链滴

从历史看未来，大规模微服务系统的困境 ---- 基于消息的架构的回归

作者：linker

原文链接：https://ld246.com/article/1528704444140

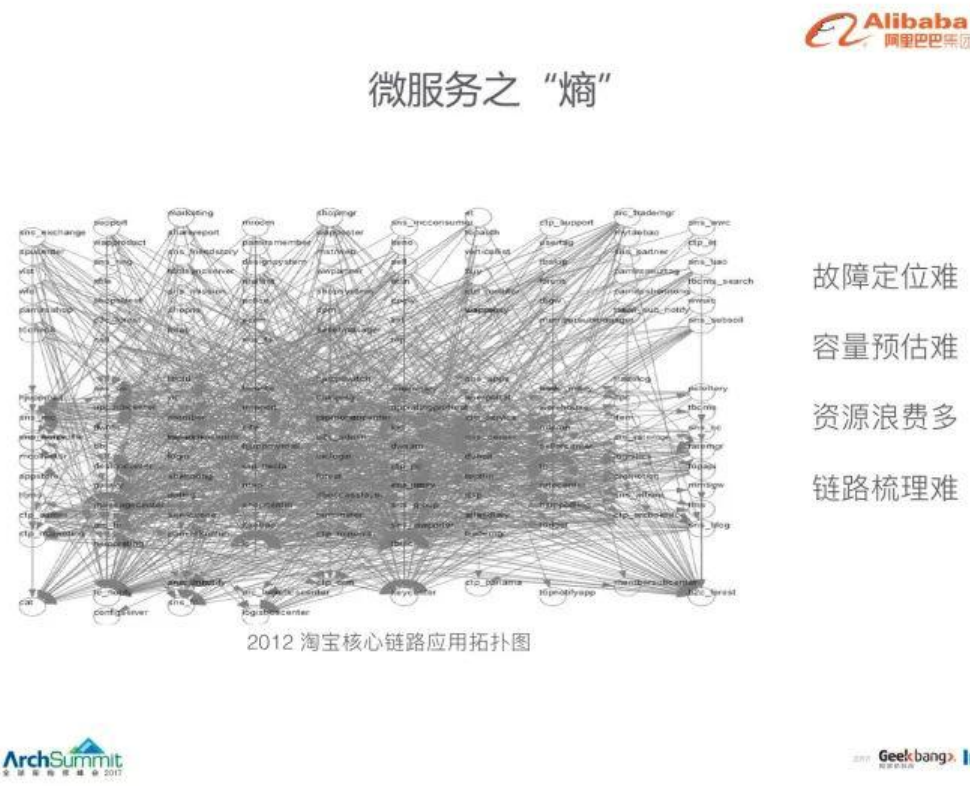
来源网站：链滴

在大规模分布式系统的架构上，微服务系统是现在很多大型互联网公司的架构方向。

这是一个务实的很好的方向，相对于旧的宏服务来说。

然而，像淘宝这种规模的系统，微服务很容易陷入一个困境，就是 聚合层的扇入扇出过大带来的接口多复杂爆炸的问题，以及聚合层过小，导致客户端请求域名过多复杂度爆炸的矛盾。

先看下淘宝现在的服务架构图：



由此图我们可以看出每一个微服务的聚合层都有非常高的扇入，处于中间的聚合层还有很高的扇出。

这就是基于RPC的微服务架构的根本缺陷所在。

下面我从几个方面来分析下原因：

RPC的流行，源于HTTP/AJAX在互联网应用中流行，在此之前，更多更大的IT系统，诸如电信、银行军事都是基于消息的。以标准最全规模最大的电信系统为例，全球的电话系统都是基于几套信令标准互通的。所谓信令也就是控制消息。电信系统的优点是，学院派，一开始就是把问题想清楚，缺点是杂。所以，在以个人网站起家的诸多互联网巨头带起来的风气里，自然看不到电信系统的影子。而互网应用，包括电商、O2O、社区等都是采用的AJAX/RPC为基础的SaaS、微服务架构。本质的根源是互联网应用出身草莽，创业开始头半年，快速上线比架构可演进要重要一千倍。

