Serverless Architectures

Server的架构一般分为几种呢?

- **Single Application**

  server部署在一起，非常笨重，变得越来越难以维护和迭代

- **Microservices**

  server分为很多的微服务，按照业务领域或者业务规则来进行划分

- **Functions**

  和singleApplication&微服务不一样，不是一种一直在运行的进程，而是在秒级内启动并执行

云服务架构的演变

- **XaaS**

  这是一个通用的...aaS的解决方案的叫法，比如以后可能会有Game as a Service

- **DBaaS**

  数据库即服务, 这是一个托管数据库的平台，可提供备份，集群和高可用性。比如 Amazon DynamicDB

- **IaaS**

  基础架构即服务，是所有XaaS的最低级别。它为我们提供了强大的功能，但需要大量的配置。IaaS提供了我们必须维护的虚拟机。IaaS与拥有物理服务器机房的区别在于，不必购买任何物理计算机，就可以在世界各地配置服务器。但是，与其他XaaS相比，IaaS的维护比较困难，并且需要一名出色的DevOps工程师来配置虚拟机以使其高效，安全地工作

  是很多国内外大型云计算服务商提供的最基本的服务

- **PaaS**

平台即服务，是指给用户提供的能力是使用由云服务提供商支持的编程语言、库、服务以及开发工具来创建、开发应用程序并部署在相关的基础设施上。用户无需管理底层的基础设施，包括网络、服务器，操作系统或者存储。用户一般只关心某些配置参数即可

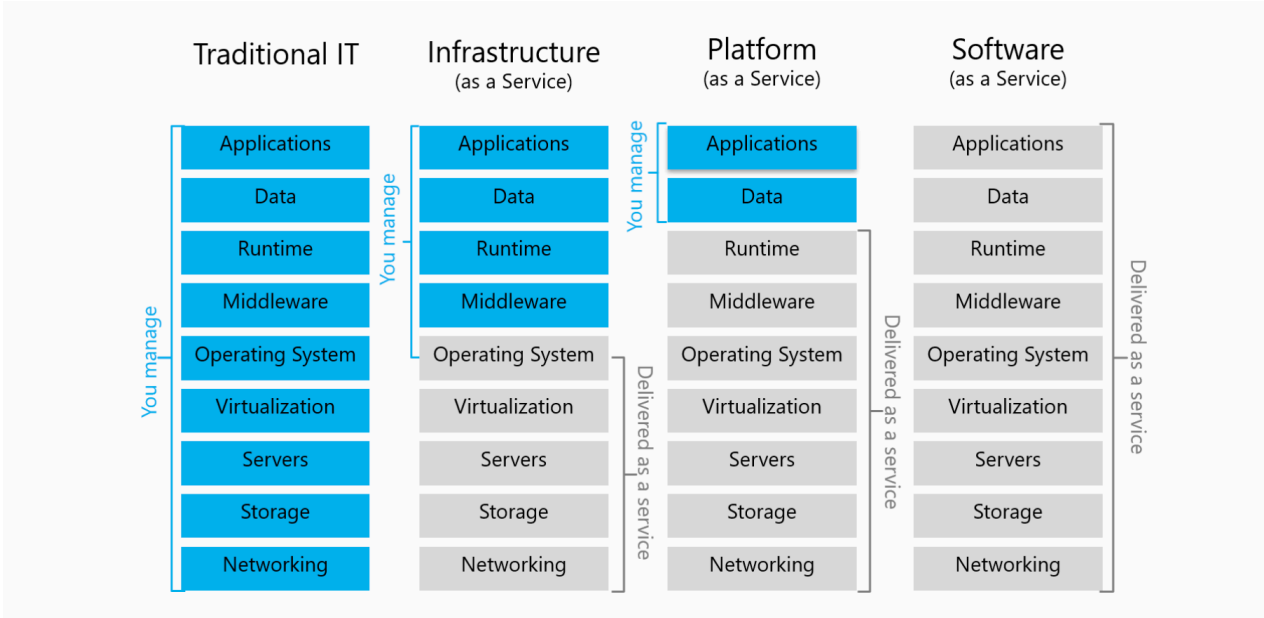
也是很多国内外大型云计算服务商提供的最基本的服务

- SaaS

  软件即服务，是指厂商通过提供特定的应用程序给用户，例如Jira，GitLab, 网盘等

  常见的厂商提供的云产品如下，IaaS是最基础的服务，PaaS是中间层，SaaS是具体的应用服务

按照美国国家标准和技术研究院的划分，云服务就是上面的这三类服务，我们看下从传统的架构到SaaS的范围，可以看出到SaaS的标准基本都是定制化的产品出现，整个的Application全部被托管



