车凌 HYPERFLOW 车云平台

系统接入与使用手册

2024年12月5日

变更履历

序号	版本号	修改条款及内容	修改人	审核人	批准人	修改日期

目　录

1. 编写目的

用于各技术团队接入车凌 HyperFlow 车云平台，获取平台能力，并应用到实际项目。

1.1. 外部系统集成&车厂系统集成

外部系统集成&车厂系统集成在整个架构中的定位是很薄的一层。类似接入层，提供转发，路由，鉴权等功能。
对外提供接口的可选技术方案：

1. 对于非常简单的接口集成，采用 Zuul 作为 http 协议转发封装组件，利用 Zuul 中的 Filter 对 http 协议进行封装，转换等。
2. 对于复杂的接口集成，可以将其设计为单独的服务，封装调用细节，提高协议的通用性和可适配性。

对外接口调用的可选技术方案：

1. 所有调用都需要通过“外部系统集成&车厂系统集成”发出。
2. 技术实现上，简单的接口通过 Zuul 对外转发。复杂的使用单独的服务向外转发。

1.2. 术语解释

序号	术语名称	说明
1	CMS	内容管理平台
2	GIS	地理信息服务平台
3	TSP	车联网服务平台
4	LBS	Location Based Service 基于位置的服务
5	CAN	CAN 是控制器局域网络(Controller Area Network, CAN)的简称，是国际上应用最广泛的现场总线之一。 CAN 总线协议已经成为汽车计算机控制系统和嵌入式工业控制局域网的标准总线协议。
6	OBD	OBD 是英文 On-Board Diagnostic 的缩写，中文翻译为“车载诊断系统”。这个系统随时监控发动机的运行状况和尾气后处理系统的工作状态。通过标准的诊断仪器和诊断接口可以以故障码的形式读取相关信息。根据故障码的提示，维修人员能迅速准确地确定故障的性质和部位。
7	CRM	客户关系管理系统。企业利用信息技术以及互联网技术来

		协调企业与顾客间在销售、营销和服务上的交互，从而提升其管理方式，向客户提供创新式的个性化的客户交互和服务的过程。其最终目标是吸引新客户、保留旧客户以及将已有客户转为忠实客户。
8	Call Center	呼叫中心是充分利用现代通讯与计算机技术，如 IVR（交互式语音应答系统）、ACD（自动呼叫分配系统）等，可以自动灵活地处理大量各种不同的电话呼入和呼出业务和服务的运营操作场所。

2. 业务需求

详见：《车联网产品功能定义清单》、PRD、SRS 等需求文档。

3. 系统架构

3.1. 总体目标

1. 可靠性（Reliable）：系统对于商业经营和管理来说极为重要，因此系统架构必须非常可靠。
2. 安全性（Secure）：系统所承载业务的商业价值极高，系统安全性非常重要。
3. 可扩展性（Scalable）：软件必须能够在用户的使用率、用户的数目增加很快的情况下，保持合理的性能，只有这样才能适应用户的市场扩展得可能性。
4. 可维护性（Maintainable）：系统的维护包括两个方面，一是排除现有的错误，二是将新的软件需求反映到现有系统中去。一个易于维护的系统可以有效地降低技术支持的花费。
5. 客户体验（Customer Experience）：软件对用户必须易于使用。

3.2. 设计原则

本系统采用微服务软件架构风格，以专注于单一责任与功能的小型功能区块为基础，利用模组化的方式组合出复杂的大型应用程序，各功能区块使用与语言无关的 API 集相互通讯。
基本原则：
1. 隔离
服务必须设计为单独相互隔离工作。每个服务应该能够处理其自己的故障，而不会影响或破坏整个应用程序或系统。 隔离和解耦特性使服务能够非常快速地从故障状态中恢复。

