УДК 82-5+80(042.5)

СОВРЕМЕННАЯ ОБЛАЧНАЯ ИНФРАСТРУКТУРА: БЕССЕРВЕРНЫЕ ВЫЧИСЛЕНИЯ

Макосий Алексей Иванович,
кандидат физико-математических наук, доцент кафедры программного обеспечения вычислительной техники и автоматизированных систем
Инженерно-технологический институт Хакасского государственного университета им. Н. Ф. Катанова (г. Абакан).
Макосий Роман,
DevOps-инженер
Фирма Base2Services (г. Мельбурн, Австралия)

Статья посвящена одному из новых и весьма популярных направлений в области облачных вычислений, известных под названием «бессерверные вычисления». Представлены такие области использования бессерверных вычислений, как автоматическое масштабирование веб-сайтов и API, обработка потока событий, манипуляция изображениями и видео. Авторами описываются основные компоненты FaaS: функции, события и ресурсы. Отмечены наиболее важные преимущества и недостатки технологии бессерверных вычислений.

Ключевые слова: *облачные вычисления, бессерверные вычисления, области использования, преимущества, недостатки.*

MODERN CLOUD INFRASTRUCTURE: SERVERLESS COMPUTING

Makosiy Aleksey Ivanovich,
Sc. D., Associate Professor, the Computer Facilities Software and Automated Systems Department,
Engineering and Technology Institute, Katanov Khakass State University (Abakan).
Makosiy Roman,
DevOps-engineer,
Base2Services Pty, Ltd (Melbourne, Australia)

The article is devoted to one of the new and very popular directions in the field of cloud computing, known as serverless computing. Such fields of serverless computing are presented as automatic scaling of websites and APIs, processing of the events flow, manipulation of images and videos. The authors describe such basic components of FaaS as functions, events, and resources and mark the most important advantages and disadvantages of the serverless computing technology.

Key words: *cloud computing, serverless computing, fields of use, advantages, disadvantages.*

В мире IT существуют различные уровни абстрагирования инфраструктуры, такие как прямой доступ к аппаратным ресурсам, работа с виртуальными машинами или с контейнерами. Для решения некоторых задач удобнее всего полностью абстрагировать функции администрирования с той целью, чтобы заказчик мог выполнять свой код, не беспокоясь об инфраструктуре и управлении серверами. Такова суть «бессерверных вычислений» [1].

Бессерверные вычисления позволяют сосредоточиться на бизнес-логике, не беспокоясь о проблемах, связанных с инфраструктурой работы приложений, что существенно упрощает управление серверами в облаке. Бессерверные вычисления можно считать последним шагом в отделении сервисов от виртуальной или физической инфраструктуры.

Можно выделить две определяющие характеристики бессерверных технологий: невидимая инфраструктура вместо настраиваемых образов виртуальных машин и схема оплаты, основанная на фактически потребляемых ресурсах вместо фиксированной почасовой ставки. Вторая характеристика не нова, так как большинство облачных сервисов уже работают по такому принципу.

Пионером в области бессерверных вычислений является Amazon Web Services. В 2014 году AWS выпустила новаторскую услугу под названием AWS Lambda, в которой предложила новый способ развёртывания в облаке веб-

приложений, мобильных приложений или приложений IoT. Вместо того, чтобы сразу развёртывать всю кодовую базу, Lambda позволяет развернуть функции приложения по отдельности в своих собственных контейнерах. Microsoft ответила, выпустив «Azure Function», а Google в феврале 2016 года создал «Cloud Function» как свою собственную службу вычислений без сервера.