- **DaaS**

  数据即服务类似于SaaS，甚至可以将其视为SaaS的子集。通常可以是一个API，它返回一些数据。货币汇率，天气查询，体育比赛结果等。流行的DaaS是Google Maps，Google Translate API

  公共api库

  spotify api

- **CaaS**

  容器即服务，是基于容器的虚拟化的一种形式，其中容器引擎，业务流程和基础计算资源作为服务从云提供商交付给用户

- **BaaS**

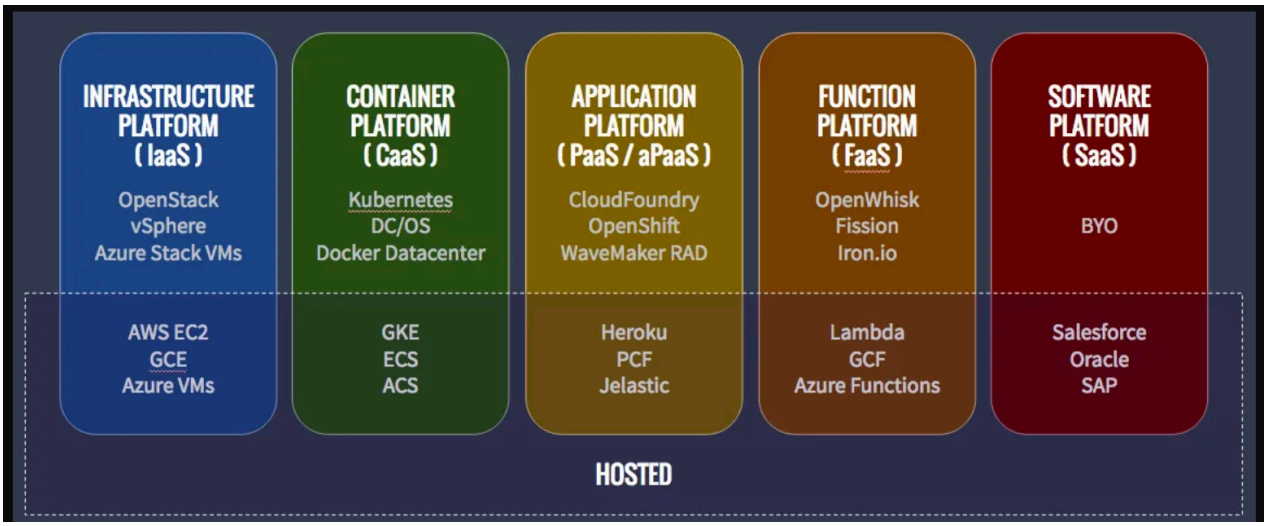
  后端即服务，它更像是用于应用程序开发而不是业务处理的SaaS。比如通用的身份验证系统（注册，登录整个流程）

- **FaaS**

  功能即服务，比PaaS更简单，顾名思义，它基于可以由给定事件触发的功能，因此它是基于事件的体系结构。开发人员只需编写一个函数，而不必考虑诸如部署，服务器资源，可伸缩性等主题。这是因为FaaS是可自动伸缩的。

  AWS Lambda， Google Cloud Functions，Microsoft Azure Functions

  整个云服务可以看作是朝着这个方向发展 IaaS –> CaaS –> PaaS –> FaaS –> SaaS



什么是Serverless

首先澄清一点并不能按照字面意思来理解为无服务器，而是说对应用开发者而言，不再需要操心大部分跟服务器相关的事务，比如服务器选购、应用运行环境配置、负载均衡、日志搜集、系统监控等，这些事情统统交给Serverless平台即可，应用开发者唯一需要做的就是编写应用代码，实现业务逻辑。