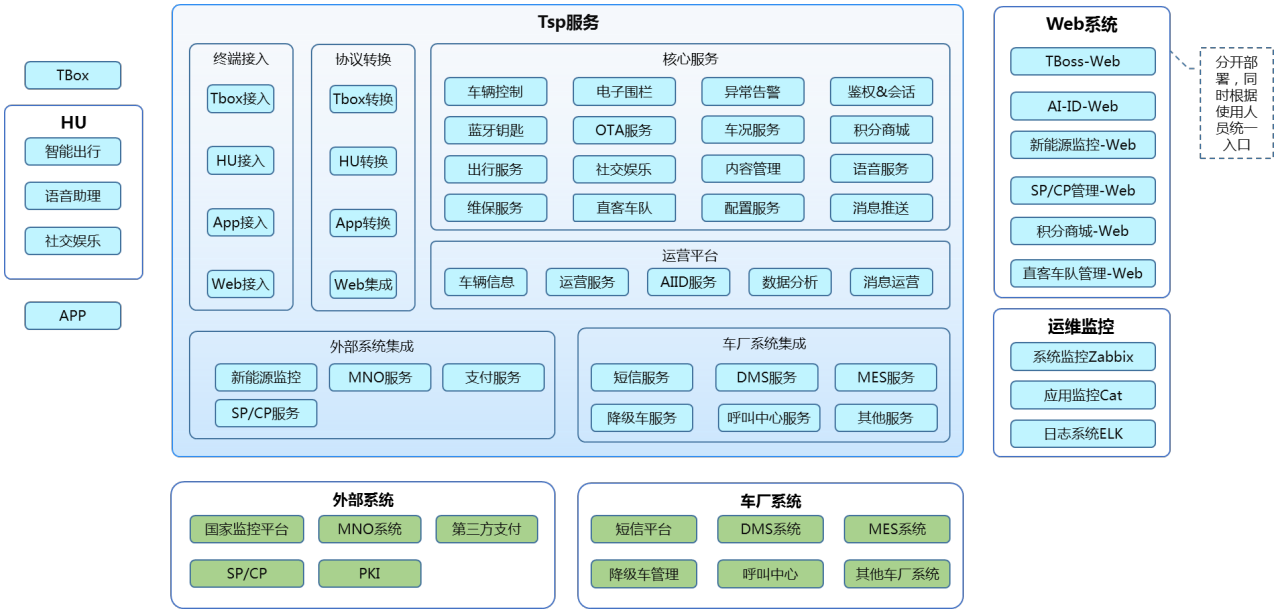
2. 自治性

服务必须设计为自主自治的。 它必须具有凝聚力，能够独立地实现其职能。 每个服务可以使用良好定义的 API（URI）独立调用。 API 以某种方式标识服务功能。

3. 单一责任

服务必须设计为高度凝聚。 单一的职责（责任）原则是服务只执行一个重要的功能。单一责任功能具有以下优点：服务组合无缝，更好的扩展，可重用性，可扩展性和可维护性。

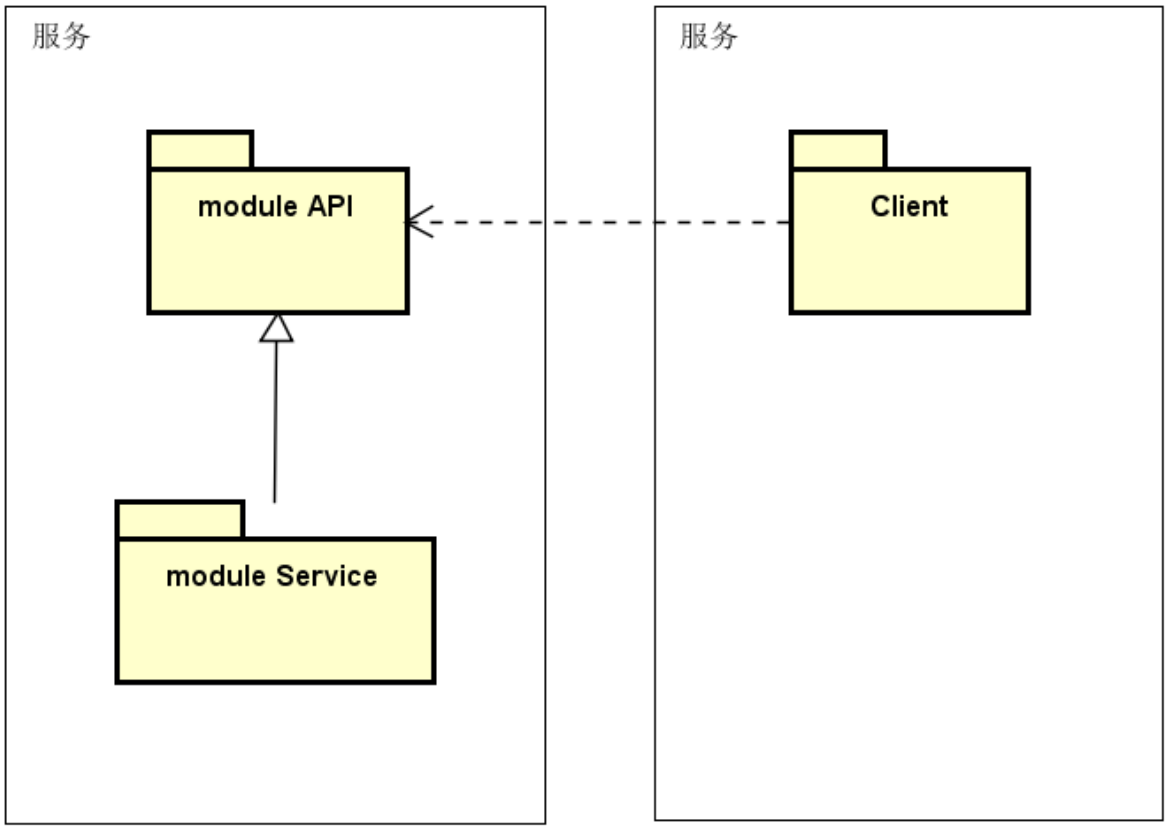
3.3. 架构图



3.4. 接口列表

序号	接口
1	《Tbox 与 Tsp 接口文档》描述 Tbox 与 Tsp 的交互方式，接口等。
2	《HU 与 Tsp 接口文档》描述 HU 与 Tsp 的交互方式，接口等。
3	《App 与 Tsp 接口文档》描述 App 与 Tsp 的交互方式，接口等。
4	《GBT 32960.2-2016 电动汽车远程服务与管理系统技术规范》
5	《物联网接口文档》按照运营商接口文档进行。
6	《第三方支付》
7	《SP/CP 接口文档》SP/CP 未定厂，不同的 SP/CP 会有不同的接口。
8	《短信平台接口文档》
9	《DMS 与 Tsp 接口文档》
10	《MES 与 Tsp 接口文档》
11	《降级车管理接口文档》
12	《呼叫中心接口文档》

3.5. 服务内的 module 划分



说明：

1. 每个服务至少包括 2 个 module：module API，module Service。
2. module API：负责抽象接口和交互时接口用到的 DTO
3. module Service：实现 module API 中的接口，开发 api 提供实际的功能。
4. Client：依赖 module API，通过 module API，或者接口格式，并调用实际的 module Service

