

了解Serverless架构

1 概述

Serverless中文译为“无服务”是一种新兴起的架构模式，公司ESB产品引入Rest微服务服务机制过程，笔者刚好参与其中，其中Serverless作为一个新起的概念，跟微服务架构相关，为此笔者查阅了许多资料，有一些初步了解，因此把自己的学习笔记分享给大家，希望能对相关学习者有所帮助。本文主要是从自身的疑问出发，来阐述什么是Serverless架构，Serverless架构有哪些优缺点以及有哪些应用案例等。

2 名词解释

什么是Serverless？它是一种基于互联网的技术架构理念。Serverless不是真的不需要服务器了，而是不用过多的关注服务器。它是指明显或充分地依赖第三方应用或服务来管理服务器端逻辑和状态，可以让你在服务部署级别而不是服务器部署级别来管理你的应用部署，甚至可以让你管理某个具体功能或端口的部署，这就能让开发者快速迭代，更快速地开发软件。

Serverless架构分为两种类型，分别为“Backend as a Service”和“Functions as a Service”。

“Backend as a Service”即BaaS，是一种新型的云服务，指在为移动和Web应用提供后端云服务，实现对逻辑和状态进行管理，包括云端数据/文件存储、消息推送（例如极光推送、个推）、应用数据分析等等。可以说BaaS是诞生于移动互联网，为了加速移动应用开发和降低成本而形成的开发架构。

“Functions as a Service”即FaaS，指这样的应用，一部分服务逻辑由应用实现，但是跟传统架构不同在于，他们运行于无状态的容器中，可以由事件触发，短暂的，完全被第三方管理，功能上FaaS就是不需要关心后台服务器或者应用服务，只需关心自己的代码即可。其中AWS Lambda是目前最佳的FaaS实现之一。

Lambda是一种计算服务，它在AWS基础设施上执行用JavaScript（node.js）、Python或Java编写的代码。源代码部署到孤立的容器上，该容器有单独分配的内存、磁盘空间和处理器。代码、配置和依赖项这一组合通常被称为Lambda函数。

3 架构特点

3.1 节约成本

我们在阿里云或者腾讯云上购买的云服务器的配置是固定的，比如2G内存，双核CPU，这样做的弊端是如果系统在特定的某一天需要支持很高的并发量，比如：双十一那天，但是服务器的内存并不够，难道要为了那一天又特定去买一些内存么？在Serverless架构下，用户能够通过网络、硬盘、CPU等计算资源，在不需要额外服务器基础设施的情况下，可以做到随时扩缩容，对数据库的存储也没有限制。它是按照调用次数进行计费的，如果平时没有访问量就不计费，所以不会有浪费的情况，有效的节约成本。

3.2 简化运维

传统服务器需要维护程序和硬件设施；而Serverless架构中，开发人员面对的将是第三方开发或自定义的API和URL，底层硬件对于开发人员是透明化的，技术团队无需再关注运维工作，能够更加专注于应用系统开发。同时，应用程序的组成逻辑会使用大量的第三方功能服务。目前，例如登陆鉴权的服务，云数据库服务等第三方服务在安全性、可用性、性能方面都进行了大量优化，开发团队直接集成第三方的服务，就能够有效的降低开发成本，同时使得应用的运维过程变得更加清晰，有效的提升了应用的可维护性。

