我们这里提出的方法完全是“黑盒”方式，把监控功能从面向功能的微服务框架中解耦。这种方法能够做到既不侵入也不破坏源代码，同时也不需要微服务进行调整。由此可知这对于已经投入生产环境的系统是个非常好的方法。为了完成这个目标，我们使用并更改了网飞公司的Zuul网管，来从微服务发出的请求中收集性能。基于例如返回时间，请求源和请求目的地等性能，我们整合了这些原始数据，然后输出一个简洁的相关信息，例如平均回复时间，拓扑结构和全部服务特性描述。

我们的结果显示我们可以获得相关且有效的信息给系统管理员，尽管我们使用的是无侵入式的方法。我们不需要规范微服务或者向架构中添加代理，只需要微服务模块已经需要的组件。因此，展示的这个方案很有用，特别是对于采用了微服务的经常改动和有弹性的系统。

论文剩余部分按如下结构组织。第二章描述我们处理的问题和我们解决它所使用的方法。第三章描述实验参数设置。第四章展示和评估实验结果，这个方法的好处和它的局限。第五章展示相关的工作。最后，第六章做出总结并提出未来的工作方向。

第二章 提出的方法

在这片论文中我们处理了监控微服务体系结构的问题。在垂直解决方案中，监控简单一些，因为应用不会随着时间变化很大。微服务从新的开发模式中逐步形成的，例如Agile或DevOps和昕的部署技术，例如容器。系统监控却不随着这些变化而变化，依然基于单体系统的应用和技术。第五章我们讨论世界上主要的科技公司都是如何挣扎在这个问题上的，他们被迫为他们的需求创建自定义的平台。确实，监控在高度动态变化的系统中是一个很复杂且困难的问题。

我们从不同的角度分析监控问题。一个关于监控的典型方法会包含规范或添加代理在尽可能多的层上，从主机到中间件，一直到应用层上。图片一展示了三个微服务的调用序列。A中一些方法调用B中的一个方法，而B中这个方法又调用了C中的一个方法。为了把这种相互依赖的调用关系传递给监控系统，消息必须携带一些标识符来体现关联关系，例如一个带有一些标识符(idx=idy)的HTTP头。不幸的是这需要修改应用源代码。虽然这种技术创建了几个监控点，但它们也是将监控与业务逻辑结合起来引起的额外故障和维护点。这违背了微服务方法论，它遵循面向函数的细粒度模块的前提。为了消除这种规范化模块的需要，我们采用了一种非传统的方法。了解到微服务使用了网关来做服务发现和重定向请求，我们添加了收集一些监控信息的功能到这个网关中。这个想法是让网关收集信息，例如返回时间，调用方和被调用方的IP和端口，以及被调用函数的身份信息。这种方法带来了一些优势，例如在不影响系统可伸缩性的情况下将监控系统和应用程序分离，因为网关和相关服务是水平可伸缩的。

在下一子段中，我们详细讨论使用的方法。首先，