3.6. 终端接入

终端接入层，是微服务架构中的网关。目前主要包括 4 个接入：TBox 接入、HU 接入、App 接入、Web 接入。

每个接入的职责如下表所属：

分类	服务	说明
终端接入	TBox 接入	所有 TBox 的请求都经过这个服务进入。与 TBox 的长连接也由这个组件负责。主要提供 TLS 的接入方式。从通信安全考虑需采用 TLS 协议。其他接入安全也由接入层负责。
	HU 接入	所有 HU 的请求都经过这个服务进入。主要提供 http/https 的方式。从通信安全考虑需采用 https 协议。其他接入安全也由接入层负责。
	App 接入	所有手机端的请求都经过这个服务进入。主要提供 http/https 的方式。其他接入安全也由接入层负责。

	Web 接入	所有 Web 端的请求都经过这个服务进入 Web 系统。主要提供 http/https 的方式。主要提供 http/https 的方式。其他接入安全也由接入层负责。

主要担负如下职责：

1. 内外的隔离

为了安全的考量，不允许外部直接访问。内部系统只接受 API 网关转发过来的请求。网关通过白名单或校验规则，对访问进行了初步的过滤。这种软件实现的过滤规则，更加动态灵活。

2. 请求路由

所有请求经过网关通过服务发现路由的指定的服务。网关可以对所有的请求有一个统一灵活的管理。由此带来的请求压力可以通过横向扩展来解决。

3. 鉴权

由于网关层是统一的入口，所以鉴权都在这个层面进行。通过调用内部的鉴权&会话服务，对所有的请求进行统一的权限认证。

4. 流量控制

在某些情况需要对 API 做限流保护。网关层可以完成请求限流功能。

5. 分析与监控

网关层是整个平台的入口，可方便的查询运行中 API 的详细调用信息。方便运维管理。对 API 的分析提供方便可靠的数据依据。以便 API 的后期迭代与维护，提高效率。

6. 日志记录

提供完整的对整个平台的日志请求收集汇总工作。

3.7. 协议转换

协议转换层，负责协议的转换，请求的集成功能。

分类	服务	说明
协议转换	Tbox 转换	负责 Tbox 协议从接入进来以后将协议进行转换后发送到后台服务。暂定作为一个独立的部署单元，后续可能采用 Jar 包的方式。
	HU 转换	负责 HU 协议从接入进来以后将协议进行转换后发送到后台服务。暂定作为一个独立的部署单元，后续可能采用 Jar 包的方式。
	App 转换	负责 App 协议从接入进来以后将协议进行转换后发送到后台服务。暂定作为一个独立的部署单元，后续可能采用 Jar 包的方式。
	Web 集成	负责 Web 请求接入后，集成多个后端服务请求结果，一起返回给客户端。暂定作为一个独立的部署单元，后续可能采用 Jar 包的方式。

3.8. 核心服务

核心服务的定义是面向非运营类业务的功能。每个服务都拥有自己的缓存和数据库。服务之间通过 REST API 实现调用。服务之间在同一个层面不进行垂直的划分。服务内部可以使用自己的中间件，例如消息队列等。

主要包括 16 个服务：

分类	服务	说明
核心服务	车辆控制	车辆控制、安防密码、故障诊断等下行指令都在这个服务完成。
	电子围栏	提供电子围栏的保存，出栏入栏的判断等功能。
	异常告警	负责停车异动、熄火状态异常移动、车辆震动、被盗报警等从 Tbox 上报的异常情况的接受和处理
	账户&鉴权	提供对用户、鉴权、Session 功能。
	蓝牙钥匙	提高蓝牙钥匙的相关功能。
	OTA 服务	提供 OTA 相关功能。
	车况服务	接受车辆平时上传的车况，提供车况接受，车况查询，车况告警等服务。
	积分商城	提供积分商城等相关服务。
	出行服务	主要为 HU 提供智能出行相关服务。
	社交娱乐	主要为 HU 提供社交娱乐相关服务。
	内容管理	提供对内容的管理。类似 CMS。
	语音服务	主要为 HU 和手机提供语音相关服务。
	维保服务	提供电子说明书，维保提醒等于维修保养相关服务。
	直客车队	提供与直客相关的服务。
	配置服务	提供所有 HU、APP、Tbox 相关的配置保存和查询功能。
	消息推送	提供所有 HU、APP 的消息推送能力。不包括短信和 Tbox。