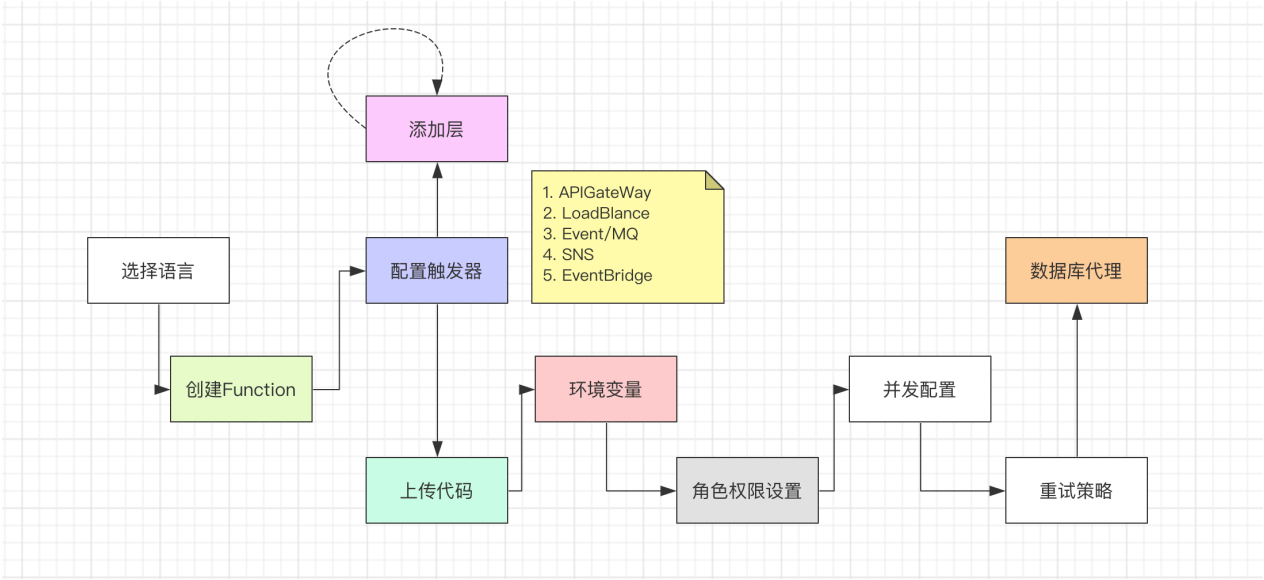
Serverless == BaaS U FaaS

Serverless可以从两个角度来看待，一个是BaaS，另一个是FaaS

- Backend as a Service, 这里的Backend可以指代任何第三方提供的应用和服务(比如数据库服务，身份验证服务)
- Functions as a Service, 应用以函数的形式存在，并由第三方云平台托管运行，比如AWS Lambda，腾讯云

