应用的微服务化分解及构造

MF1933099 卫昱阳

研究背景

多年来，传统的 IT 架构一直面临如下的问题：

1. 扩展困难：传统的烟囱式架构，接口较多，耦合紧密，使得快速扩展较为困难。
2. 成本高昂：低下的软件复用程度使得开发成本随之上升。

而基于传统 IT 架构开发的单体应用因此也面临许多的缺点：

1. 应用的逻辑较为复杂、不同的功能模块之间耦合度高、涉及的代码量庞大,修改代码难度高,版本在维护更新时迭代复用程度低。
2. 由于系统模块之间的耦合程度高，因此对于错误的隔离性差,任何一个模块产生的错误都有可能造成整个系统无法使用。
3. 在应用程序中涵盖了所有的业务逻辑功能,涉及到大量启动模块,导致系统启动的时间较长。
4. 可伸缩性差，如果需要对系统进行扩容，只能针对整个应用都进行扩容,无法实现对应用中某个功能模块进行单独的扩容。
5. 由于问题的修复需要对于整个应用系统进行，因此修复应用中出现的问题时所需要的周期较长。

随着软件规模的扩大，这些缺点导致的问题会越发严重，直接影响到软件开发的效率和成本。而使用微服务化的开发方式，恰好可以解决单体应用的这些缺点，相比单体应用，微服务有如下的优势：

1. 将冗余的单体应用拆分为多个高内聚、低耦合的服务功能，便于开发团队进行代码管理。
2. 开发者可以独立自由开发,自主选择开发技术,提供 API 服务。
3. 开发者能够对指定的服务进行单独的修改。
4. 可以进行快速的迭代开发,迭代开发的周期短。
5. 每个微服务的部署相对独立,可以选择更适合于服务资源需求的硬件进行使用,同时开发者不需要关心协调其它服务部署对本服务的影响,从而使得部署效率得到了提高。

正是由于微服务的这些优势，可以使用微服务的开发方式提供基于不同领域的功能模块（譬如为一个天气类应用提供基本的定位、天气预报等功能模块供开发者使用），简化开发过程，提高开发效率，降低开发成本。

研究现状

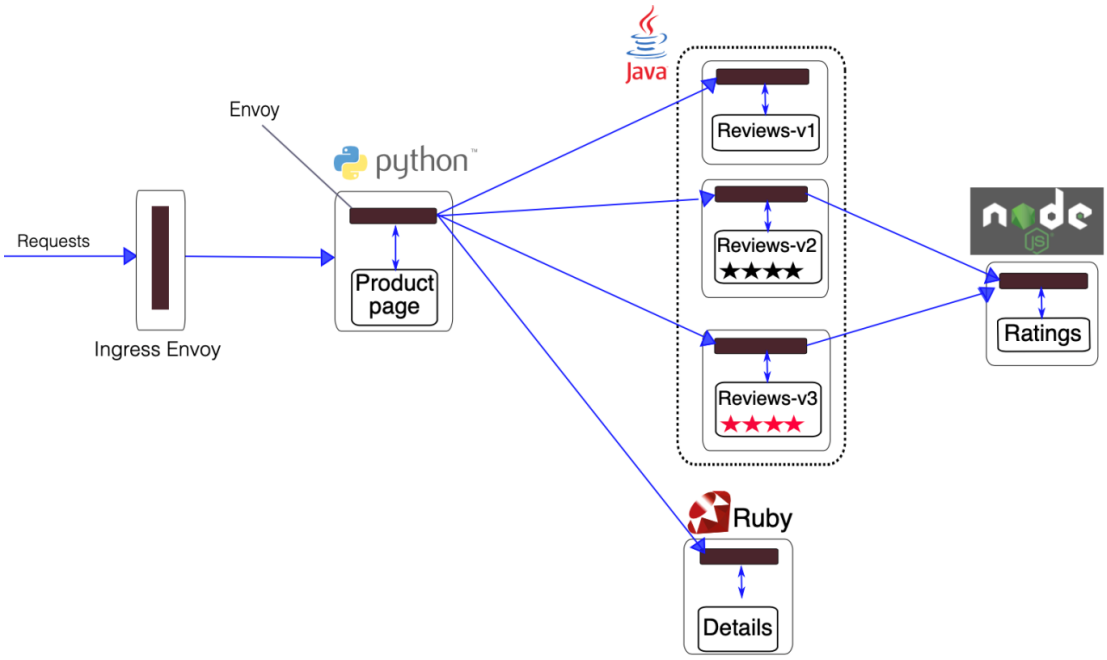
目前在 Web 端等开发平台已经有很多较为成熟的微服务研究和相关的生产实践，下面主要介绍 istio 和 Netflix 这两个研究和实践的内容。