Основные области использования бессерверных вычислений. Среди наиболее популярных сфер применения данной технологии можно назвать:

– ***автоматическое масштабирование вебсайтов и API***. Бессерверные веб-сайты и приложения могут быть написаны и развёрнуты без предварительной настройки инфраструктуры. Наилучшая часть в этом заключается в том, что backends автоматически масштабируются в зависимости от спроса. Всплески трафика обслуживаются без проблем со скоростью и доступностью;

– ***обработка потока событий***. Бессерверные вычисления могут запускаться как следствие какого-либо события, предоставляя эластичные, масштабируемые потоки событий без обслуживания сложных кластеров. Эти потоки событий могут обслуживать аналитические системы, обновлять вторичные хранилища данных и кэши, системы мониторинга;

– ***манипуляция изображениями и видео***. Бессерверные вычисления позволяют создавать улучшенные изображения и видеоуслуги для приложений, а также динамическое изменение размера изображений или изменение транскодирования видео для разных устройств. Приложения также всё чаще полагаются на такие вещи, как распознавание изображений, чтобы улучшить пользовательский интерфейс;

– ***IoT***. Большинство IoT-систем обычно полагаются на общие шаблоны в том, как они обрабатывают и хранят данные. Во-первых, этим системам необходимо принимать данные от различных датчиков, расположенных в разных местах. Затем эти системы обрабатывают и анализируют потоковые данные для получения информации в режиме реального времени. Бессерверные вычисления предлагают необходимую гибкость и отказоустойчивость при минимальных затратах на инфраструктуру.

Архитектурные принципы бессерверных вычислений. Хребтом бессерверных вычислений является функция как услуга (FaaS) – один из видов облачных вычислений, обеспечивающий платформу, которая позволяет клиентам разрабатывать, запускать и управлять функциональными возможностями приложений, практически избегая сложностей построения и поддержки инфраструктуры. Разработчик пишет функции, фрагменты приложения и задаёт правила, основанные на событиях, которые вызывают код в нужные моменты [2].

Основными компонентами FaaS являются ***функции***, ***события*** и ***ресурсы***. Функции – это независимые единицы развёртывания, которые могут выполнять разные действия, например, сохранение пользователя в базе данных, обработку файлов, выполнение запланированных задач и т. п. Всё, что запускает выполнение функции, рассматривается как событие, в частности, загрузка файла на файловый сервер, публикация сообщений и т. п. Ресурс относится к инфраструктуре или компонентам, используемым функциями. Например, службы базы данных (для сохранения данных пользователей/сообщений/комментариев), службы файловой системы (для сохранения изображений или файлов), службы очереди сообщений (для публикации сообщений).

Очень важным моментом во всём этом является то, что программное обеспечение, работающее в такой среде, должно этой среде соответствовать. Одного только FaaS недостаточно для создания полностью функционального приложения. Приложения часто требуют работы с базами данных, службами

аутентификации, очередями и другими. Большинство этих сервисов доступны в данной среде облачных провайдеров и могут быть использованы FaaS напрямую через SDK.

Эти услуги практически не требуют обслуживания. Например, DynamoDB – база данных класса NoSQL от Amazon в качестве службы хранения данных позволяет разработчикам приобретать услугу, основанную на пропускной способности, а не на объёме хранения. Если включено автоматическое масштабирование, то база данных будет автоматически масштабироваться. Кроме того, администраторы могут запрашивать изменения пропускной способности, а DynamoDB будет распространять данные и трафик по нескольким серверам, что позволяет прогнозировать производительность. Другие услуги также легко доступны для FaaS.

Функции аутентификации, работы с БД и ряд других могут перекладываться с серверов приложений и баз данных через специальное API стороннему провайдеру – так называемой Backend-as-a-Service (BaaS) платформе (во всех этих XaaS аббревиатурах скоро уже можно будет запутаться).

BaaS-платформами пользуется множество разработчиков игр, социальных и корпоративных приложений. Но рынок здесь ещё очень незрелый. Ранее флагманом движения BaaS считалась компания Parse. Она была основана в 2011 году и предоставляла решение для удалённой работы с данными (обработка и хранение), которое упрощало написание серверной бизнес-логики на JS. В 2016 году Facebook купил стартап, но прекратил обслуживание клиентов через год. Это вызвало лёгкую панику среди разработчиков и позволило войти в эту среду новым игрокам, например, российской компании Prof-ItVentures с её продуктом Scorocode. Указывая на молодость рынка, можно также отметить, что сегодня большинство применений FaaS пока что представляют собой обычные приложения, завёрнутые в новую среду.

