

微服务核心研究之--编排

沉落的星星 关注

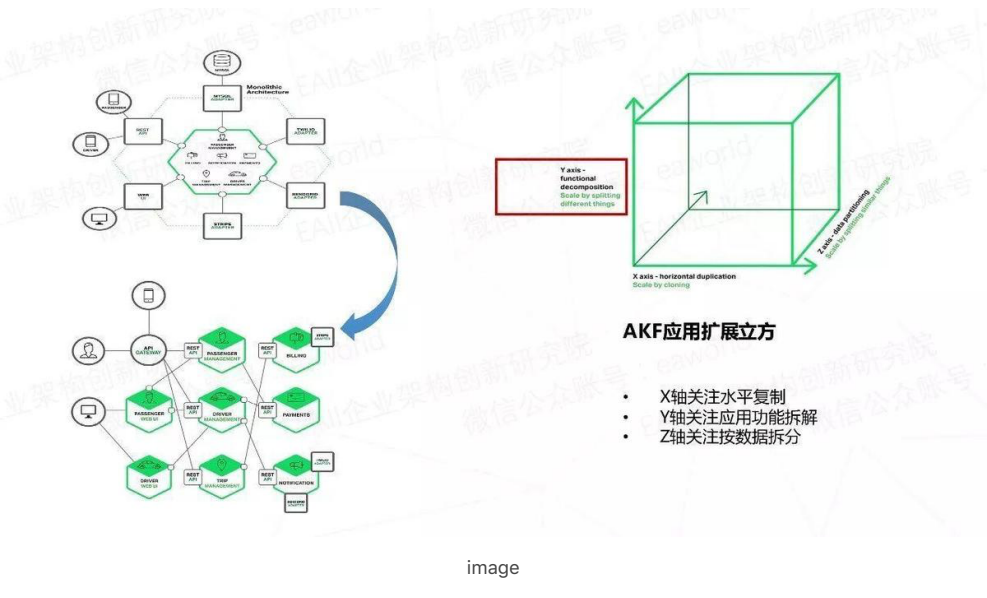
1 2019.07.18 13:39:22 字数 4,240 阅读 9,101

目录：

- 一、微服务编排的必要性
- 二：3种常见的微服务编排方式
 - 1、Orchestration（编制）
 - 2、Choreography（编排）
 - 3、API网关
- 三、微服务编排的框架（Orchestration方式）
 - 1、流程编排的思路
 - 2、流程编排的模型
 - 3、适配参数
 - 4、流水号
 - 5、调用链分析
- 四、微服务编排的事务一致性
- 五、微服务编排的监控工具支撑

一、微服务编排的必要性

微服务是目前流行的一种新兴的软件架构风格，在微服务体系结构中，可以将应用分解为多个更小颗粒度的服务, 各个服务可以由不同的团队并行独立开发、部署。



(图片来源：<https://www.nginx.com/blog/introduction-to-microservices/>)

以一个出租车调度软件为例，最开始是一个单体应用，应用核心是业务逻辑，由定义服务、域对象和事件的模块完成。尽管也是模块化逻辑，但是最终它还是会打包并部署为单体式应用。随着时间增加，功能逐渐增多，代码越来越多，这个软件就会越来越难维护。这时使用微服务

华为云

Q元试用


40+高配云产品

云主机 | 安全 | 数据库

企业0元体验



广告

沉落的星星 关注

总资产 13 (约1.26元)

从基因进化角度揭秘：人类为什么会衰老？为什么无法永生？(转载)

阅读 11

为什么要去IOE（转载）

阅读 32

电脑技巧：酷睿i3、i5、i7

阅读 246

架构就是不错的选择。一个微服务一般完成某个特定的功能，比如订单管理、客户管理等等。每一个微服务都有自己的业务逻辑和适配器。一些微服务还会发布API给其它微服务和应用客户端使用。其它微服务完成一个Web UI，运行时，每一个微服务实例可能是一个Docker容器。

《The Art of Scalability》(中文书名：架构即未来)一书介绍了一个应用横向扩展所需要遵守的AKF扩展模型。根据AKF扩展模型，横向扩展实际上包含了三个维度，而横向扩展解决方案则是这三个维度上所做工作的结合。V维度是业务复制，V维度是应用功能拆解，Z维度是控制数据拆