istio

istio 是一个完全开源的服务网格，可以透明地在现有的分布式应用程序上分层。它也是一个平台，其中包括允许它集成到任何日志记录平台、遥测或策略系统的 API。istio 的多样化功能集使得分布式微服务架构能够高效地运行，此外还提供保护、连接和监控微服务的统一方法。
istio 具有很多优势：

- HTTP、gRPC、WebSocket 和 TCP 流量的自动负载均衡。
- 通过丰富的路由规则、重试、故障转移和故障注入，可以对流量行为进行细粒度控制。
- 可插入的策略层和配置 API，支持访问控制、速率限制和配额。
- 对出入集群入口和出口中所有流量的自动度量指标、日志记录和跟踪。
- 通过强大的基于身份的验证和授权，在集群中实现安全的服务间通信。

我们以 istio 的一个官方示例 Bookinfo（下图）为例简单介绍 istio 应用。



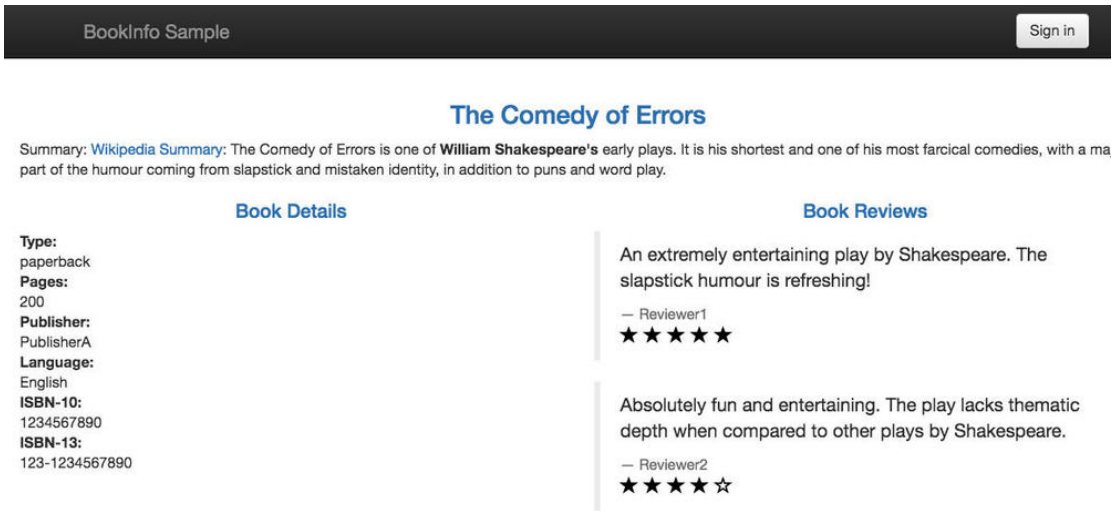
Bookinfo 应用分为四个单独的微服务：

- productpage：productpage 微服务会调用 details 和 reviews 两个微服务，用来生成页面。
- details：这个微服务包含了书籍的信息。
- reviews：这个微服务包含了书籍相关的评论。它还会调用 ratings 微服务。
- ratings：ratings 微服务中包含了由书籍评价组成的评级信息。

reviews 微服务有 3 个版本：

- v1 版本不会调用 ratings 服务。
- v2 版本会调用 ratings 服务，并使用 1 到 5 个黑色星形图标来显示评分信息。
- v3 版本会调用 ratings 服务，并使用 1 到 5 个红色星形图标来显示评分信息。

