[TECHO] | TVP Tencent Cloud Valuable Professional intel

/serverless/DAYS

Techo × TVP 开发者峰会

/serverless/ DAYS China 2021

无服务器，大有未来

Serverless, Empower More

Serverless的服务治理

陈皓 @ MegaEase

MegaEase

自我介绍

- 20+年工作经验，超大型分布式系统基础架构研发和设计
- 擅长领域：金融、电子商务、云计算、大数据
- 职业背景
  - 阿里巴巴资深架构师（阿里云、天猫、淘宝）
  - 亚马逊高级研发经理（AWS、全球购、商品需求预测）
  - 汤森路透资深架构师（实时金融数据处理基础架构）
- 目前创业MegaEase，致力于为企业提供云原生技术架构产品
  - 支撑高可用高并发高性能的微服务分布式Cloud Native架构
  - 为40+公司提供过软件技术服务

陈皓

Weibo: @左耳朵耗子
Twitter: @haoel
Blog: coolshell.cn
Wechat: haoelx

目录

惨淡经营

- 2006年 – Zimiki 伦敦Fotango推出“Pay as you go”，商业上失败，2017年关闭。
- 2008年 – Google App Engine。仅限于Python。包括具有60秒超时的HTTP函数，以及具有自己的超时的Blob存储区和数据存储区。不允许内存中的持久性。〔后来更名为Google Cloud Platform〕
- 2010年 – PiCloud平台。旨在通过三个操作来简化云计算。Cloud.call在云上运行您的自定义函数，cloud.result检索函数的返回值，cloud.map将您的函数映射到多个参数。〔2013年被Dropbox收购，之后再也不做Serverless了〕
- 2011年 – dotCloud公司，Docker公司前身，最终也是以失败告终
- 国内的各种APP Engine – TAE, SAE …〔基本也是以失败告终〕

卷土重来

Serverless == FaaS ?

- 2014年 – AWS Lambda，函数式服务化计算模型。2015年发布API Gateway
- 2015年 – Kubernetes, 基于Docker容器的服务调度系统。
- 2016年 – Google Cloud Functions. IBM Cloud Function. Azure Functions…等
- 2017年 – Cloudflare Workers
- Knative, OpenFaaS, Kubeless, Fn, OpenLambda, IronFuncitons, Fission, Apache OpenWhisk

基础设施越来越完善

- 不必关心系统管理和基础计算/存储/网络资源
- 不必关心服务器的运维，容量和自动化伸缩
- 不必关心代码的部署、监控和日志
- 我们的 Code 可以很快地变成一个 Service

Serverless == FaaS ?

- 我们真的可以做到函数级的业务开发吗？
- 如果我们能做到，那么我们怎么管理和运维粒度如此小的巨多的服务呢？
- 试想，如果我们生产环境下的服务数据提高1-2个数量级，我们要怎么管理呢？
- 表面上简单的东西，后面肯定不简单！

