производительности должна предшествовать стадия тщательного исследования и анализа.

В настоящее время существуют две конкурирующие платформы для разработки веб-приложений одинаковой сложности, использующих в качестве основы язык Java: Java EE и Spring. Целью данного исследования являлась разработка двух веб-приложений на платформах Java EE и Spring соответственно, осуществляющих запись и выборку данных из БД, и сравнение их производительности. В качестве СУБД использовался MySQL, а в качестве сервера приложений – GlassFish. Для проведения нагрузочного тестирования и отметки ключевых показателей использовалось средство автоматизации нагрузочного тестирования JMeter.

В связке Spring и Hibernate для эффективного доступа к базе данных использовался пул соединений, представляющий собой кэш для открытых соединений между приложением и СУБД. Использование такого решения позволяет сильно сократить накладные расходы на установку и инициализацию нового соединения между приложением и СУБД.

Нагрузочное тестирование веб-приложений проводилось с помощью варьирования количеством пользователей и регистрируя ключевые показатели. Наиболее важные регистрируемые показатели: пропускная способность (з\с - число запросов в секунду), время реакции, процент загрузки ЦП серверного компьютера, процент загрузки ЦП БД.

Анализируя результаты зависимости з\с и времени ответа от количества пользователей (запись в БД) для Java EE приложения можно сделать вывод о том, что до 750 количества пользователей производительность линейно возрастает, а время отклика остается относительно небольшим. От 750 до 1250 время отклика резко повышается, а производительность достигает своего максимума в точке 1250. Дальнейший рост числа пользователей только уменьшает производительность и значительно увеличивает время отклика.

Анализируя результаты зависимости з\с и времени ответа от количества пользователей (выборка из БД) для Java EE приложения можно сделать вывод, что до 1000-1250 пользователей производительность растет, а время отклика остается небольшим, после этого значения время отклика резко возрастает, а производительность при этом остается практически неизменной.

Проводя анализ зависимости з\с и времени ответа от количества пользователей (запись в БД) для Spring приложения, наблюдается, что до 1500-1750 пользователей время отклика остается на низком уровне, а производительность растет. Далее время отклика резко возрастает, а производительность остается на

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:

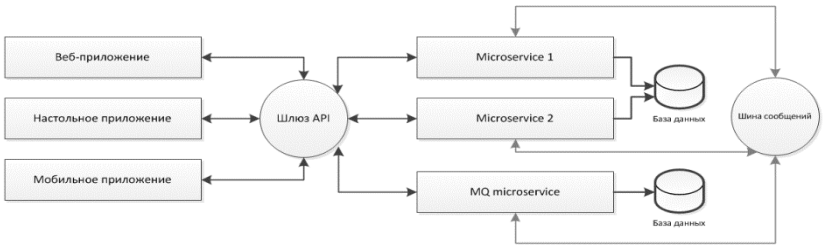

Рис. 1. Структурная схема приложения с микросервисной архитектурой

Преимущества микросервисной архитектуры:

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