工作流程

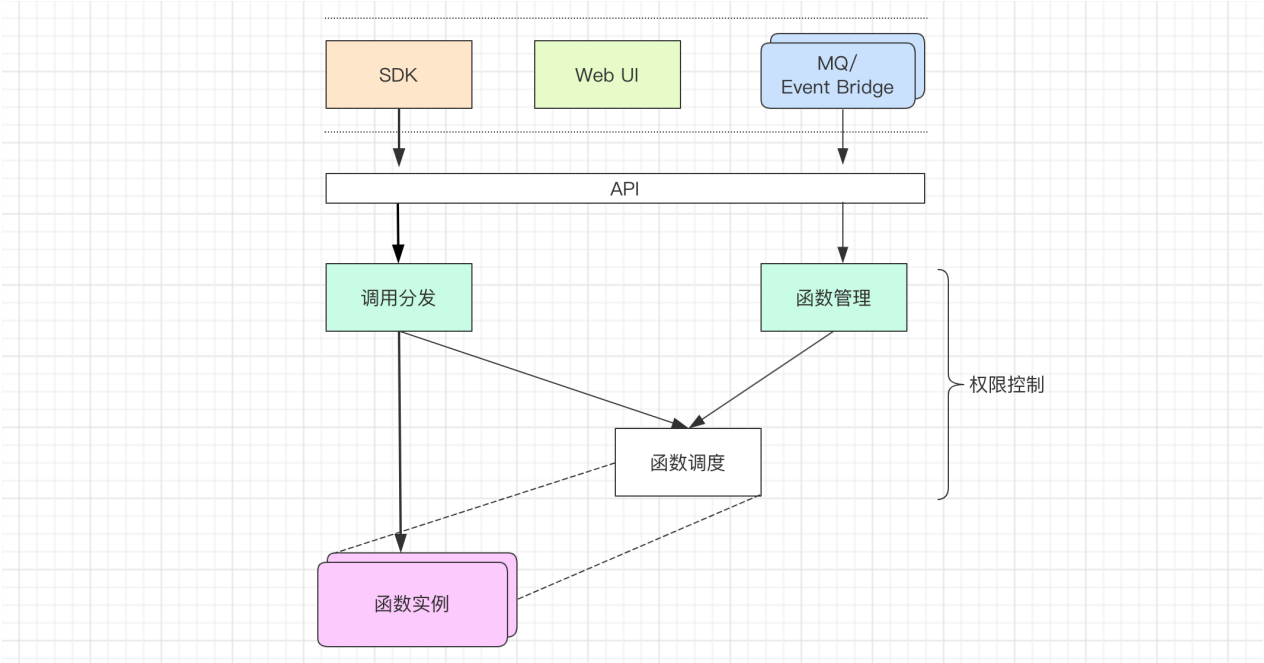
指定Function name -> 添加触发器(主要流程, 包含各种触发器) -> 上传代码(zip/jar) -> 配置环境变量 -> 配置并发 -> 异步调用 -> 数据库代理



架构

FaaS为用户提供SDK/WEBUI两种使用方式，并通过事件注册与回调机制与其它云组件打通，提供标准的API接口。调用分发根据函数所属的区域，用户，名字，版本号，鉴权等信息申请函数实例，并将调用均匀的分发到可用函数实例。函数管理负责创建/修改/删除函数，并提供函数代码管理，版本管理等功能。函数调度根据函数资源需求选择合适的位置创建/销毁函数实例。函数实例部署用户定义的函数，负责函数的执行及监管。