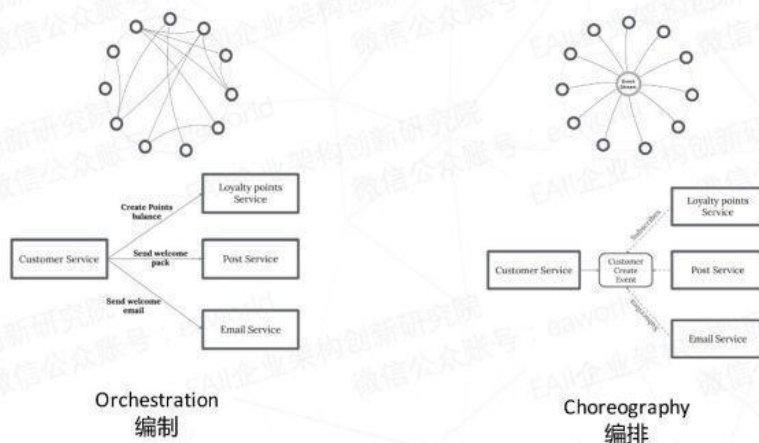
微服务核心研究之--编排



沉落的星星

关注

赞赏支持



image

当一个系统采用了微服务架构后，原有的业务可能并没有发生变化，但系统已被拆分成了很多新的微服务，与传统架构相比，微服务架构下会更依赖通过各微服务之间的协作来实现一个完整的业务流程，这种协作就是服务编排。编排涉及到RPC、分布式事务等，需要有完善的编排框架来支撑。

二：3种常见的微服务编排方式

目前有3种常见的微服务编排方式，实现微服务的组合与协调，可根据开发项目的实际情况进行选择。

1、Orchestration（编制）

Orchestration面向可执行的流程：通过一个可执行的流程来协同内部及外部的服务交互，通过流程来控制总体的目标、涉及的操作、服务调用顺序。

Orchestration和BPM、ESB的思想很相似，首先要有一个流程控制服务，该服务接收请求，依照业务逻辑规则，依次调用各个微服务，并最终完成处理逻辑。可以把控制服务视作BPM、ESB引擎，微服务视作BPM、ESB的各种组件。

Orchestration实现方案多是同步的。

优点：

流程控制服务时时刻刻都知道每一笔业务究竟进行到了什么地步，监控业务成了相对简单的事情。

缺点：

1) 流程控制服务很容易控制了太多的业务逻辑，耦合度过高，变得臃肿。

推荐阅读

三面字节跳动被虐得“体无完肤”，15天读完这份pdf，终拿下美团研发
阅读 56,090

对于二本渣渣来说，面试阿里P6也太难了！（两年crud经验，已拿
阅读 31,054

离开菜鸟&新的面试体验
阅读 6,729

聊聊skywalking的SamplingService
阅读 477

金星问许晴：50岁了还单身，你想要怎样的男人？许晴的回答很露骨
阅读 31,045

华为云

0元试用 40+高配云产品

云主机 | 安全 | 数据库

企业0元体验

广告

2) 各个微服务退化为单纯的增删改查，失去自身价值。

2、Choreography（编排）

Choreography面向协作：通过消息的交互序列来控制各个部分资源的交互，参与交互的资源都

写下你的评论...

评论0

赞11

...

Choreography可以看作一种消息驱动模式，或者说是订阅发布模式，每笔业务到来后，各个监听改事件的服务，会主动获取消息，处理，并可以按需发布自己的消息。可以把不同队列看作不同种类的消息，微服务看作消息处理函数。

Choreography实现方案多是异步的。

优点：

耦合度低，每个服务都可以各司其职。

缺点：

- 1) 业务流程是通过订阅的方式来体现的，很难直接监控每笔业务的处理，因此难于调试。
- 2) 由于没有预定义流程，所以很难在事前保证流程正确性，基本靠事后分析数据来判断。
- 3) 当一个业务流程会嵌入到多个服务中时，维护会很困难。

建议：

- 1) 小粒度的服务需要组合，服务的粒度越小，越需要组合。
- 2) 增加相应的监控系统，来保证业务顺畅进行。

3、API网关

API网关可以看作一种简单的接口聚合/拆分的方式：每笔业务到来后先到达网关，网关调用各微服务，并最终聚合/拆分需反馈的结果。

API网关其实就是一个适配网关，比如对于Web端，可以一个页面同时发起几十个请求，而对于移动端，最好是一个页面就几个请求。而采用API网关，后面的微服务可以是相同的。

