УДК 004.051

СЕРВИС AWS LAMBDA

А.П. ГАЙДЕЛЬ
(*Представлено: канд. физ.-мат. наук, доц. О.Н. ПЕТРОВИЧ*)

В данной статье рассматриваются вычислительная платформа AWS Lambda — ключевой компонент бессерверных вычислений – ее плюсы и минусы, возможности и предоставляемые сервисы.

Введение. Все чаще люди переходят на AWS Lambda ради масштабируемости, производительности, экономии и возможности обрабатывать миллионы запросов в месяц. Для этого не нужно управлять инфраструктурой, на которой работает сервис. А автоматическое масштабирование позволяет обслуживать тысячи одновременных запросов в секунду.

Основной раздел. AWS Lambda — это событийно-ориентированный сервис бессерверных вычислений, который позволяет выполнять код без выделения и администрирования серверов и дополнять другие сервисы AWS на основе пользовательской логики. Lambda автоматически реагирует на различные события (так называемые триггеры), например, на HTTP-запросы через Amazon API Gateway, изменение данных в корзинах Amazon S3 или таблицах Amazon DynamoDB; либо можно запустить свой код через вызовы API, используя AWS SDK и переходы между состояниями в AWS Step Functions.

Lambda выполняет код на высокодоступной вычислительной инфраструктуре и полностью отвечает за администрирование нижележащей платформы, включая обслуживание серверов и операционной системы, выделение ресурсов, автоматическое масштабирование, мониторинг кода и ведение журналов. То есть вам достаточно загрузить свой код и настроить, как и когда он должен выполняться. В свою очередь, сервис позаботится о его запуске и обеспечит высокую доступность вашего приложения.

AWS Lambda — это удобная вычислительная платформа, подходящая для решения множества задач, разумеется, если язык и среда выполнения вашего кода поддерживаются сервисом. Если вы хотите сосредоточиться на коде и бизнес-логике, поручив обслуживание серверов, выделение ресурсов и масштабирование стороннему поставщику за разумные деньги, вам точно стоит перейти на AWS Lambda.

Lambda идеально подходит для создания программных интерфейсов, а если использовать сервис вместе с API Gateway, можно значительно сократить расходы и быстрее выйти на рынок. Есть различные способы использования функций Lambda и варианты организации бессерверной архитектуры — каждый сможет выбрать что-то подходящее с учетом поставленной цели.

Lambda позволяет выполнять широкий спектр задач. Так, благодаря поддержке CloudWatch можно создавать отложенные задания и автоматизировать отдельные процессы. Нет никаких ограничений по характеру и интенсивности использования сервиса (учитываются расход памяти и время), и вам ничто не мешает планомерно работать над полноценным микросервисом на основе Lambda.

Здесь можно создавать сервис-ориентированные действия, которые не выполняются постоянно. Типичный пример — масштабирование изображений. Даже в случае распределенных систем функции Lambda не теряют своей актуальности.

Как и большинство сервисов AWS, Lambda предоставляется по принципу общей ответственности AWS и клиента по части безопасности и соблюдения нормативных требований. Этот принцип снижает операционную нагрузку на клиента, поскольку AWS берет на себя задачи обслуживания, администрирования и контроля компонентов сервиса — от операционной системы хоста и уровня виртуализации до физической безопасности объектов инфраструктуры.

Если говорить конкретно об AWS Lambda, то AWS отвечает за управление нижележащей инфраструктурой, связанными базовыми сервисами, операционной системой и платформой приложений. В то время как клиент несет ответственность за безопасность своего кода, хранение конфиденциальных данных, контроль доступа к ним, а также к сервису и ресурсам Lambda (Identity and Access Management, IAM), в том числе в пределах используемых функций.

Основное преимущество Lambda заключается в том, что, выполняя функцию от вашего имени, сервис сам выделяет необходимые ресурсы. Вы можете не тратить время и силы на администрирование систем и сосредоточиться на бизнес-логике и написании кода.

Сервис Lambda разделен на две плоскости. Первая — плоскость управления. Согласно Википедии, плоскость управления (control plane) — это часть сети, отвечающая за транспортировку сигнального трафика и маршрутизацию. Она является главным компонентом, принимающим глобальные решения о выделении, обслуживании и распределении рабочих нагрузок. Кроме того, плоскость управления выступает в роли сетевой топологии поставщика решения, отвечающей за маршрутизацию трафика и управление им.

Вторая плоскость — плоскость данных. У нее, как и у плоскости управления, свои задачи. Плоскость управления предоставляет API для управления функциями (CreateFunction, UpdateFunctionCode) и контролирует взаимодействие Lambda с другими сервисами AWS. Плоскость данных управляет API вызовов (Invoke API), который запускает функции Lambda. После вызова функции плоскость управления выделяет либо выбирает существующую, заранее подготовленную для этой функции среду выполнения, а затем выполняет в ней код.

