Министерство образования Республики Беларусь
Учреждение образования
Белорусский государственный университет
информатики и радиоэлектроники

УДК 004.415.22

Намакарский
Андрей Александрович

Программное средство на базе PaaS платформы

АВТОРЕФЕРАТ
на соискание степени магистра информатики и вычислительной техники

по специальности 1-40 80 02 Технологии виртуализации и облачных вычислений

_____

Научный руководитель
Луцик Юрий Александрович
Кандидат технических наук, доцент

_____

Минск 2018

ВВЕДЕНИЕ

С развитием технологий архитектура серверной части приложений изменятся в сторону большей гибкости. Огромную популярность имеет конвейерная архитектура. Но, не смотря на все ее плюсы, такие как слабая связность компонентов, возможность довольно быстрого внесения изменений в обработчики, отличную горизонтальную масштабируемость, у этого архитектурного паттерна есть один большой недостаток – сложность настройки и поддержки серверного окружения. Существуют десятки инструментов для решения этой задачи, однако, они требую высокой квалификации системных администраторов. Убрав необходимость поддержи серверного окружения можно существенно снизить затраты на разработку проектов требующих наличия серверной части. Один из них это FaaS, который по свой сути близок к PaaS.

FaaS (Function as a service) – дословно переводится как “функция как сервис”, архитектурный паттерн относящийся к классу бессерверных архитектур. Идея данного архитектурного паттерна заключается в следующем: серверная часть приложения разбивается на набор функций, не хранящих состояния между вызовами, то есть результат выполнения функции не должен зависеть от состояния памяти сервера и локальной файловой системы, а зависит только лишь от переданных параметров. При наличии таких ограничений горизонтальное масштабирование представляется довольно простой задачей. Масштабирование выполняется провайдером для поддержания требуемого уровня производительности, в данном случае провайдером является FaaS платформа.

Принцип организации структурных компонентов серверной части при данном архитектурном походе показан на рисунке 1.1:

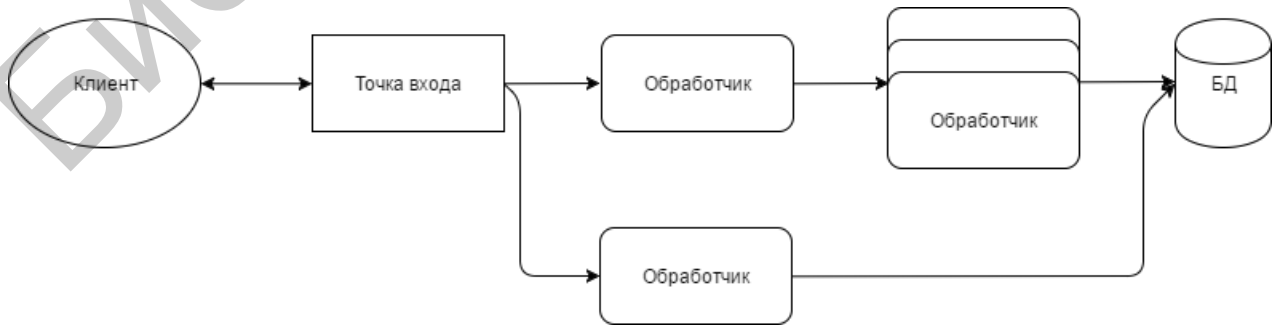

Рисунок 1.1 – Архитектурная схема с использованием FaaS подхода

На схеме видно, что для клиентской части приложения все выглядит так же, как и в случае с обычной клиент серверной архитектурой, что достигается с

помощью единой точки входа, которая является подобием роутера. Данное решение позволит не вносить изменения в клиентскую часть, если она уже существует.

Серверная часть обладает некоторыми особенностями:

a. Отсутствует необходимость в настройке и обслуживании серверов.

b. Отсутствует необходимость в использовании фреймворков, в данном случае обработчик будет простейшей программой, которая должна иметь определенный интерфейс. Это ведет к слабой связности компонентов.

c. Так как отсутствует серверная часть приложения, доставка изменений представляет собой простейшую задачу по загрузке кода. Например, в виде ZIP-архива.

d. Задача горизонтального масштабирования полностью ложится на плечи FaaS-платформы и от разработчиков не требуется никакой конфигурации.

e. Функции вызывается с помощью событий, определенных провайдером, например, запрос на определенный пользователем URL, по таймеру и так далее.

Однако, данный подход также не лишен недостатков. При запуске функции-обработчика тратится дополнительное время на запуск изолированной среды с пользовательским кодом. Также существует необходимость ограничения выполнения функции-обработчика по времени, на случай если внутри этой функции произошло зацикливание. Также наличие большого количества функций-обработчиков может добавить трудностей при поддержке приложений. Данная проблем может быть частично решена с использованием гибридной архитектуры, в которой некоторая часть проекта будет реализована с применением FaaS подхода.

ОБЩАЯ ХАРАКТЕРИСТИКА РАБОТЫ

Целью данной работы является исследование построениями бессерверных архитектур. Задачей является разработка программного комплекса позволяющего реализовывать бессерверные архитектуры с использованием FaaS подхода, с использованием PaaS платформы, разрабатываемой в рамках данной работы, и определение места данного подхода среди других облачных технологий.

Разработанное решение позволяет создавать приложения, основанные на небольших функциональных блоках – функциях, и их объедение в большие блоки для реализации полноценных приложений. Данный подход подобен микросервисной архитектуре за тем исключением, что он подразумевает более мелкое деление функционала на модули. Помимо этого подразумевается отсутствие необходимости настройки окружения для объединения функций в большие модули.