优点：

对外接口相对稳定。

缺点：

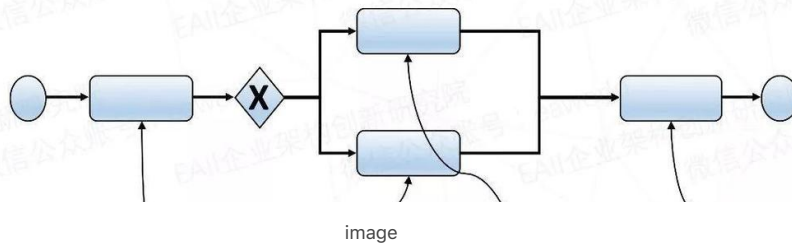
只适合业务逻辑较为简单的场景，业务逻辑过于复杂时，网关接口耦合度及复杂度会急剧升高，变得臃肿。

三、微服务编排的框架（Orchestration方式）

对编排流程、适配参数、调用链分析等方面思路的考量，构成微服务编排的框架思路。

1、编排流程的思路

原子服务提供REST接口或者监听事件，通过流程编排这些原子服务来实现一个新的复杂服务。

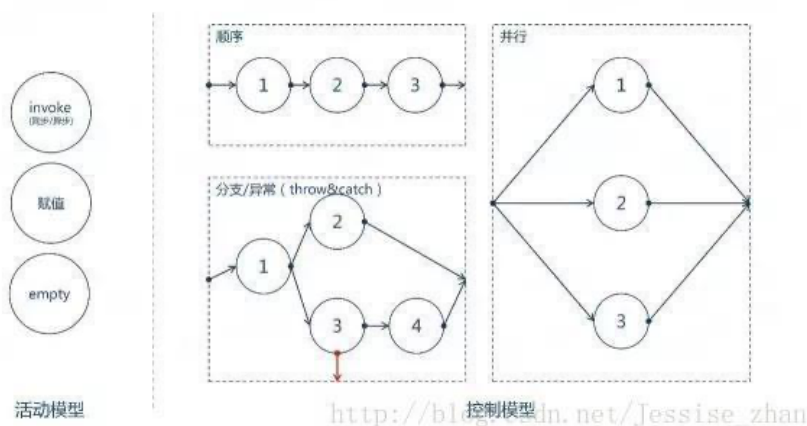


image

2、编排流程的模型

- 活动模型。例如（赋值、invoke（调用）、空）
- 控制模型。例如（顺序、分支、循环、异常抛出、异常捕获、并行）

当然，有很多编排框架提供了更多方便的活动，比如普元的编排框架提供了本地调用、rest调用、webservice调用等活动，从而在使用上更加的方便，有了这些基本的模型，我们就能方便的编排出复杂的业务流程。（如下图）



http://blog.cdn.net/Jessise_zhan

20170605155021851.jpg

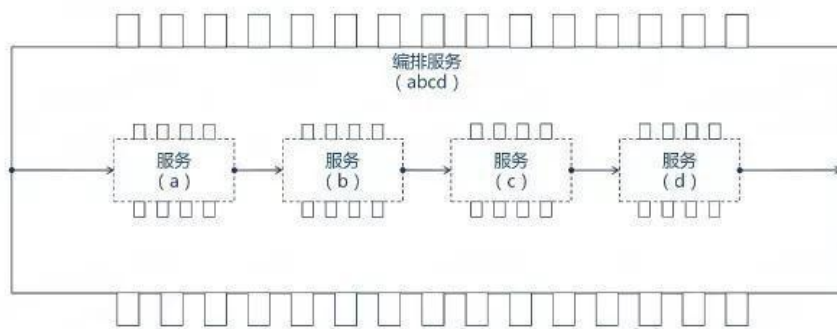
在invoke（调用）的时候我们知道有同步和异步的区别。同步实现起来简单，但是在多级级联编排的时候要避免因为某个服务的长响应时间导致雪崩效应，一般可以通过设置合理的超时时间限流和服务熔断策略来避免；同样，在异步调用的时候，应该能自动缓存上下文和避免缓存爆掉，能自动建立异步响应和请求之间的关联。同样，提到并行也必须考虑不同的聚合方式，比如是部分聚合还是全部聚合。（如下图）