AWS Lambda поддерживает множество языков программирования, включая Java 8, Python 3.7, Go, NodeJS 8, .NET Core 2 и другие, через соответствующие среды выполнения. AWS регулярно их обновляет, распространяет исправления безопасности и выполняет прочие операции по обслуживанию этих сред. Lambda позволяет использовать и другие языки при условии, что вы сами внедрите соответствующую среду выполнения. И тогда уже вам придется заниматься ее обслуживанием, в том числе следить за безопасностью.

Каждая функция работает в одной или нескольких выделенных средах, которые существуют лишь в течение жизненного цикла этой функции, а затем уничтожаются. В каждой среде одновременно выполняется лишь один вызов, но она используется повторно, если возникает множество серийных вызовов одной и той же функции. Все среды выполнения работают на виртуальных машинах с аппаратной виртуализацией — на так называемых microVM. Каждая microVM назначается конкретной учетной записи AWS и может многократно использоваться средами для выполнения различных функций в этой учетной записи. MicroVM упаковываются в структурные блоки аппаратной платформы Lambda Worker, которой владеет и управляет AWS. Одна и та же среда выполнения не может использоваться разными функциями, равно как microVM уникальны для разных учетных записей AWS (рис. 1).

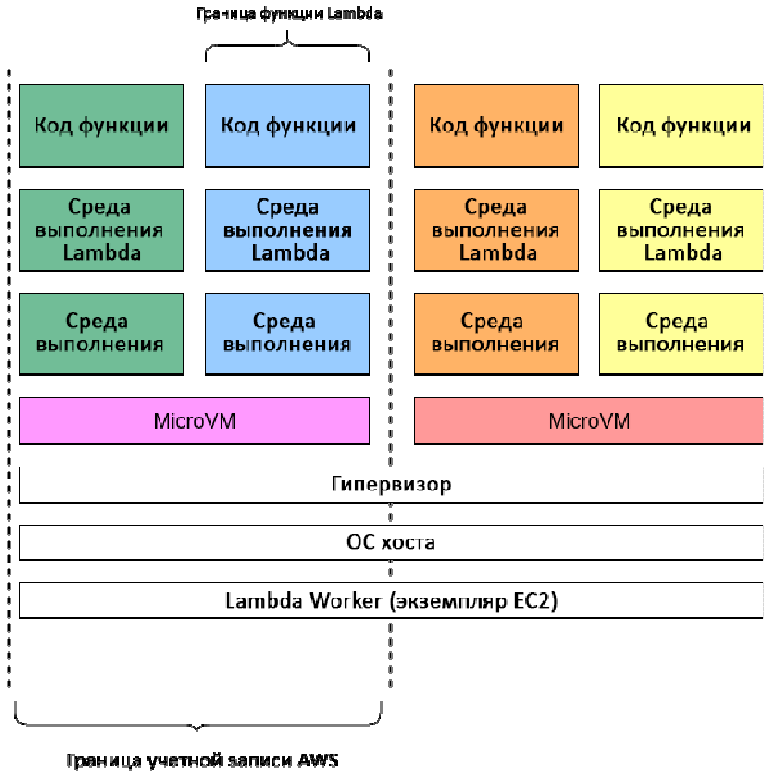

Рисунок 1. – Модель изоляции в AWS Lambda

Изоляция сред выполнения реализована с помощью нескольких механизмов. На высшем уровне каждой среды присутствуют отдельные копии следующих компонентов:

– Код функции
– Любые слои Lambda, выбранные для функции
– Среда выполнения функции
– Минимальное пользовательское пространство на базе Amazon Linux

Для изоляции разных сред выполнения применяются следующие механизмы:

– cgroups — ограничение доступа к ресурсам ЦП, памяти, пропускной способности накопителей и сети для каждой среды выполнения;

– namespaces — объединение в группы ID процессов, ID пользователей, сетевых интерфейсов и других ресурсов, которыми управляет ядро Linux. Каждая среда выполнения работает в своем пространстве имен;

– seccomp-bpf — ограничение системных вызовов, которые можно использовать в среде выполнения;

– iptables и routing tables — изоляция сред выполнения между собой;

– chroot — предоставление ограниченного доступа к нижележащей файловой системе.

В сочетании с проприетарными технологиями изоляции AWS перечисленные механизмы гарантируют надежное разграничение сред выполнения. Изолированные таким образом среды не могут обращаться к данным других сред и изменять их.

Хотя несколько сред выполнения одной учетной записи AWS могут выполняться на одной microVM, ни при каких обстоятельствах microVM не могут совместно использоваться разными учетными записями AWS. Для изоляции microVM в AWS Lambda используется всего два механизма: экземпляры EC2 и Firecracker. Изоляция гостей (guest isolation) в Lambda на основе экземпляров EC2 применяется с 2015 года. Firecracker — это новый гипервизор с открытым исходным кодом, специально разработанный AWS для бессерверных рабочих нагрузок и представленный в 2018 году. Физическое оборудование, на котором выполняются microVM, совместно используется рабочими нагрузками разных учетных записей.