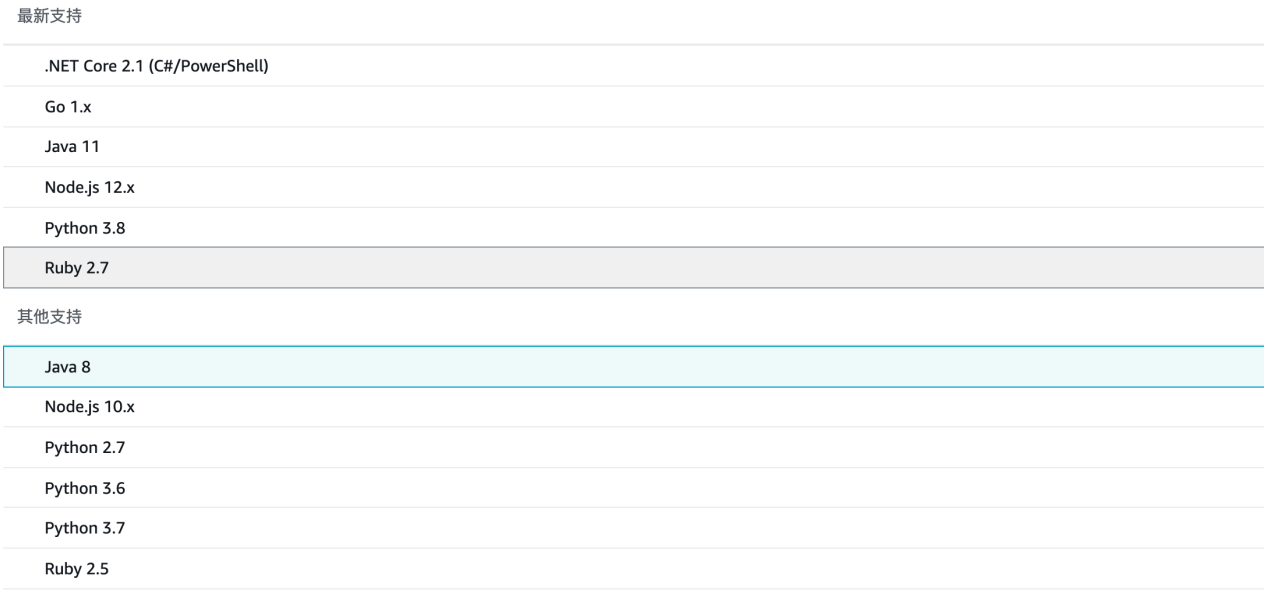
FaaS平台也支持函数的版本进行调用和部署



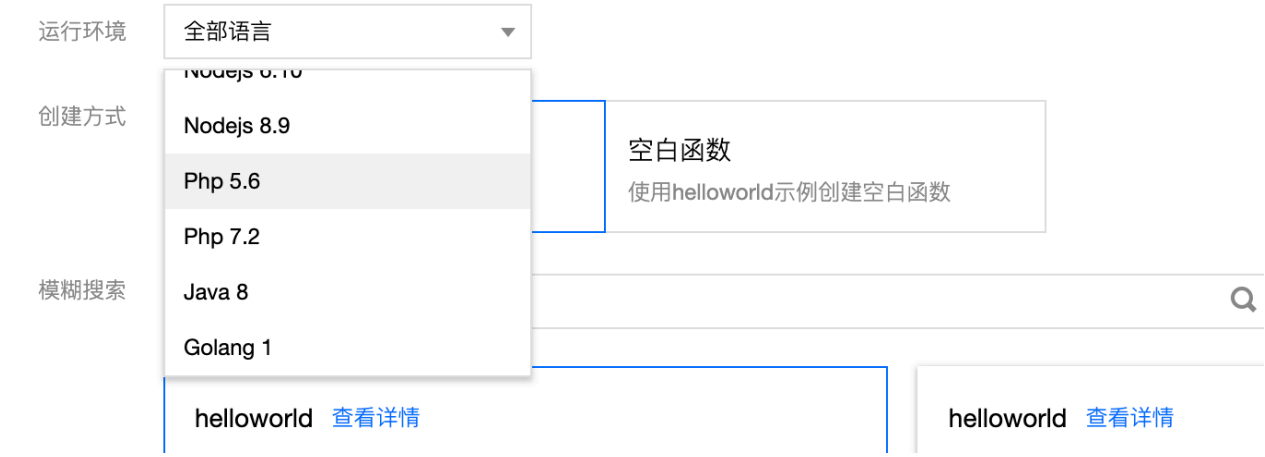
Sample

支持的语言和环境

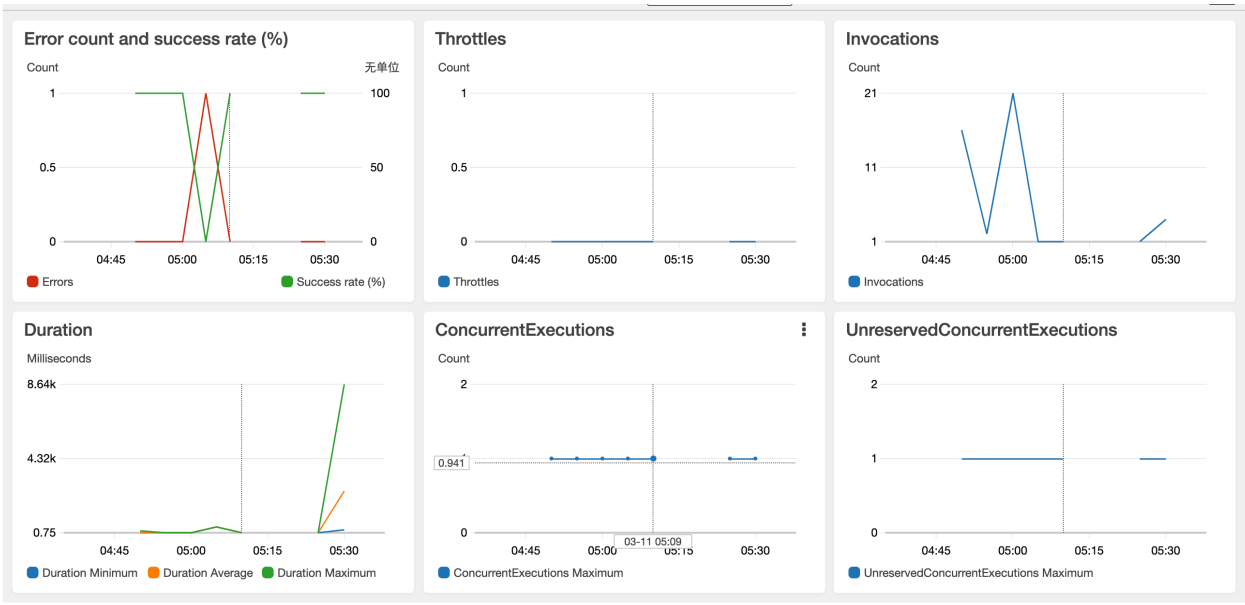
AWS



腾讯云



监控



调用日志

Recent invocations

#	Timestamp	RequestID	LogStream	DurationInMS	BilledDurationInMS	MemorySetInMB	MemoryUsed
1	2020-03-11T05:38:21.949Z	74538638-c206-47cf-97a4-ff081c30c57c	2020/03/11/[$LATEST]3872f5fdcb5c45288d39ebe4ce641c07	148.7	200	512	85
2	2020-03-11T05:02:45.093Z	25ba48f1-be0a-4938-9952-0f4ead53c993	2020/03/11/[$LATEST]ff722ce0d31d4f39bebaa89ac5467a59	1.01	100	512	86
3	2020-03-11T05:02:44.687Z	ff334134-65e9-4070-94af-96a1c03d3bc0	2020/03/11/[$LATEST]ff722ce0d31d4f39bebaa89ac5467a59	0.9	100	512	86
4	2020-03-11T05:02:44.146Z	7ccad644-f9f3-4006-9c1a-7a718c9a3a0d	2020/03/11/[$LATEST]ff722ce0d31d4f39bebaa89ac5467a59	0.98	100	512	86
5	2020-03-11T05:02:43.645Z	ac34f9fe-9fef-47d3-b0c2-a581f56ae3e0	2020/03/11/[$LATEST]ff722ce0d31d4f39bebaa89ac5467a59	1.1	100	512	86
6	2020-03-11T05:00:52.771Z	56f008d9-8e9e-47e2-93a0-18be6565035d	2020/03/11/[$LATEST]ff722ce0d31d4f39bebaa89ac5467a59	0.87	100	512	86
7	2020-03-11T05:00:51.568Z	c42d702c-f45d-4863-872c-62ea3a701974	2020/03/11/[$LATEST]ff722ce0d31d4f39bebaa89ac5467a59	0.89	100	512	86
8	2020-03-11T05:00:50.305Z	b584db10-5a87-424d-a073-48ddc4a9b7bf	2020/03/11/[$LATEST]ff722ce0d31d4f39bebaa89ac5467a59	0.96	100	512	86
9	2020-03-11T05:00:44.969Z	85671f3e-56ba-4a47-95a6-d3500d1703c8	2020/03/11/[$LATEST]ff722ce0d31d4f39bebaa89ac5467a59	1.12	100	512	86

Most expensive invocations in GB-seconds (memory assigned * billed duration)