你有没有想过这些问题？

Serverless 如何进行服务发现？

Serverless 如何进行健康检查？

Serverless 如何做灰度发布 或 A/B测试？

Serverless 需要监控哪些指标，以及调用链？

Serverless 相互间的依赖关系如何管理？

Serverless 的容错处理（重试、限流、降低、隔离、熔断）SLA如何保障 ？

Serverless is Dead

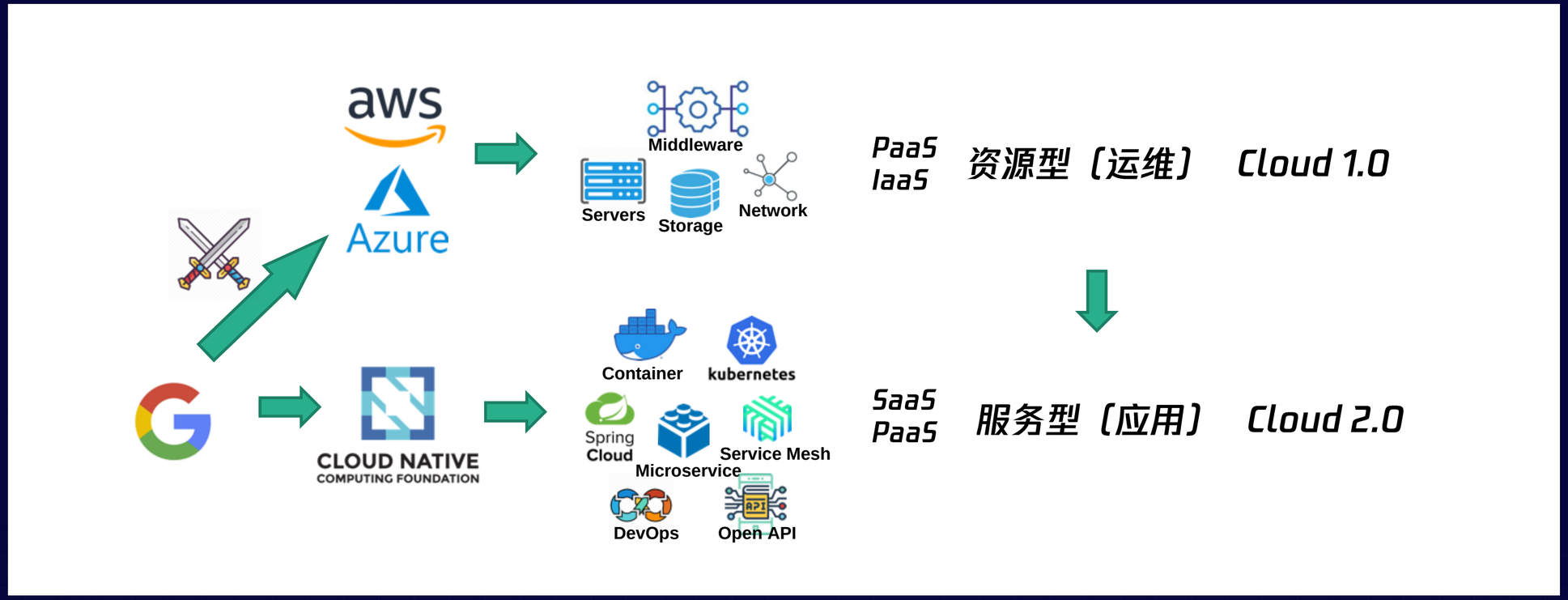
What we should instead be focusing on is what we're seeing to be the new way of doing modern application development

- Greatly reduced operational burden
- Tighter alignment to costs w/ usage
- Developers can/could/should be able to do almost anything
  - understated warning to Ops/DevOps/SRE/-ish folks
- Opinionated platforms that allow for multiple use-cases
- Opinionated platforms that bake in true best of breed practices, security, scale, performance, cost aspects for you

aws

https://www.slideshare.net/ChrisMunns/serverless-is-dead-132954772

“无服务器”并不是没有服务器，无服务器表示，应用程序的开发人员无需担心基础设施，如：容器，VM或物理服务器的容量规划，配置，管理，维护，操作或扩展。



应用服务是一等公民

对外 API 是重中之重

整体 SLA 是头等大事

是不是Serverless无所谓

我们主要关心的是 Service 而不是 Resource

- 有没有提升开发效率，可以更快地开发和上线
- 是不是可以有更高的性能和更好的稳定性，扩展性和安全性
- 有没有降低运维的成本
- 有没有很好地管理好成本和使用量
- 开发人员可以更容易很自然地融合到DevOps/SRE中来

Serverless需要的基本配套设施

所有的一切都需要为服务和应用SLA（开发和运维）服务

资源伸缩编排

通过Kubernetes和容器的
解决方案可以容易搞定

全栈可观测性

需要从应用层、中间件
层，资源层全面关联数据

服务治理

服务健康检查、容错设计
服务注册发现，配置管理

流量管理

流量编排调度，流量保护
限制，流量着色管控

可观测的重点

- 不是仅仅只是收集更多的监控数据，而是需要关联数据，数据不关联则没有意义。
- 需要找到这样的关联：从API → 服务 → 服务调用链 → 中间件 → 基础资源
- 需要解决的问题是
  - “急诊”- 快速故障定位
  - “体验”- SLA报告 + 容量分析
- 需要关联的数据有：
  - 基础资源的的Metrics（如：CPU，Memory, I/O，Network…）
  - 中间件的Metrics, Logs（如：JVM、Redis、MySQL、Kafka、Proxy/Gateway…）
  - 应用的Metrices, Logs（如：Access Log, Application log, Throughput, Latency, Error…）
  - API的调用链跟踪（需要穿过应用内的异步调用，消息中间件）