Хотя среды выполнения Lambda уникальны для разных функций, в них можно повторно вызывать одну и ту же функцию, то есть среда выполнения может просуществовать несколько часов, прежде чем будет уничтожена.

В каждой среде выполнения Lambda также имеется файловая система с разрешением на запись, доступная через каталог /tmp. К его содержимому нельзя обращаться из других сред выполнения. Что касается сохранения состояний процессов, записанные в /tmp файлы существуют в течение всего жизненного цикла среды выполнения. За счет этого возможно аккумулирование результатов нескольких вызовов, что особенно полезно для таких затратных операций, как загрузка моделей машинного обучения.

Интерфейс Invoke API можно задействовать в двух режимах: в режиме событий и в режиме «запрос — ответ». В режиме событий вызов добавляется в очередь для последующего выполнения. В режиме «запрос — ответ» функция вызывается мгновенно с предоставленной полезной нагрузкой, после чего возвращается ответ. И в том и в другом случае функция выполняется в среде Lambda, но с различными путями полезной нагрузки.

Во время вызовов типа «запрос — ответ» полезная нагрузка поступает от API обработки запросов (API Caller), такого как AWS API Gateway или AWS SDK, в балансировщик нагрузки, а затем — в службу выполнения вызовов Lambda (Invoke Service). Последняя определяет подходящую среду для выполнения функции и передает туда полезную нагрузку, чтобы завершить вызов. Балансировщик нагрузки получает трафик с TLS-защитой через Интернет. Трафик в пределах сервиса Lambda — после балансировщика нагрузки — проходит через внутренний VPC в определенном регионе AWS.

Вызовы по событиям могут выполняться незамедлительно либо добавляться в очередь. В некоторых случаях очередь реализована с помощью сервиса Amazon SQS (Amazon Simple Queue Service), который передает вызовы в службу выполнения вызовов Lambda посредством внутреннего опрашивающего процесса (poller). Передаваемый трафик защищен TLS, при этом какое-либо дополнительное шифрование данных, хранящихся в Amazon SQS, не предусмотрено.

Вызовы по событиям не возвращают ответы — любую ответную информацию Lambda Worker попросту игнорирует. Вызовы на основе событий Amazon S3, Amazon SNS, CloudWatch и других источников обрабатываются сервисом Lambda в режиме событий. Вызовы из потоков Amazon Kinesis и DynamoDB, вызовы очередей SQS, балансировщика нагрузки приложений и API Gateway обрабатываются в режиме «запрос — ответ».

Вы можете производить мониторинг и аудит функций Lambda с помощью различных механизмов и сервисов AWS, включая следующие:

– Amazon CloudWatch — собирает различные статистические данные, такие как количество запросов, продолжительность выполнения запросов и число запросов, завершившихся ошибкой.

– Amazon CloudTrail — позволяет вести журналы, непрерывный мониторинг и сохранять сведения об активности в учетной записи, связанные с вашей инфраструктурой AWS. У вас будет полная хронология действий, выполненных с помощью консоли AWS Management Console, AWS SDK, инструментов командной строки и других сервисов AWS.

– AWS X-Ray — обеспечивает полную видимость всех этапов обработки запросов в вашем приложении на основе карты его внутренних компонентов. Позволяет анализировать приложения в ходе разработки и в производственной среде.

– AWS Config — вы сможете отслеживать изменения конфигурации функций Lambda (включая их удаление) и сред выполнения, тегов, имен обработчиков, размера кода, распределения памяти, настроек времени ожидания и параметров параллелизма, а также роли выполнения Lambda IAM, подсети и привязки групп безопасности.

Заключение. Событийно-ориентированная архитектура — это не просто ускорение или улучшение существующих бизнес-процессов, это другой бизнес. Раньше невозможно было реализовать такую архитектуру, не было фундамента в виде облачных платформ, но теперь это реальное технологическое и стратегическое преимущество.

ЛИТЕРАТУРА

1. Blog.kr [Электронный ресурс] AWS LAMBDA. Событийно-ориентированная архитектура. – Режим доступа: https://blog.kr.pp.ru/post/2018-12-24/. – Дата доступа: 10.09.19.
2. Wikipedia [Электронный ресурс] Событийно-ориентированное программирование. – Режим доступа: https://ru.wikipedia.org/wiki/Событийно-ориентированное_программирование/. – Дата доступа: 20.09.19.
3. Aws.amazon [Электронный ресурс] AWS Lambda. – Режим доступа: https://aws.amazon.com/ru/lambda/. – Дата доступа: 21.09.19.