3.9. 运营平台

运营平台定位于提供基础数据的维护以及运行相关能力。为核心服务提供基础数据支持。

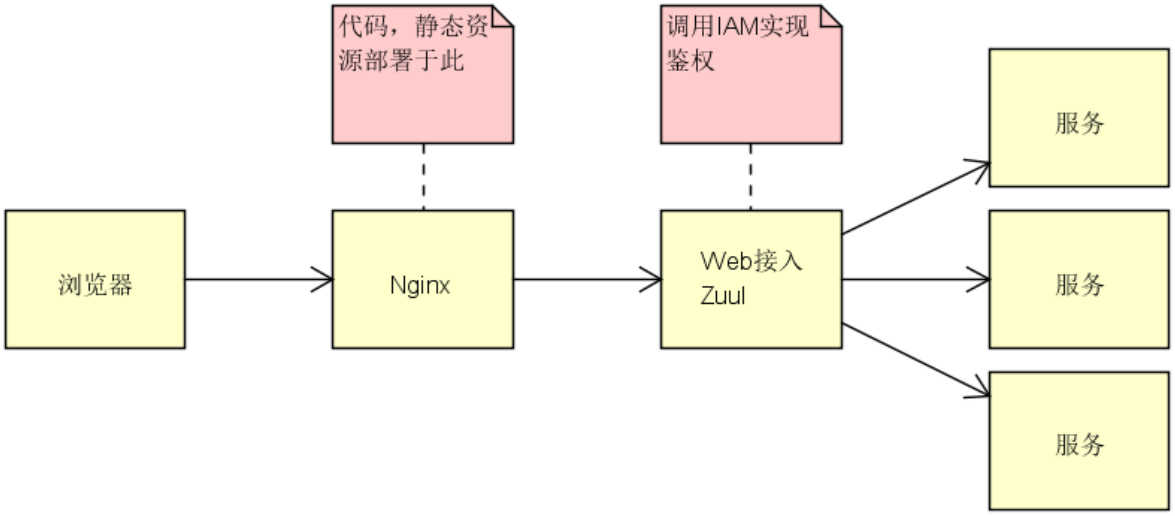
主要包括如下 5 个服务：

分类	服务	说明
运营平台	车辆信息	提供车、配件、车型等基础信息的接受和查询功能。
	运营服务	提供 T 服务相关功能、亲密 ID、蓝牙钥匙等运营相关功能。如果是管理和运营账户放到此处管理。
	ID 服务	提供对车主账号相关能力，配置信息 ID 中保存。
	数据分析	提供数据统计和分析功能。
	消息运营	为运营消息提供管理功能。通过消息推送服务发送消息。

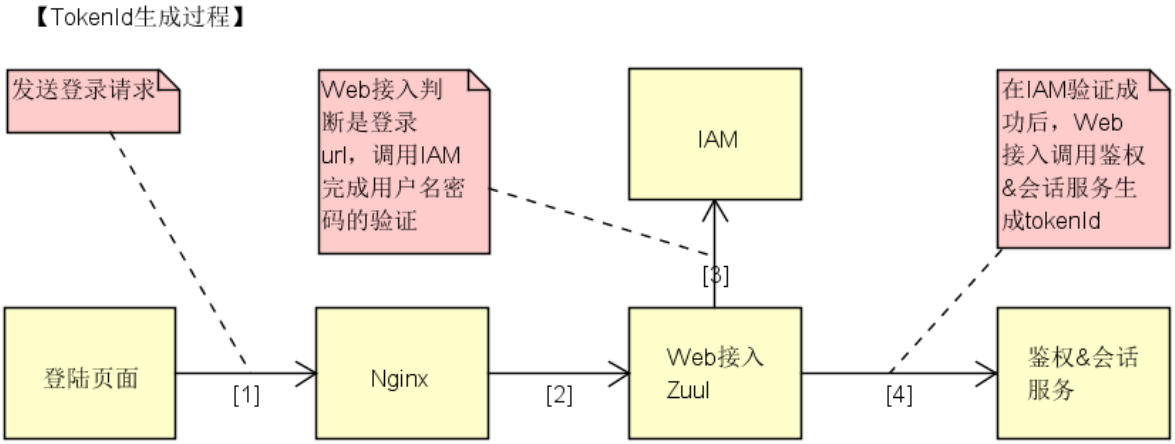
3.10. Web 系统

Web 系统，提供一系列的管理界面，不提供业务逻辑。管理界面对后台服务的调用都经过 Web 接入实现。从目前的功能上来看，Web 系统主要和运营平台交互。在形式上是纯 JavaScript 以及其他静态资源。JS 代码透过 ngnix 和 zuul 调用后台服务。Web 系统本身的功能，例如菜单信息等保存于“运营服务”。所有服务需要的信息都通过 json 传递，不通过 cookie 方式传递。在后续的开发和设计中，如果发现大量需要聚合的服务，可以在 Web 接入后面增加 Web 服务，负责聚合各个服务的返回结果。
左侧菜单静态资源的实现：左侧菜单静态资源加载完毕后，会发起对后台的请求。此请求进过 Nginx->Zuul->运营服务，获得结果。也就是说权限和菜单列表会保存在运营服务中。

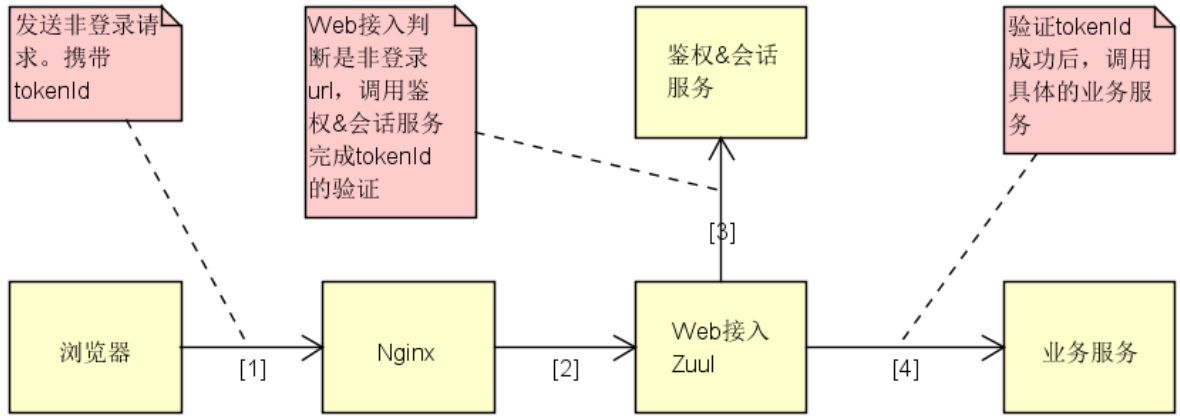
架构图如下：



举例：



【TokenId验证过程】



主要包括 6 个 Web 系统：

分类	服务	说明
Web 系统	Tboss-Web	提供运营管理界面。
	AI-ID-Web	提供 AI-DI 管理界面
	新能源监控	提供监控大屏、监控平台配置功能。
	SP/CP 管理-Web	提供 SP/CP 的管理界面。
	积分商城	提供积分商城的管理和前端界面
	直客车队管理-Web	提供直客车队的管理和前段界面。