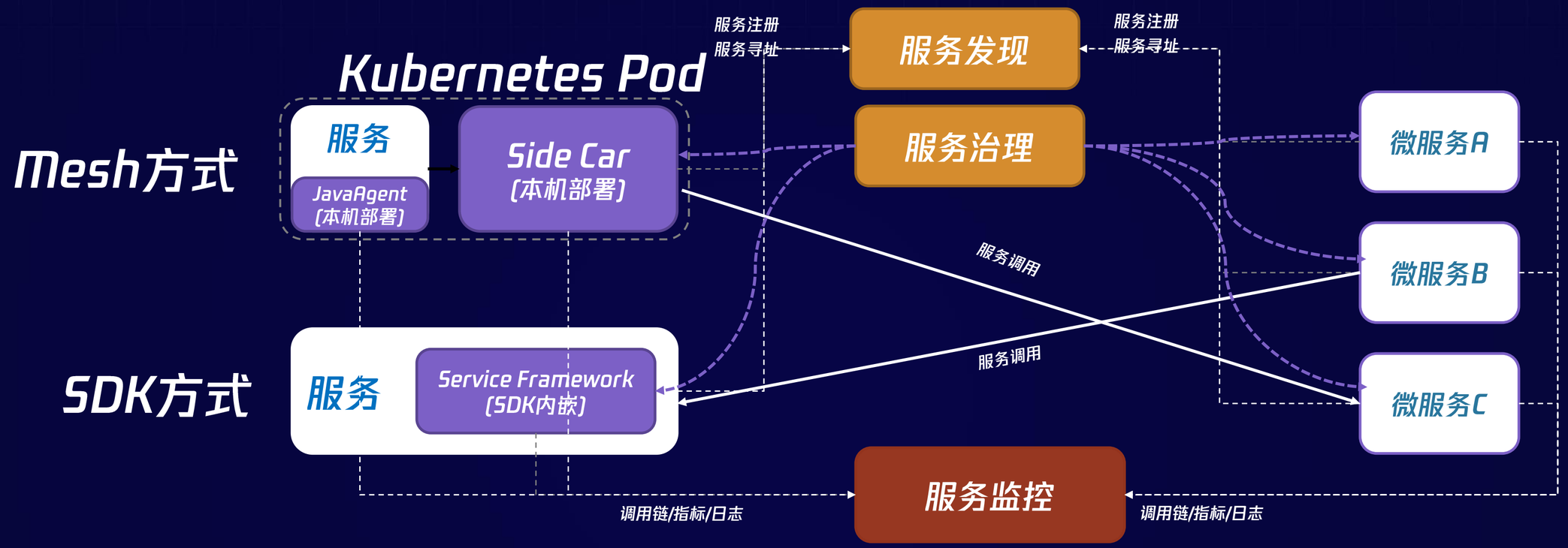
流量管理的重点

- 流量保护 – 按吞吐量限流，按响应时间限流。
- 流量着色 – 能够进行流量着色处理，以便后台服务可以进行流量调度（把不同的流量分配给不同的服务实例）
- 灰度发布 – 能够进行七层的流量规则路由
- 流量过滤 – 能够进行流量过滤，比如：协议校验、白名单管理、权限认证……
- 流量编排 – 可以进行 API 编排、聚合
- 流量降级 – 可以对请求进行降级（使用降级版本的API，通过API响应缓存降低请求……）
- 容错管理 – 对后端的服务进行负载均衡，熔断，重试等操作

服务治理的重点

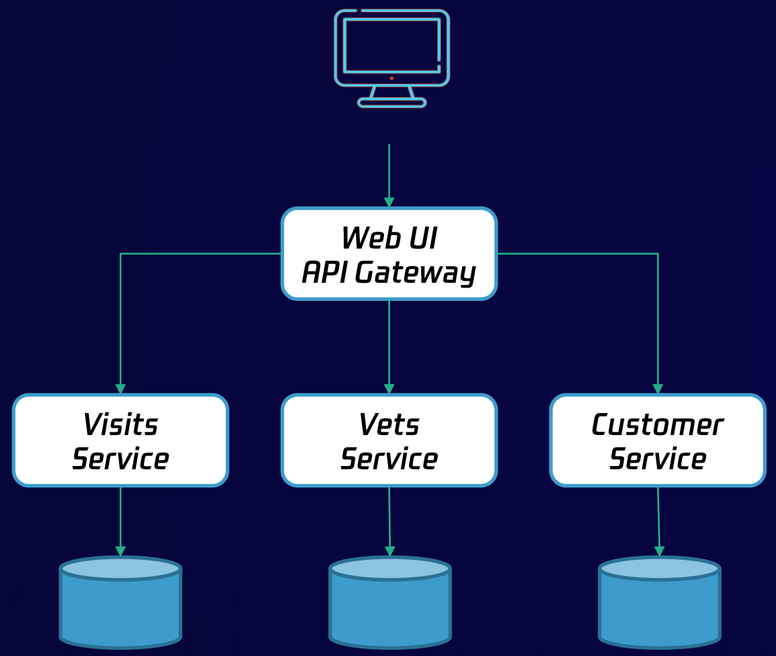
- 服务注册发现
  - Severless的服务启动后，如何进行注册和发现？
  - 支持灰度的多版管理怎么做？
  - 是用 K8s 的服务注册发现，还是用 Java 系统的服务注册发现？
- 服务配置管理
  - CMDB必需是服务为视角的，而不是资源为视角。
  - 为了支持灰度发布，服务配置也需要是多版本的
- 服务健康检查
  - 服务的健康检查应该怎么做？K8s的liveness 和 readiness
- 服务流量治理
  - 限流、重试、熔断……
- 服务的可观测性
  - 服务运行时的吞吐量（qps）、响应时间（p99, p90,p80...）、错误率、GC次数时长
  - 服务运行时的相关日志 Access log, Application log
  - 服务运行时的调用链跟踪日志 Tracing Log
  - 服务运行时的资源利用

