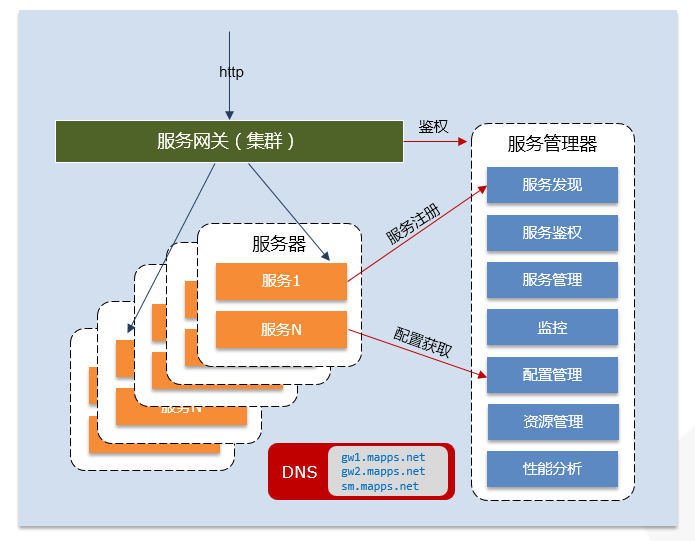
**微服务框架知识传递**

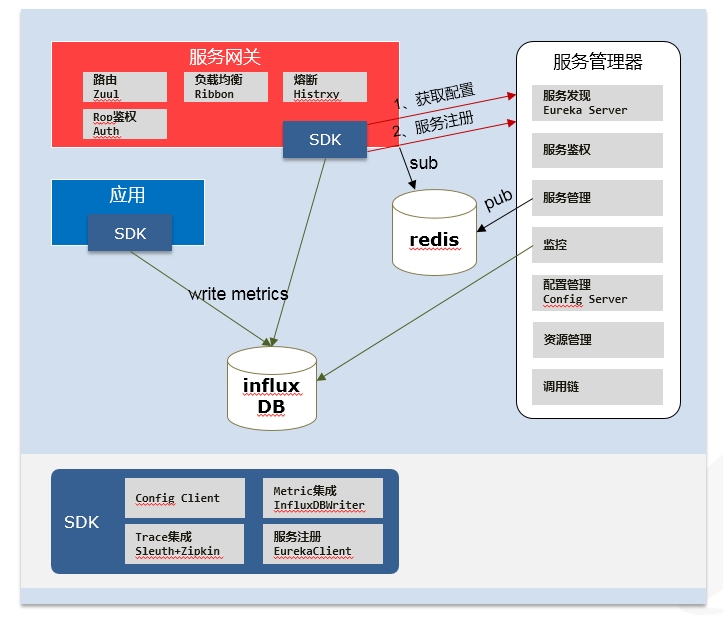
# 整体架构



微服务架构主体由三个逻辑部分组成，服务网关、服务管理和应用（服务）组成；通过服务管理提供服务注册、服务发现以及配置管理，网关和应用注册到服务管理器，请求通过网关分发，网关从服务发现获取应用（服务）列表。配置管理对应用的参数、依赖的服务和资源进行配置、分配。

# 技术实现

## 组件视图



有以下组件：

服务网关（mapps-gateway）：

路由：将请求分发到不同注册的应用（服务）上；

负载均衡：在同一应用（服务）多台服务器之前通过不同分负载策略分发；

熔断：为避免雪崩效应，对错误的服务进行隔离或熔断

Rop统一鉴权：对于需要鉴权控制的服务，通过token或者appkey的方式进行统一鉴权；

服务管理（mapps-servicemanager）：

服务发现：提供一种机制，能够进行服务注册、服务发现和服务的状态管理；

服务管理：对于注册的服务进行管理，如路由、鉴权等；

配置管理：服务应用（服务）的参数、依赖的服务和资源进行配置、分配；

监控：对SDK采集的metrics数据进行分析，实现系统的监控和预警；

资源管理：对于系统接入的资源进行管理，如数据库、redis等；

调用链：对系统中的调用进行跟踪，实现性能分析、问题跟踪，未实现，后续可基于zipkin改造。

SDK（mapps-msclientsdk）：

配置客户端：从服务管理获取配置，并通过环境变量的修改实现系统参数的生效。同时实现系统的restart的功能，便于服务管理控制应用（服务）。

Metrcis集成：实现对metrics数据采集到集成；

Trace集成：调用跟踪的集成，实现调用过程的跟踪；

服务注册：实现服务的注册功能，注册应用（服务）到服务管理；

## 实现方式

开源+自研，开源为主，自研为辅。基于Spring Cloud Nexflix，加上部分自研开发

服务网关：

路由：利用Zuul开源组件实现，路由表部分通过pubsub机制和服务管理实现同步；

负载均衡：基于Ribbon组件实现；

熔断：基于Histrxy实现；

Rop统一鉴权：自研；

服务管理：

服务发现：集成Eureka Server；

服务管理：自研；

配置管理：自研；

监控：基于influxdb提供的能力，自研；

资源管理：自研；

SDK：

配置客户端：配置客户端自研，通过rest请求获取， 重启功能集成了Spring Cloud的RestartEndpoint。

Metrcis集成：集成Spring Boot Actuator的metric功能，自研实现InfluxDB的writer以及Rop的性能指标的采集；

Trace集成：基于Spring Cloud Sleuth，自研Rop服务的跟踪, 采集的trace数据输出到Ziklin；

服务注册：集成Spring Cloud Eureka ；

# 参考资料

Spring Cloud： <http://cloud.spring.io/spring-cloud-static/Dalston.SR4/>

Netflex OSS：<http://netflix.github.io/>