JavaScript 跨域问题的解决方法：

1、document.domain+iframe 的设置

2、动态创建 script

3、利用 iframe 和 location.hash

4、window.name 实现的跨域数据传输

5、使用 HTML5 postMessage

前端 JS 框架选型：

vue.js 是一套构建用户界面的 渐进式框架。与其他重量级框架不同的是，Vue 采用自底向上增量开发的设计。Vue 的核心库只关注视图层，并且非常容易学习，非常容易与其它库或已有项目整合。另一方面，Vue 完全有能力驱动采用单文件组件和 Vue 生态系统支持的库开发的复杂单页应用。Vue.js 的目标是通过尽可能简单的 API 实现响应的数据绑定和组合的视图组件。

（1）模块化，目前最热的方式是在项目中直接使用 ES6 的模块化，结合 Webpack 进行项目打包

（2）组件化，创造单个 component 后缀为.vue 的文件，包含 template(html 代码)，script(es6 代码),style(css 样式)

（3）路由，vue 非常小巧，压缩后 min 源码为 72.9kb，gzip 压缩后只有 25.11kb，想比 Angular 为 144kb，可以自驾搭配使用需要的库插件，类似路由插件(Vue-router)，Ajax 插件(vue-resource)等。

在公司层面，车联网事业部有专门的 VUE 架构支持组，提供 VUE 的技术支持。

3.11. 监控系统

监控系统为运维工作提供监控与日志查询界面。

分类	服务	说明
运维监控	系统监控 Zabbix	提供对操作系统的监控能力。通过在目标机器上安装插件收集 cpu、内存、磁盘 IO，网络，进程等信息。
	应用监控 Cat	提供对应用和服务的监控能力。在框架层面将应用的调用，异常等信息传送到 Cat 服务器并进行展示。对业务代码无侵入。
	日志系统 ELK	提供对日志的收集和查询能力。收集应用打印出来的日志到 ELK 服务器，供查询使用。与业务代码无关。

4. 技术架构

4.1. 总体目标

- 高可用性
- 高可扩展性
- 自动化运维

4.2. 设计原则

- 不过度设计
- 抽象化
- 松耦合
- 服务可复用
- 服务可监控
- 水平可扩展
- 使用成熟技术

4.3. 架构图

4.4. 技术选型

本次架构选择微服务架构风格进行，微服务架构是以专注于单一责任的小型功能模块为基础、通过 API 集互相通信的方式完成复杂业务系统搭建的一种设计思想。微服务，关键其实不仅仅是微服务本身，而是系统要提供一套基础的架构，这种架构使得微服务可以独立的部署、运行、升级，不仅如此，这个系统架构还让微服务与微服务之间在结构上“松耦合”，而在功能上则表现为一个统一的整体。这种所谓的“统一的整体”表现出来的是统一风格的界面，统一的权限管理，统一的安全策略，统一的上线过程，统一的日志和审计方法，统一的调度方式，统一的访问入口等等。使用微服务可以让开发效率更高、沟通成本更低、响应速度更快、迭代周期更短。本次我们选择 Spring Boot、Spring Cloud 作为微服务的基础框架进行开发。

Spring Cloud 与 Dubbo 比较：

背景：

Spring Cloud，从命名我们就可以知道，它是 Spring Source 的产物，Spring 社区的强大背书可以说是 Java 企业界最有影响力的组织了，除了 Spring Source 之外，还有 Pivotal 和 Netflix 是其强大的后盾与技术输出。其中 Netflix 开源的整套微服务架构套件是 Spring Cloud 的核心。

Dubbo，是阿里巴巴服务化治理的核心框架，并被广泛应用于阿里巴巴集团的各成员站点。

社区活跃度：

我们选择一个开源框架，社区的活跃度是我们极为关注的一个要点。社区越活跃，解决问题的速度越快，框架也会越来越完善，不然当我们碰到问题，就不得不自己解决。而对于团队来说，也就意味着我们不得不自己去维护框架的源码，这对于团队来说也将会是一个很大的负担。

在社区活跃度上，Spring Cloud 毋庸置疑的优于 Dubbo，这对于没有大量精力与财力维护这部分开源内容的团队来说，Spring Cloud 会是更优的选择。

架构完整度：

下表列举了一些核心部件，大致可以理解为什么之前说 Dubbo 只是类似 Netflix 的一个子集了吧。当然这里需要申明一点，Dubbo 对于上表中总结为“无”的组件不代表不能实现，而只是 Dubbo 框架自身不提供，需要另外整合以实现对应的功能。

	Dubbo	Spring Cloud
服务注册中心	Zookeeper	Spring Cloud Consul
服务调用方式	RPC	Spring Cloud Feign
服务网关	无	Spring Cloud Zuul
断路器	不完善	Spring Cloud Hystrix
分布式配置	无	Spring Cloud Config
客户端负载均衡	无	Spring Cloud Ribbon
消息总线	无	Spring Cloud Bus
消息驱动	无	Spring Cloud Stream
服务跟踪	无	Spring Cloud Sleuth

文档质量：

虽然 Spring Cloud 的文档量大，但是如果使用 Dubbo 去整合其他第三方组件，实际也是要去阅读大量第三方组件文档的，所以在文档量上，我觉得区别不大。对于文档质量，由于 Spring Cloud 的迭代很快，难免会出现不一致的情况，所以在质量上我认为 Dubbo 更好一些。而对于文档语言上，Dubbo 自然对国内开发团队来说更有优势。