Разработанное решение позволяет избавиться от некоторых недостатков, которые присутствуют в популярных облачных платформах, таких как AWS Lambda, а именно- ограничения по времени выполнения функций и ограниченный набор языков программирования доступный конечным пользователем. Также данное решение позволяет избавиться от зависимости от третьей стороны, что может быть существенным ограничением для некоторых проектов.

Данное решение может быть полезно как малым, так и крупным компаниям, а также индивидуальным разработчикам, перед которыми стоит необходимость в разработке серверной части. Также данное решение может быть полезно для быстрого прототипирования проектов. Оно будет полезно разработчикам мобильных приложений, нуждающимся в написании серверной части приложения, но не имеющих специальных знаний, ввиду своей простоты и независимости от языка программирования.

КРАТКОЕ СОДЕРЖАНИЕ РАБОТЫ

В Обзоре литературных источников описывается проблематика данной работы, а также положение данной проблемы среди других решений. Делается обзор существующих решений в сфере облачных технологий. Также описываются специфика каждого облачного решения, сильные и слабые стороны. Описывается положение FaaS и PaaS среди облачных решений.

В теоретической части данной работы в полной мере описывается FaaS подход, его применимость для различных классов веб-приложений. Теоретические и практические детали дали подхода, его сильные и слабые стороны.

В практической части описывается примененный инструментарий для создания работы и проблемы, которые он позволяет решить. Описывается программная архитектура разработанного решения. Описывается схема эксперимента и делается краткий обзор его результатов.

В заключении делаются выводы о применимости данного решения, его достоинствах и недостатках.

ЗАКЛЮЧЕНИЕ

Результатом данной работы является комплекс программного обеспечения позволяющий строить приложения, основывающиеся на бессерверной архитектуре и проведенный сравнительный анализ достоинств и недостатков приложений основывающихся на данном классе архитектур.

Данный программный комплекс построен по принципу частной облачной платформы, которая позволяет развертывать как несколько небольших функций, так и тысячи. Построение по принципу частного облака позволит использовать данное программное обеспечение клиентам заботящихся о безопасности своих данных, либо не могущих позволить отдавать их хранение и обработку третьей стороне ввиду риска утечки.

Разработанное программное обеспечение реализует изоляцию пользовательских функций друг от друга с помощью инструмента контейнеризации Docker. Данный факт означает невозможность воздействия функций друг на друга. Также использование Docker использовать данное решение на основных операционных системах без необходимости дополнительной конфигурации.

Также данный комплекс разрабатывался с учетом потенциальных высоких нагрузок, что достигается с помощью кластерной архитектуры данного приложения. В короткие сроки приложение может быть масштабировано до десятков и сотен машин, входящих в кластер. В результате нагрузочного тестирования достигнуты показатели в пять запросов в секунду на вычислительном кластере, состоящем из двух машин имеющих всего одно вычислительное ядро и один гигабайт памяти.

Популярные решения для создания бессерверных архитектур, такие как AWS Lambda имею ряд недостатков, таких как:

- Ограниченное время выполнения функции.
- Ограничение по количеству вызываемых функций.
- Ограниченный набор доступных языков программирования

Что может быть сдерживающим фактором для некоторых классов задач. Разработанный комплекс не имеет этих ограничений. Время выполнения функций может быть ограничено администратором кластера, либо разработчиком функции. Количество вызываемых функций ограничено только лишь вычислительной мощностью кластера и может быть увеличено. Набор языков программирования не ограничен, так как используется платформа контейнеризации Docker а прием и передача данных внутрь функции осуществляется с помощью чтения стандартных потоков ввода и вывода. Также функции могут быть реализованы с помощью стандартных утилит UNIX-подобных систем. Помимо этого в качестве функции может быть использовано любое стороне программное обеспечения предоставляемое в виде бинарных исполняемых файлов UNIX и

поддерживающее ввод и вывод с помощью стандартных потоков ввода и вывода.

Однако помимо преимуществ данное программное обеспечение имеет ряд недостатков по сравнению с конкурентами:

- отсутствие возможности создания функций с помощью пользовательского интерфейса;
- необходимость поддержки окружения;
- отсутствие автоматического масштабирования.

Однако эти недостатки не являются критичными и могут быть исправлены в будущем

Также был проведен анализ бессерверных архитектур. Бессерверные приложения, несмотря на запутанное название, это стиль архитектуры в котором вы меньшей степени полагаются на запуск серверных как части приложений, чем обычно. Это достигается с помощью FaaS и BaaS сервисов. BaaS позволяет тесно интегрировать сторонние сервисы в клиентскую часть разрабатываемых приложений. FaaS же позволяет перенести серверный код из долгоживущих компонентов в небольшие изолированные функции.

Бессерверный подход не будет верным решением для каждого конкретного случая. Он не заменит все существующие архитектуры. FaaS несет в себе как достоинства – автомасштабирование и упрощение развертывания, так и недостатки – тестирование, мониторинг.

Однако данные недостатки не перешивают достоинства данного подхода. Снижение затрат на эксплуатацию, поддержку и разработку, упрощение развертывания и воздействия на окружающую среду. Важнейшим аспектом является сокращение цикла обратной связи при создании новых приложений и компонентов приложений. Быстрый вывод приложения в эксплуатацию и получение отзывов от конечных пользователей позволяет компаниям и отдельным разработчикам быстрее выводить их продукты на рынок, а также быстрее вносить изменения в уже существующие продукты.

Бессерверные технологии находятся в зачаточном состоянии на данный момент. В ближайшие годы предвидится множество достижений в данной области и существующий архитектурный инструментарий может претерпеть изменения.

СПИСОК ПУБЛИКАЦИЙ

Намакарский, А.А. Разработка FaaS-платформы: Тезисы докл. к 53-я научно-техническая конфернция аспирантов, магистрантов и студентов БГУИР