Bookinfo 运行效果如下：



在这个应用中：

- 用户可以使用登录、登出服务（点击 Sign in 按钮）
- 刷新页面，bookinfo 从信息源获取原信息（包含：details、reviews、ratings），将原信息传递给 productpage
- productpage 对获取的原信息进行加工，并将加工后的信息呈现在生成的页面上。
- productpage 提供文字、图片类型的信息展现的服务

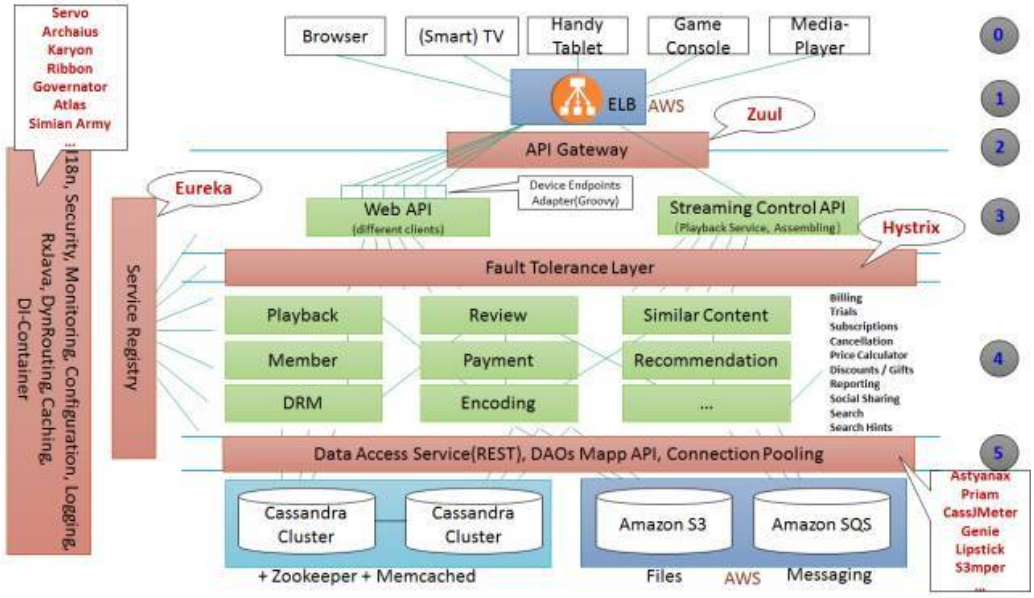
Netflix

Netflix（官方中文名：网飞）是一间在世界多国提供网络视频点播的 OTT 服务公司，并同时在美国经营单一费率邮寄 DVD 出租服务。Netflix 还是业界微服务和 DevOps 组织的楷模，有大规模生产级微服务的成功实践：