Для облегчения перехода на новую технологию на рынке появились бессерверные фреймворки, которые предоставляют CLI для упрощения процесса создания и развёртывания бессерверного приложения и шаблоны, построенные над FaaS, с тем чтобы дать простой способ создания и настройки бессерверных приложений на многих языках, включая Python, Node.js, Java и некоторые другие.

Использовать работающие приложения с бессерверными фреймворками нецелесообразно. Чаще всего эти фреймворки используются для создания новых приложений либо для добавления функций или дополнительных услуг в работающие приложения. Наиболее известными фреймворками являются Serverless.com, APEX и AWS SAM.

Serverless.com – проект с открытым исходным кодом, поддерживающий пока только AWS Lambda с планами поддержки других крупных поставщиков облачных вычислений в будущем. Serverless.com использует YAML для описания функций, событий и связанных ресурсов. Во время развёртывания платформа Serverless преобразует весь синтаксис, описанный на YAML, в один шаблон AWS Cloud Formation.

APEX использует JSON для задания правил поведения и различных аспектов работы приложения без сервера. В APEX можно отметить поддержку языков, которые не поддерживаются Lambda, например, Golang.

AWS SAM является стандартной моделью приложений для приложений без сервера, размещённых в AWS Lambda. AWS SAM использует шаблон Cloud Formation для развёртывания приложения в виде стека Cloud Formation и определяет набор объектов, которые могут быть включены в шаблон Cloud Formation для простого описания общих компонентов бессерверных приложений.

Шаблоны описывают бессерверное приложение в соответствии с текстовыми файлами AWS SAM JSON или YAML и вводят несколько новых ресурсов и типов свойств, которые могут быть встроены в раздел ресурсов в шаблоне. Шаблоны могут включать все другие разделы шаблонов и использовать встроенные функции Cloud Formation для доступа к свойствам, доступным только во время выполнения.

Преимущества и недостатки бессерверных вычислений. Отметим некоторые важные преимущества и недостатки новой технологии. Главное преимущество бессерверной архитектуры состоит в том, что она позволяет сосредоточиться только на функционале приложения, а не на обслуживании инфраструктуры и её параметрах, таких как масштабируемость, высокая доступность и т. п.

Но размышления о вопросах поддержки инфраструктуры в актуальном состоянии и взаимодействии этого процесса с установленными приложениями идут вразрез с природой бессерверных вычислений. Обслуживание зависимостей среды – это не то, чем должен заниматься бессерверный разработчик.

Очевидным преимуществом является сокращение объёма инвестиций, необходимых для поддержки работы приложений. Обратной стороной можно назвать повышенную сложность их разработки и тестирования. Бессерверная архитектура требует структурного разделения одного приложения на сервисы и функции, что значительно увеличивает время разработки. Точно так же сложно выполнение интеграционных тестов и тестирование нагрузки для платформы Serverless, поскольку существует зависимость от внешних систем при запуске сценариев тестирования.

В качестве положительной характеристики можно также отметить снижение сложности установки и развёртывания приложений, разработанных для платформы Serverless. Процедуры компиляции и загрузки намного проще по сравнению с традиционной упаковкой и развёртыванием.

В настоящее время многие разработчики и администраторы впечатлены идеями и подходами бессерверных вычислений. Но маловероятно, что бессерверный подход станет панацеей и вряд ли заменит все существующие подходы и программные архитектуры. Однако можно с уверенностью сказать, что эта область будет развиваться и двигать вперёд сферу разработки программного обеспечения и облачных вычислений.

Библиографический список

1. Gyanendra Rai, Prashant Pasricha, Santosh Pandey, etc. Serverless Architecture: Evolution of a New Paradigm. – URL: https://www.globallogic.com/gl_news/serverless-architecture-evolution-of-a-new-paradigm (дата обращения: 17.09.18).
2. Mike Roberts. Serverless Architectures. – URL: https://martinfowler.com/articles/serverless.html (дата обращения: 17.09.18).