基于以上四方面的比较综合考虑，我们选择 Spring Cloud 作为我们本次微服务开发的基础框架。

4.5. Spring Boot

Spring Boot 是伴随着 Spring4.0 诞生的；从字面理解，Boot 是引导的意思，因此 Spring Boot 帮助开发者快速搭建 Spring 框架；Spring Boot 帮助开发者快速启动一个 Web 容器；Spring Boot 继承了原有 Spring 框架的优秀基因；Spring Boot 简化了使用 Spring 的过程。

4.6. Spring Cloud

Spring Cloud 是一个集成了众多开源的框架，利用 Spring Boot 的开发便利性实现了服务治理、服务注册与发现、负载均衡、数据监控，REST API 发布方式等，基本囊括了分布式框架所需要的所有功能。是一套易开放、易部署、易维护的分布式开发工具包。

4.7. Consul

Consul 是 HashiCorp 公司推出的开源工具，用于实现分布式系统的服务发现与配置。与其他分布式服务注册与发现的方案，Consul 的方案更“一站式”，内置了服务注册与发现框 架、具有以下性质：分布一致性协议实现、健康检查、Key/Value 存储、多数据中心方案。

本次使用的是 Spring Cloud Consul 组件，下表为 Consul 与 Eureka 的对比表格。

Feature	Consul	Eureka
服务健康检查	服务状态，内存，硬盘等	可配支持

多数据中心	支持	—
kv 存储服务	支持	—
一致性	raft	—
cap	ca	ap
使用接口(多语言能力)	支持 http 和 dns	http（sidecar）
watch 支持	全量/支持 long polling	支持 long polling/大部分增量
自身监控	metrics	metrics
安全	acl /https	—
spring cloud 集成	已支持	已支持

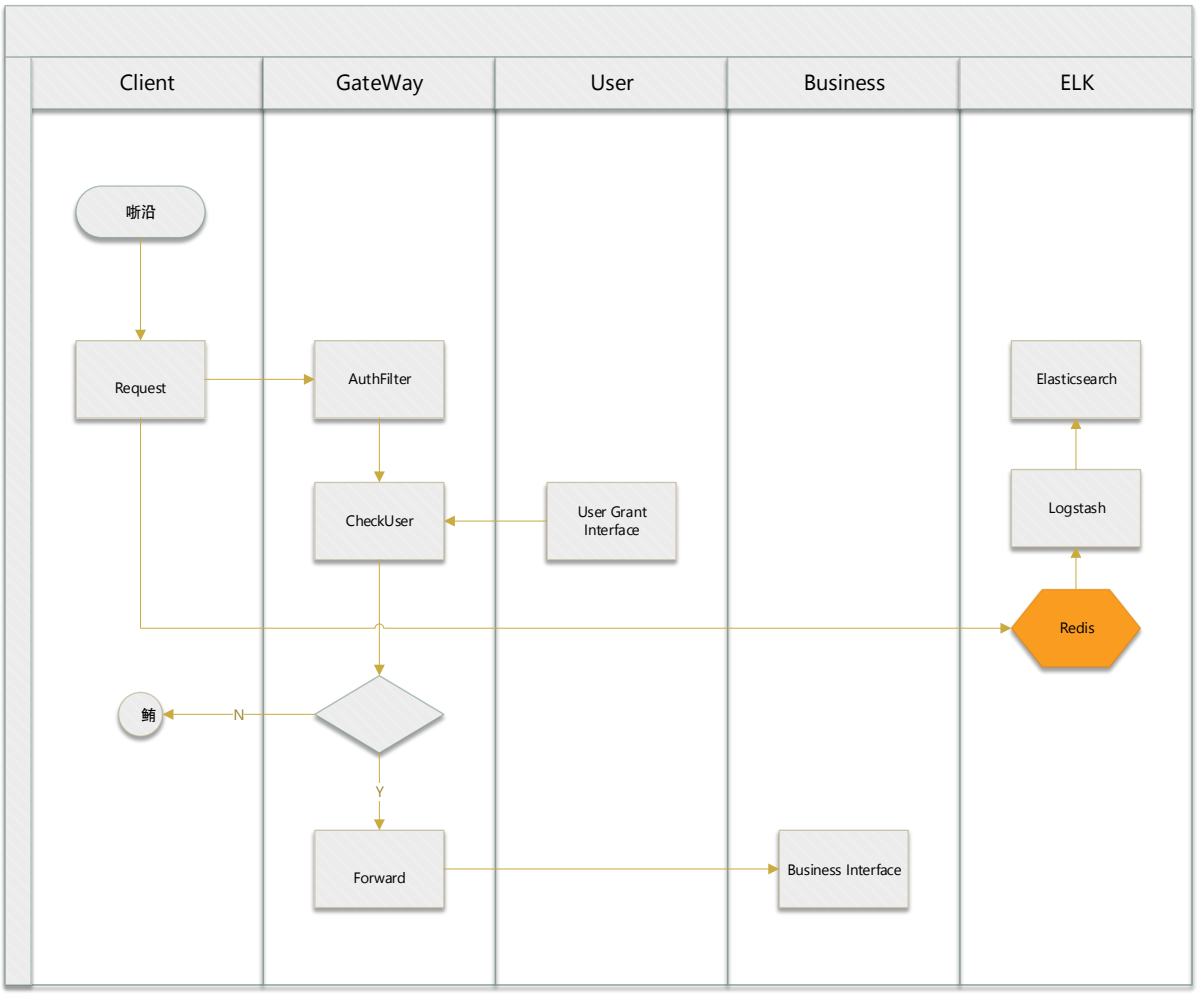
平台选择 Consul 的理由：

- 适合混合云部署
- 适合与 Docker 和 K8S 结合
- 适合灰度部署

4.8. Zuul

Zuul 是 Netflix 出品的一个基于 JVM 路由和服务端的负载均衡器。它具有认证压力测试、金丝雀测试、动态路由、负载削减、安全、静态响应处理、主动/主动交换管理等功能。