#	Timestamp	RequestID	LogStream	BilledDurationInMS	MemorySetInMB	BilledDurationInGBSeconds
1	2020-03-11T04:50:34.496Z	b7f5e530-9021-44c9-abdc-35dba9f4f83c	2020/03/11/[$LATEST]ff722ce0d31d4f39bebaa89ac5467a59	200	512	0.1
2	2020-03-11T05:38:21.949Z	74538638-c206-47cf-97a4-ff081c30c57c	2020/03/11/[$LATEST]3872f5fdcb5c45288d39ebe4ce641c07	200	512	0.1
3	2020-03-11T05:00:52.771Z	56f008d9-8e9e-47e2-93a0-18be6565035d	2020/03/11/[$LATEST]ff722ce0d31d4f39bebaa89ac5467a59	100	512	0.05
4	2020-03-11T05:02:43.645Z	ac34f9fe-9fef-47d3-b0c2-a581f56ae3e0	2020/03/11/[$LATEST]ff722ce0d31d4f39bebaa89ac5467a59	100	512	0.05
5	2020-03-11T05:02:44.146Z	7ccad644-f9f3-4006-9c1a-7a718c9a3a0d	2020/03/11/[$LATEST]ff722ce0d31d4f39bebaa89ac5467a59	100	512	0.05
6	2020-03-11T05:02:44.687Z	ff334134-65e9-4070-94af-96a1c03d3bc0	2020/03/11/[$LATEST]ff722ce0d31d4f39bebaa89ac5467a59	100	512	0.05
7	2020-03-11T05:02:45.093Z	25ba48f1-be0a-4938-9952-0f4ead53c993	2020/03/11/[$LATEST]ff722ce0d31d4f39bebaa89ac5467a59	100	512	0.05
8	2020-03-11T04:56:31.977Z	cd66cac4-4d53-4a8d-962c-32d8b62d8565	2020/03/11/[$LATEST]ff722ce0d31d4f39bebaa89ac5467a59	100	512	0.05
9	2020-03-11T04:56:34.977Z	1cdb8c79-9279-4a10-bcdc-efc0f6d348a5	2020/03/11/[$LATEST]ff722ce0d31d4f39bebaa89ac5467a59	100	512	0.05

容器化相比

Docker/Kubernetes

FaaS的缩放是自动的，并且是透明且细粒度的。容器平台仍然需要开发者管理集群的大小和形状。可以把容器服务囊括在底层，统一打包成对外的服务。基于Kubernetes的FaaS平台

拆箱即用，功能即服务

不需要关注机器资源，以及应用程序的创建

不需要特定的框架或者包依赖

自动的水平缩放

系统级别的上下文事件总线

启动延迟和冷启动

- **语言/代码量/库数量**
- **处理事件的能力关乎于事件handler的启动，AWS会释放不活跃的handler**

是否适用?

这里有很多条件可以供参考，取决于应用程序的流量分布

- **无服务器函数最适合运营成本居高不下，但并不必要的场合。开发者通常会借助这种函数开发新应用，而非将现有应用重建为无服务器模式。**
- **无服务器函数通常能让需要大量计算的应用获得最显著的成本收益，但无法保证通过采用无服务器架构就能在计算方面节约更多成本。**
- **可将代码连接至需要捆绑在一起的服务。目前的无服务器产品虽然能提供很多便利，但往往是专有的，会导致与其他相关云产品的紧密“锁定”。**
- **可以在概念证实阶段多花一些时间，看看无服务器技术是否真的最适合你从事的项目。**