下图展示了 Netflix 从外到内的微服务分层视图，在这个分层视图中，我们重点关注第 3 层 “聚合服务层” 和第 4 层 “后台基础服务层”：

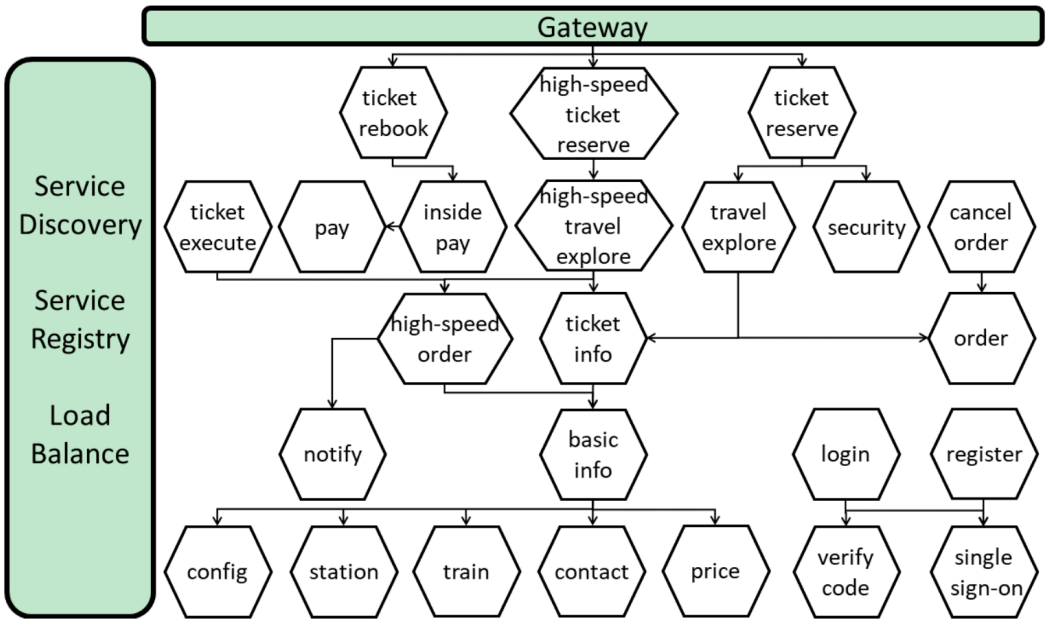
- 聚合服务层：负责对后台微服务进行聚合裁减加工后暴露给前端设备，在 Netflix 的体系中，该层也称边界服务层(Edge Service)，或者设备适配层。考虑到设备的多样性和前端业务的多变性，Netflix 前端团队大量使用 Groovy 脚本作为聚合层的主要开发语言，同时兼容 Java 又具有脚本易于变更特性。
- 后台基础服务层：提供支持 Netflix 业务的核心领域服务（Playback, Member, Review ,Payment etc），在 Netflix 体系中，该层也称为中间层服务（Middle Tier Service)。

这两层统称为 Netflix 的微服务，总体实现 Netflix 业务能力输出。



TrainTicket

尽管在某些领域已经有大规模的微服务架构的生产实践，然而在业内对于微服务系统（Microservice System）的定义标准仍旧处于探究的范畴。现有的相关工作尝试定义微服务平台的行业标准，对某一领域的微服务系统平台进行了探究——设计了一个火车票购票系统的微服务平台，如下图：



在这个中等规模的微服务系统中，根据火车票购票的领域特征，按照功能模块进行微服务的划分设计。

未来研究趋势

可以采用领域驱动（Domain-driven）的方式，对于某个领域的应用进行收集，分析其源代码以及应用运行使用情况，提取其中的领域知识，从而构建出该领域的模型。在得到领域模型之后，基于模型设计该领域应用开发的微服务平台，从而为该领域应用进行微服务开发提供帮助。

在这个微服务平台的基础上，可以进行如下的研究：

1. 微服务应用开发中，代码等重用度的度量。
2. 一个应用最合适的微服务化程度（应用中不是所有的内容都需要微服务化）及其优化点，如何度量这个优化程度并探寻其最优的程度；构造微服务的策略。
3. 探究能够证明微服务的相对独立性，无副作用（相比 API 对全局变量的修改可能造成一些副作用）的案例。

此外，使用微服务开发应用的成本并不低，大企业能够承担微服务开发的成本而中小企业未必有这个经济实力，而搭建了微服务平台，将减少中小公司或个人开发者开发微服务架构软件的成本，因此，搭建一个工业级的微服务平台对于未来微服务应用的开发具有深远的价值和意义。