微服务核心研究之--编排 - 简书

image

3、适配参数

流程编排完成之后，我们还需要给每个被编的服务提供正确的参数，是一个适配的过程。
一个编排服务（abcd）由a、b、c、d服务编排而成，每个服务都会有自己的出参入参。适配的过程就是从上下文中给入参赋值以及将出参的结果写入到上下文中。（如下图）



http://blog.csdn.net/Jessise_zhan

20170605155122120.jpg

编排服务执行到不同阶段，组成上下文的模型也是不一样的。从最初服务的开始执行的时候，上下文中只有系统级的参数和入参（请求报文），到执行完一个被编服务后上下文就会增加这个被编服务的出参（响应报文），执行上下文是一个不断增大的过程。所以适配不仅仅存在于编排服务的入参和被编服务的入参之间，还存在于被编服务和在其之前的服务出参之间。（如下图）

20170605155200918.jpg

实现适配最直接的方式是用手工编码完成点到点的映射赋值，但有更高效的方式，通过使用元数据对所有的出参和入参标记着色，然后就可以自动完成同样颜色之间的自动映射。这种标志着色可以靠数据字典实现。（如下图）

image

这里的数据字典是指抽象出业务含义的基本数据项，如账户，交易额等。通过这些数据字典可以定义出服务所需的的数据结构（服务参数和服务返回值），这样不同的数据结构之间可以按照数据字典进行自动适配。（如下图）

image

4、流水号

编排服务在执行上下文的组成模型过程中，框架也会产生一部分数据，这一部分数据主要是流水号(id)和安全方面的考量。按照《基于微服务的企业应用架构设计范式》流水号的生成应该遵循GAIR模式。

GlobalID: 全局流水号，如果请求中的globalId为空，则编排服务生成，否则保持不变。

AnswerID:响应流水号，服务提供者生成，可以作为提供者受理的凭证

InRequestID: 前台流水号，由前台生成

RequestID:请求流水号，编排服务的协调器生成，生成规则由服务提供者定义。（如下图）



20170605155354092.jpg

5、调用链分析

随着服务的增多，对调用链的分析也会越来越复杂。在一个由很多微服务组成的系统中，他们之间的调用关系会形成复杂的网络。

Google针对服务化应用全链路追踪的问题发表了Dapper论文，介绍了他们如何进行服务追踪分析，其基本思路是在服务调用的请求和响应中加入ID，标明上下游请求的关系，利用这些信息，可以可视化地分析服务调用链路和服务间的依赖关系。（如下图）



image

通过服务调用追踪生成的服务调用栈，可以查看在哪一步出现了错误，以及发现哪里的调用较慢，进行系统优化。（如下图）



image

能编排流程，能适配参数，这个编排框架已经具备运行的能力，接下来要考虑的就是事务的一致性问題。

四、微服务编排的事务一致性

依据CAP理论，分布式系统需要在可用性和一致性之间做出选择。如果选择提供一致性，就需要付出在满足一致性之前阻塞其他并发访问的代价，阻塞持续的时间往往不能确定，尤其是在系统已经表现出高延迟时或者网络故障导致失去连接时，因此，可用性一般是更多人的选择，但是在服务和数据库之间维护数据一致性是非常根本的需求，编排框架应该选择满足最终一致性。

补偿模式是一种很好的实现最终一致性的途径。补偿模式核心思想是：针对每个操作，都要注册一个与其对应的补偿（撤销）操作，一般来说操作本身和其补偿操作会在一个事务里完成，当其后续操作失败后，需要按相反顺序完成前面注册的所有撤销操作。