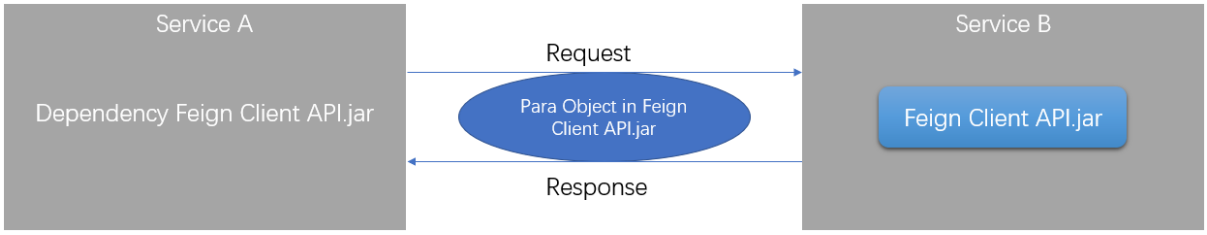
Zuul 网关的权限控制示例图：



4.9. Feign

Feign 是一个声明 web 服务客户端，这便得编写 web 服务客户端更容易，使用 Feign 创建一个接口并对它进行注解，它具有可插拔的注解支持包括 Feign 注解与 JAX-RS 注解，Feign 还支持可插拔的编码器与解码器，Spring Cloud 增加了对 Spring MVC 的注解，Spring Web 默认使用了 HttpMessageConverters，Spring Cloud 集成 Ribbon 和 Eureka 提供的负载均衡的 HTTP 客户端 Feign。

Feign 调用关系示例图：



调用关系说明：

1. Service A 调用 Service B 开放的接口时，使用 Feign Client 进行调用。
2. Service A 依赖 Service B 提供的接口 jar，jar 中包含指定接口以及接收参数等。
3. Service A 调用 API 对 Service B 进行服务访问。

4.10. Ribbon

Spring Cloud Ribbon 是一个基于 Http 和 TCP 的客服端负载均衡工具，它是基于 Netflix Ribbon 实现的。它不像服务注册中心、配置中心、API 网关那样独立部署，但是它几乎存在于每个微服务的基础设施中。

针对 Ribbon 架构没有另行封装或重写，针对 Zuul 网关参照 5.8 的相关配置；针对各个微服务而言，通过 Feign 组件中的相关 Ribbon 配置实现重试等相关操作。

4.11. Hystrix

Hystrix 的中文含义是豪猪，因其背上长满了刺，而拥有自我保护能力。Netflix 的 Hystrix 是一个帮助解决分布式系统交互时超时处理和容错的类库，它同样拥有保护系统的能力。Hystrix 的设计原则包括：资源隔离、熔断器和命令模式。

针对 Hystrix 架构没有另行封装或重写，针对 Zuul 网关参照 5.8 的相关配置；针对各个微服务而言，通过 Feign 组件中的相关 Hystrix 配置实现重试等相关操作。

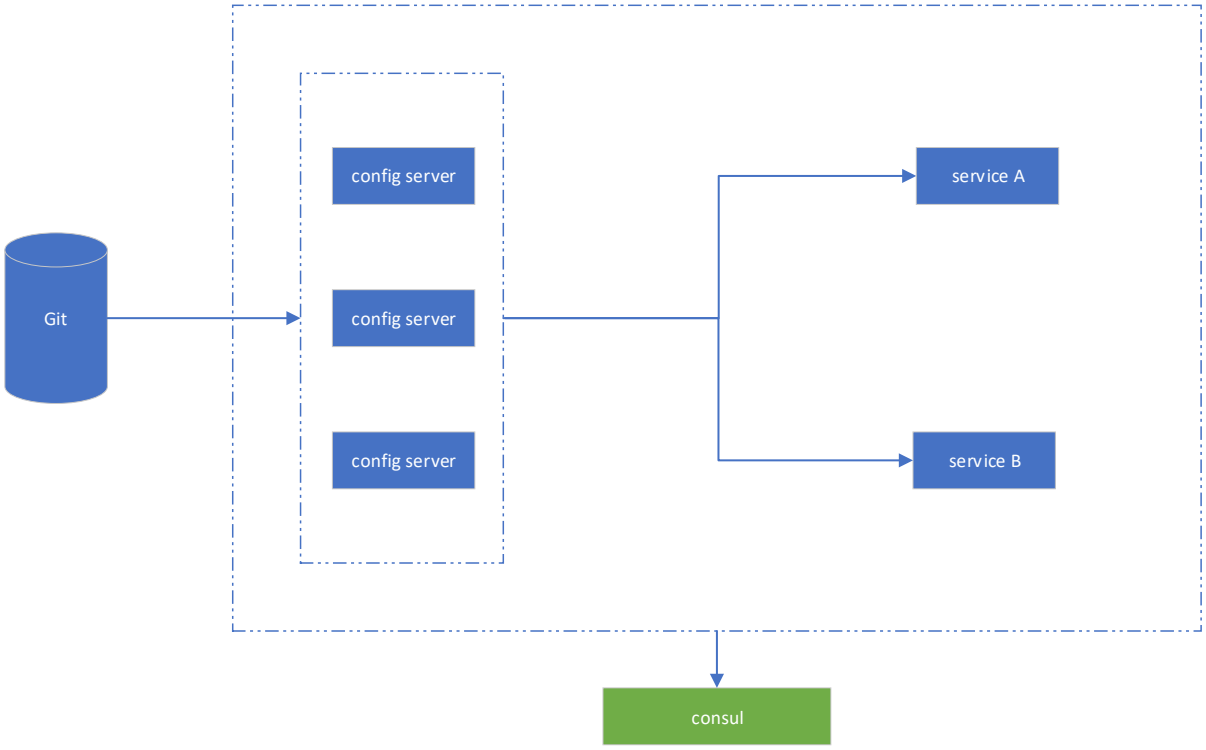
关于 Hystrix 的请求缓存和请求合并功能，需要根据业务继承 HystrixCommand 实现，在架构层不做过多的封装。