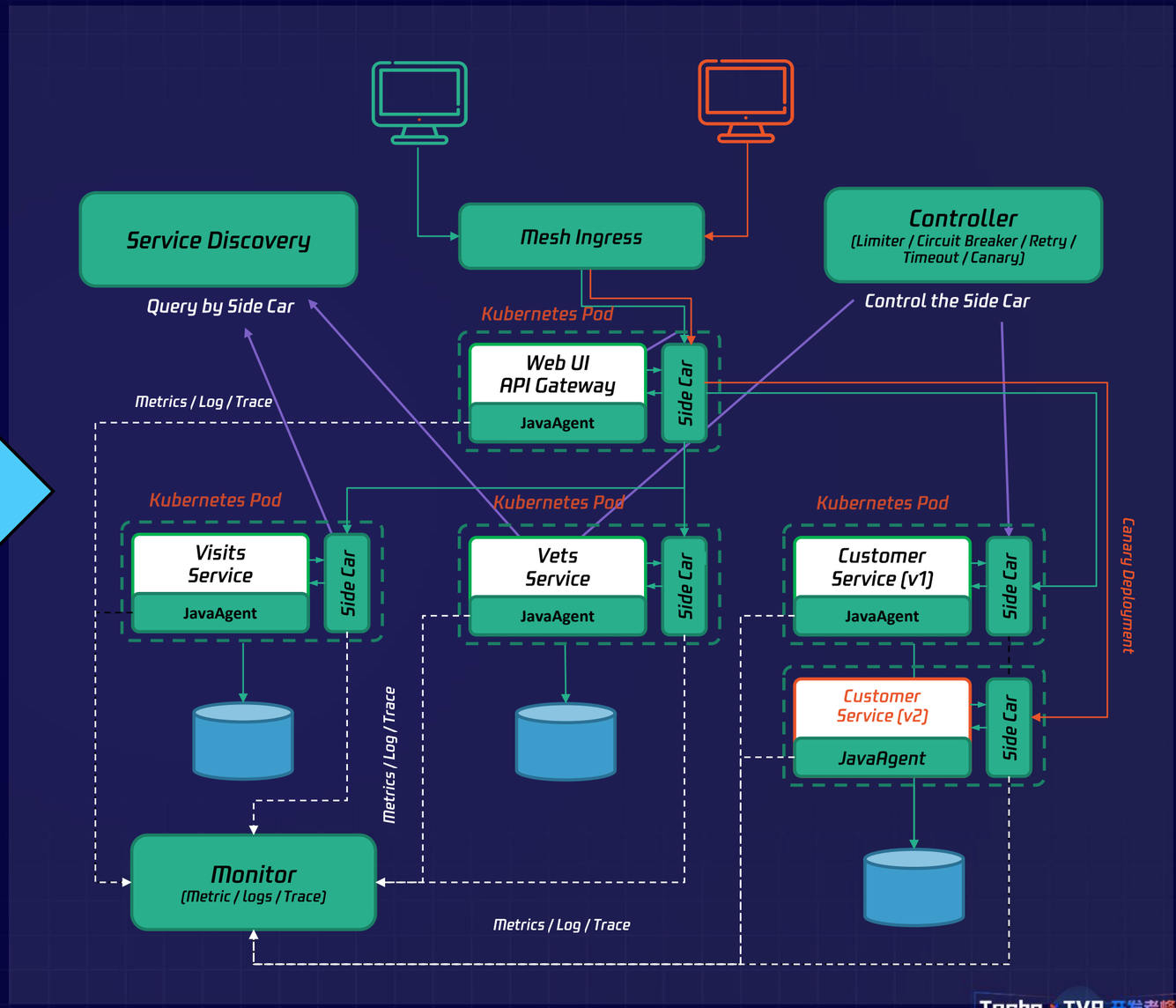
两种做法



- 通过在服务的机器上安装一个Side Car边车服务。把【服务A】对外的流量全部劫持到Side Car上。
- 并通过Java Agent的字节码注入技术，采集【服务A】的调用链和指标数据。
- 这样就可以把一个服务在完全不知情的情况下进行服务治理和监控。
- Mesh 这个方法对服务和应用来说完全透明。

Mesh的无侵入式方案





MegaEase

THANK YOU!

欢迎关注我们的开源项目

https://github.com/megaease