通过一个实例来说明：一家旅行公司提供预订行程的业务，可以通过公司的网站提前预订飞机票、火车票、酒店等。假设一位客户规划的行程是，(1)上海-北京6月19日9点的某某航班，(2)某某酒店住宿3晚，(3)北京-上海6月22日17点火车。在客户提交行程后，旅行公司的预订行程业务按顺序串行的调用航班预订服务、酒店预订服务、火车预订服务。最后的火车预订服务成功后整个预订业务才算完成。如果火车票预订服务没有调用成功，那么之前预订的航班、酒店都得取消。取消之前预订的酒店、航班即为补偿过程。（如下图）

image

在补偿模式中，我们要求参与补偿的微服务必须提供补偿操作，并且补偿操作必须是幂等的，补偿框架可以在异常时自动调用补偿操作完成补偿。

跟RPC比，补偿模式的核心价值是少了锁资源的代价，流程也相对简单，但实际操作中，补偿操作不太好定义，其中间状态处理也会比较棘手。

现在RESTful作为一个轻量级的rpc协议已经被广泛采用，能不能很好的支持RESTful服务的事务一致性也是衡量编排框架的是否成熟的一个标准。

一个通过RESTful扩展规范来支持补偿模式事务一致性的思路：通过PATCH的HTTP Method来表示compensation操作，并且支持通过服务来查询编排服务执行的状态。（如下图）

The diagram area is currently blank, showing only the placeholder text 'image'.

补偿模式一个常见的坑：由于是通过rpc的调用，因为网络和调度的关系，可能出现补偿请求比原交易先到达的情况，这会导致补偿操作直接会失败，因为此时原交易尚未发生，最终原交易到达时会被成功的执行，最终就导致了事务不一致。

填这个坑的办法就是在编排框架发现补偿操作补偿的原交易不存在时，补记录一条原交易的流水，从而保证原交易晚到时会因为记录流水失败而不会成功。（如下图）

The diagram area is currently blank, showing only the placeholder text '图片描述'.

五、微服务编排的监控工具支撑

在生产环境中，我们需要通过查看日志来排除故障，应该有支持日志全路径回放的工具，来帮助我们快速定位故障。

本文所讲的编排实际是编制，是一种集中式的控制，也就意味着如果被编排的服务有响应缓慢的情况，可能会影响到其他服务。这时候我们需要更快的监控来帮助我们发现这类服务，从而尽早优化。监控工具需要具备以下功能：

- 通过可视化分布式系统的模块和他们之间的相互联系来理解系统拓扑。点击某个节点会展示这个模块的详情，比如它当前的状态和请求数量。

- 实时监控应用内部的活动线程。
- 可视化请求和响应数量来定位潜在问题（请求时间段分布、错误请求、响应时长等）。
- 在分布式环境中为每个调用生成可视图，定位瓶颈和失败点。
- 查看应用上的其他详细信息，比如CPU使用率，内存/垃圾回收，TPS，和JVM参数。

11人点赞 >

服务端技术

"小礼物走一走，来简书关注我"

赞赏支持

还没有人赞赏，支持一下

沉落的星星

15年互联网产品管理 关注用户需求满足与商业目标实现的平衡 专...

总资产13 (约1.26元) 共写了22.8W字 获得243个赞 共172个粉丝

关注

被以下专题收入，发现更多相似内容

我的微服务

解决方案

微服务

推荐阅读

微服务架构的核心要点和实现原理

摘要：本文中，我们将进一步理解微服务架构的核心要点和实现原理，为读者的实践提供微服务的设计模式，以期让微服务在读者...

大齐老师

阅读 3,155 评论 0 赞 19

更多精彩内容 >



微服务架构下的事务一致性

本文转载自： https://mp.weixin.qq.com/s?__biz=MzAwNzM1NjQyNg==&m...

大风过岗

阅读 2,076 评论 0 赞 3

传统分布式事务不是微服务中一致性的最佳选择

微服务架构中应满足数据最终一致性原则

微服务架构实现最终一致性的三种模式

对账是最后的终极防线

微服务编排之道

感谢肥侠、Yi YANG、loveis715、郭斌、李斌、itegel、田向阳、Chris Richardson等...

EAI企业架构创新研究院

阅读 13,550 评论 2 赞 12

微服务的4个设计原则和19个解决方案

转载本文需注明出处：微信公众号EAWorld，违者必究。微服务架构现在是谈到企业应用架构时必聊的话题，微服务之所...

EAI企业架构创新研究院

阅读 2,152 评论 0 赞 40

【鹊桥仙】玉兰

霓裳束素，空阶霜冷，玉面低回知语。新妆谁似试羽衣，照冰轮，清影芬馥。绵苞乍露，临风皎皎，傍柳依翠初舞...

眉山月 阅读 71 评论 6 赞 6