4 应用模式

4.1 UI-驱动型应用

让我们从服务端逻辑上来探讨下，一个3层的客户端导向系统。常见的电子商务应用是一个好的例子（这里我拿一个网上宠物店来当例子）

传统上，架构将看起来像这样，并且假设它在服务器端的Java中实现，使用HTML / Javascript组件作为客户端：



1 概述

2 名词解释

3 架构特点

3.1 节约成本

3.2 简化运维

4 应用模式

4.1 UI-驱动型应用

4.2 消息驱动应用

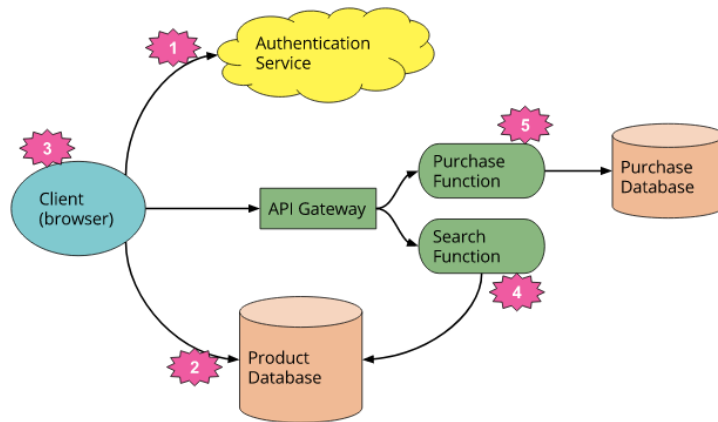
5 架构缺陷

5.1 厂商平台绑定

客户端(浏览器) ---> 宠物店服务器 ---->数据库

使用这种架构，客户端可能相对不智能，系统中的许多逻辑 - 认证，页面导航，搜索，事务 - 由服务器应用程序实现。

使用无服务器架构，可能会更像是这样：



认证服务--->产品数据库--->客户端(浏览器) --->API网关--->购买功能--->购买数据库

认证服务--->产品数据库--->客户端(浏览器) --->API网关--->搜索功能--->产品数据库

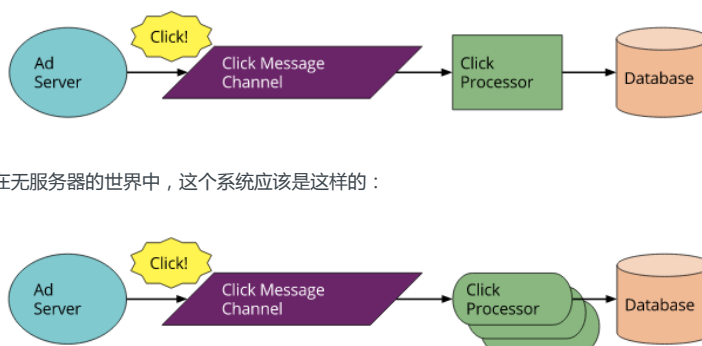
下面是两者区别：

1. 将原来应用的认证逻辑系统删除，替换成一个第三方的BaaS服务。
2. 允许客户端直接访问数据库，比如产品列表，数据库在第三方主机上比如AWS Dynamo，这样，我们访问数据库的安全策略就和访问服务器资源不同。
3. 前面两点意味着非常重要的第三点，原来在宠物店的逻辑现在迁移到客户端了，比如跟踪用户会话，理解应用的UX用户体验结构，比如分页，从数据库中读取和转为可用的视图等，客户端其实在这时已经变成了一个单页应用。
4. 一些UX相关功能可能会保留在服务器端，比如计算敏感或需要访问大量数据，比如搜索功能，对于这种功能我们不总是让其运行在服务器端，而是实现一个FaaS函数方式来响应http请求，客户端通过API网关来访问这个FaaS函数。
5. 使用FaaS函数来替代购买功能，因为安全原因还是让其放在服务器，不需要在客户端再实现一遍，这也是通过API网关调用实现的。

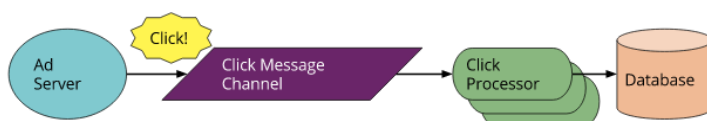
4.2 消息驱动应用

另一个不同的例子是一个后端数据计算服务。比如你正在写一个用户为中心的应用，需要快速响应UI请求，但是你又需要捕获发生的所有不同类型的活动。让我们来思考下在线广告系统，当一个用户点击一个广告时，你需要快速的导向到广告，但是同时你又将这个点击计数，从而向广告商收费。

传统方式下这种系统的架构可能是类似这样的：“广告服务器”会以同步的方式响应用户（由于只是例子，我们并不需要关心具体的交互），同时需要向渠道发布一条消息并由负责更新数据库的“点击处理器”应用程序以异步的方式处理，例如扣掉广告主的部分预算。



在无服务器的世界中，这个系统应该是这样的：



和上一个例子相比，本例中两种架构的差异很小。我们将需要长期运行的消费者应用程序替换为一个在供应商提供的事件驱动的上下文中运行的FaaS函数。请注意该供应商同时提供了消息代理（Message Broker）和FaaS环境，这两个系统非常紧密地相互结合在一起。

通过实例化（Instantiating）函数代码的多个副本，FaaS环境可以并行处理多个点击，这主要取决于最初流程的编写方式，同时这也是一个需要考虑的新概念。

5 架构缺陷

5.1 厂商平台绑定

平台会提供Serverless架构给大玩家，比如AWS Lambda，运行它需要使用AWS指定的服务，比如API网关，DynamoDB，S3等等，一旦你在这些服务上开发一个复杂系统，你会粘牢AWS，以后只好任由他们涨价定价或者下架等操作，个性化需求很难满足，不能进行随意的迁移或者迁移的成本比较大，同时不可避免带来一些损失。Baas行业内一个比较典型的事件，2016年1月19日Facebook关闭曾经花巨额资金收购的Parse，造成用户不得不迁移在这个平台中产生一年多的数据，无疑需要花费比较大的人力和时间成本。

5.2 没有行业标准




目前的情况也只适合简单的应用开发，缺乏大型成功案例的推动。对于Serverless缺乏统一的认知以及相应的标准，无法适应所有的云平台，目前尚不成熟，各厂家自说自话，更多是在一种探索、观望过程中，真正在做的可能并没有提标准，而提标准的有不少还在概念模糊阶段。

6 心得总结

Serverless无服务器架构是开发人员和企业组织需要考虑、研究和采用的最新理念，它是依赖第三方应用或服务来管理服务器端逻辑和状态的技术架构，但是其实它并不能替代服务器。Serverless是最新流行微服务的一种表现形式，是新一代云服务和开发架构的实践，是云计算发展重点方向之一。

随着开发人员积极采用AWS Lambda等计算服务，这种架构可能会迅速发展起来。但是，这一架构目前来看还不是特别成熟，有待考证。笔者认为我们可以熟悉下业内Serverless架构的经典产品，并进行学习，进而开发属于自己的Serverless产品，数通畅联的拳头产品ESB企业服务总线中就提供Rest微服务服务组件、以及微服务服务管理、监控、统计功能，能满足微服务架构下的常见典型应用场景，如有需求敬请关注。

相关博客

<div><div>[云框架]FaaS & Serverless架构 / [Cloud Frameworks]Function as a Service & Serverless Architectures</div><div>一只饭盒</div><div>200</div></div>	<div><div>基于 Docker 的 Serverless 架构实践 —— UCloud 通用计算产品的实现及其应</div><div>OSC源创君</div><div>30180</div></div>	<div><div>Serverless</div><div>全栈路</div><div>180</div></div>
---	--	---