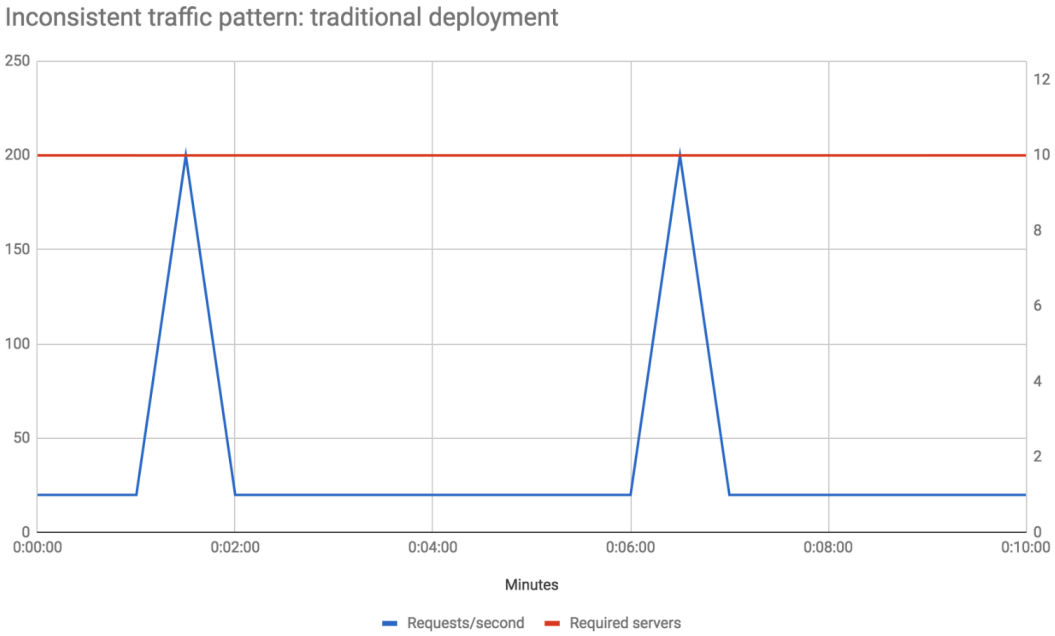
好处

- 降低运营成本（基础设施/规模经济效应Serverless经济架构论文），某些场景IaaS/PaaS也可以提供

Table 1: Comparing hostings price for one hour of operation, assuming 200 ms of runtime, executing every five minutes.

Service instance	Billable unit	Unit cost (USD)	Fail-over costs (%)	Cost of 12 x 200ms exec'ns	% reference price
Lambda (128 MB)	100 ms	$0.000000208	included	$0.000004992	24.94%
Lambda (512 MB)	100 ms	$0.000000834	included	$0.000020016	100.00%
Heroku Hobby (512 MB)	1 month	$7.00	100%	$0.0097222222	48572.25%
AWS EC2 t2.nano (512 MB)	1 hour	$0.0059	100%	$0.0118	58952.84%
AppEngine B1 (128MB)	1 hour	$0.05	100%	$0.1	499600.32%
AppEngine B4 (512MB)	1 hour	$0.20	100%	$0.4	1998401.28%

- 扩展成本 （偶尔请求/流量分布不一致），优化代码是节约资源的根本



- 减化流程管理
  - 收益可以扩展到基础架构成本之外，举个例子，A -> B -> C(底层服务). 底层服务的容量仍然需要手动动态的扩容增加
  - 减少打包和部署复杂性
  - 持续产品实验（借助于全局的MQ消息）
- 事件驱动，上下游资源的解耦合，响应完请求后立即释放资源，不需要进程常驻
- 无状态，函数可以任意进行迁移和重用

缺点

- 供应商控制（宕机/成本变化/功能丧失/强制模版）

- 多租户控制，权限控制
- 供应商锁定，很难轻易换供应商（需要统一标准，目前还在推行标准比如Knative）
- 安全问题，多租户问题，厂商监控敏感数据和权限下放的问题
- 无状态
    - 函数不是常驻内存的，响应完请求资源会被释放，对编译型语言不友好（JVM的JIT编译，逃逸分析优化手动不再有意义）
    - 启动优化，目前许多厂商在做优化，主要是通过磁盘的多级优化以及对容器化的镜像优化
    - 请求完函数进程被kill, 使用本机内存的优化手段不再有意义，下一次的请求分配拿不到本机的缓存。函数都使用机外缓存，如Redis和DB，增大了网络IO的开销
- 迁移成本以及学习成本

业界尝试

Netflix抽象了通用的API平台组件给Client工程师使用

阿里巴巴对于通用业务中间件进行扩展

给我们的启示

- Serverless需要找到合适的业务场景来体现价值，没必要完全切换
- 通用化的业务场景可以找到利用的场景
    - 通用的push服务
    - 图片上传压缩
    - CI/CD
    - ...

- **Serverless是以另一种架构的角度来看待业务**
- **自研 VS 加入?**
- **GDPR规范**

相关参考链接

- https://rajeevkuruganti.com/cloud-iaas-paas-saas-and-more/
- https://www.einfochips.com/blog/serverless-architecture-the-future-of-software-architecture/
- https://martinfowler.com/articles/serverless.html
- https://mp.weixin.qq.com/s?__biz=MzA5OTAyNzQ2OA%3D%3D&chksm=88931c6cbfe4957a702e66221e1bf997c4ba5a66de279294b08cccadd3ff5d6cabf103657484&idx=1&mid=2649694991&mpshare=1&scene=23&sn=818dea0cb058a08ac6b66ee865204630&srcid=0907dIsFi2ho3ez9orBMGatf
- https://mp.weixin.qq.com/s/Gj_qPPTn6KN065qUu6e-mw