但是有两个例外，一个是腾讯的QQ，一个是诸多PC客户端网游。由于腾讯QQ的创始团队有浓厚的信背景。而网游追求单服务器高负载能力以降低成本。在互联网初期，服务器能力低下的时代，用HTP/AJAX的网页单服务器带不了多少人在线，不适合低成本高在线人数为追求的MMORPG类客户端网。

从历史我们可以看出，选择什么架构，取决于：速度和成本的折中。

因此，我们应该看到虽然RPC/微服务似乎成为了互联网的唯一选择，并不是经过深思熟虑的长远考虑更多的是基于惯性，而这个惯性的起点基于快速上线一个简单Web站点的需求。

说了这么多，到底RPC/微服务 和 消息/信令 系统的架构层面的区别是什么呢?

区别是：有没有在系统每一层固定下所有通过该层的 通讯协议 的细节。

假设，我们有一个 系统 负责提供 算术 功能。

在基于RPC/微服务的系统中，可能设计是，分两层，对外网关层 叫 算术Gateway，包装内部 加法Sevice、减法Service、乘法Service、除法Service的所有的接口，对外提供服务。这样设计的结果是，术Gateway的对外的接口非常多，而且要重复下层服务的Schema。**这就导致了Gateway需要知道层的业务逻辑。**对接口的依赖也是一种依赖,对于网关来说,即便是和微服务云内部的接口构成了依关系,也是一个巨大的负担.

在基于消息/信令的系统中，可能涉及是，也有一个算术Gateway，可以接受 一种类别叫 算术运算的息。每个消息还有子类别，可能是 加、减、乘、除。 这样的好处是，Gateway 无需理解到子类别的理逻辑和接口细节，只要知道两点：1. 自己能处理的主消息类型 2. 下层所能处理的子消息类型。如Gateway可以方便的路由消息给下层的消息处理器。这也是电信系统的通用设计方式，每一层信令系都只针对自己的业务域，信令包含子信令，信令的处理器只要知道能处理对应子信令的消息处理器并需要了解子信令的Schema。

在国内的网游，以及交通银行的手机银行系统中，广泛的使用Erlang/OTP平台。该平台来自于世界大的电信设备制造商爱立信。在Erlang/OTP中，每个Process都是一个Actor负责处理自己邮箱的消。而亚马逊最新的ServerLess架构却和二十年前的Erlang/OTP架构有异曲同工之妙。

由此可见，ServerLess/Actor/消息/信令，其实有很深的设计渊源，是同一种思想的不同领域实现。质上就是把消息Schema的固化和消息的处理解耦。在RPC/微服务的架构中，每一层，都必须用某种言/IDL唯一缺点的描述自己能处理的消息的全部Schema。而消息/信令架构天然没这个约束。

以上是从Schema的层面分析的区别，从同步和异步的暗示上来说，RPC/微服务架构诱骗开发人员用同步的思想来设计接口，无端制造出了超时重试导致雪崩、同步阻塞浪费线程等问题。任何一个大型统天然是异步，任何同步的系统的努力都会随着系统的增大而成本越来越高。异步、非阻塞是大部分息系统设计背景。这点更加重了RPC/微服务架构的使用成本。

更糟糕的是，随着系统规模的扩大，很多RPC/微服务系统发现自己必须存在很多环路调用，但是环路是RPC/微服务架构的大敌。为此不得不引入诸如Kafka等消息队列来引入异步性，解除环路。于是系的复杂度Double。

为何系统复杂了，就会出现环路，这是因为任何复杂系统必然是一个 图计算 系统，而且 必然是一个**向有环图**。为了把一个有向有环图适配到一个树状的RPC/微服务架构中，架构师们花费了不少脑力。然，这是一种巨大的浪费。

以上是从 架构抽象层面 的分析。

另一个角度来说，RPC的易用性并不比消息高，否则，我们用的终端命令行就应该是函数调用的样式作，而不是现在的交互会话的样式了。人类更喜欢会话方式。更不用说，在每个『消息』的末尾加上&』就可以异步化处理消息的简便表达方式，好理解好使用。

综上所述，人类徘徊了20多年的以后，大规模分布式系统的架构来设计又慢慢的回到了 消息/信令 架。