4.12. Stream

Spring Cloud Stream 是一个用来为微服务应用构建消息驱动能力的框架。它可以基于 Spring Boot 来创建独立的，可用于生产的 Spring 应用程序。他通过使用 Spring Integration 来连接消息代理中间件以实现消息事件驱动。Spring Cloud Stream 为一些供应商的消息中间件产品提供了个性化的自动化配置实现，引用了发布-订阅、消费组、分区的三个核心概念。

4.13. Spring Config

在分布式系统中，由于服务数量巨多，为了方便服务配置文件统一管理，实时更新，所以需要分布式配置中心组件。在 Spring Cloud 中，有分布式配置中心组件 spring cloud config ，它支持配置服务放在配置服务的内存中（即本地），也支持放在远程 Git 仓库中。在 spring cloud config 组件中，分两个角色，一是 config server，二是 config client。



Spring Config Server 相关配置：

```
spring.cloud.config.server.git.uri：配置 git 仓库地址
spring.cloud.config.server.git.searchPaths：配置仓库路径
spring.cloud.config.label：配置仓库的分支
spring.cloud.config.server.git.username：访问 git 仓库的用户名
spring.cloud.config.server.git.password：访问 git 仓库的用户密码
```

Spring Config Client 相关配置：

```
spring.cloud.config.label=git 对应分支
spring.cloud.config.profile=dev
spring.cloud.config.uri=配置服务中心地址
```

关于配置文件中数据的安全性，Spring Cloud 具有用于在本地解密属性值的环境预处理器。 它遵循与 Config Server 相同的规则，并通过加密具有相同的外部配置。 因此，您可以使用{cipher}*形式的加密值，只要有一个有效的密钥，那么在主应用程序上下文获取环境之前，它们将被解密。 要在应用程序中使用加密功能，您需要在您的类路径中包含 Spring Security RSA（Maven 协调“org.springframework.security:spring-security-rsa”），并且还需要 JVM 中强大的 JCE 扩展。

4.14. CAT

CAT 是大众点评开源的一套基于 java 的实时应用监控平台，主要应用于服务中间件框架（MVC 框架、RPC 框架、持久层框架、分布式缓存框架）的监控，为开发和运维提供各项性能指标、健康检查、自动报警等可视化服务。

CAT 能做什么：

- 1. 机器状态信息。CPU 负载、内存信息、磁盘使用率这些是必需的，另外可能还希望收集 Java 进程的数据，例如线程栈、堆、垃圾回收等信息，以帮助出现问题时快速 debug。
- 2. 请求访问情况。例如请求个数、响应时间、处理状态，如果有处理过程中的时间分析那就更完美了。
- 3. 异常情况。譬如缓存服务时不时出现无响应，我们希望能够监控到这种异常，从而做进一步的处理。

4. 业务情况。针对具体业务进行统计报表等等。

CAT 集成方式：

需要根据业务情况分析，在业务代码级调用 CAT 的 SDK 进行埋点，并配置到指定的 CAT Server 端。

4.15. ELK

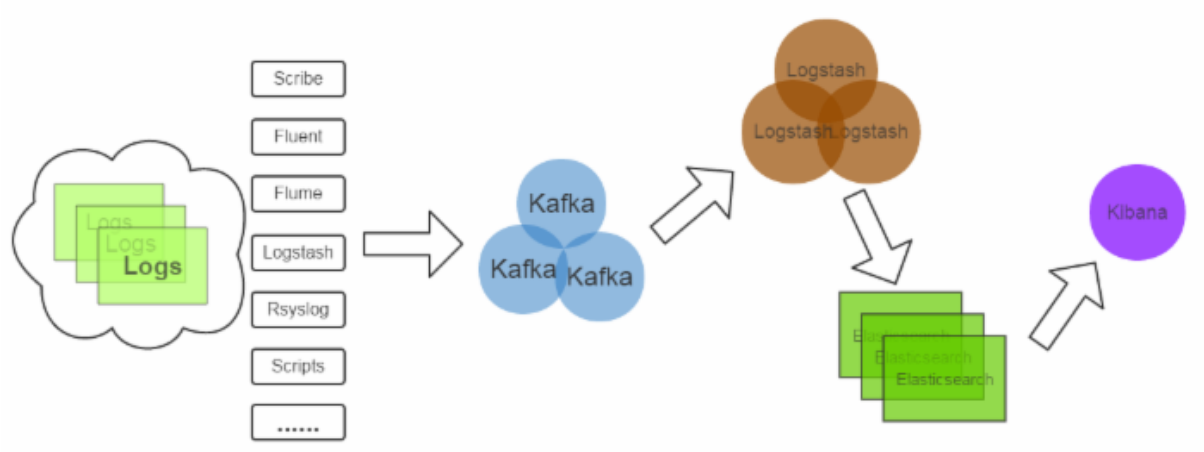
ELK 由 Elasticsearch、Logstash 和 Kibana 三部分组件组成。

Elasticsearch 是个开源分布式搜索引擎，它的特点有：分布式，零配置，自动发现，索引自动分片，索引副本机制，restful 风格接口，多数据源，自动搜索负载等。

Logstash 是一个完全开源的工具，它可以对你的日志进行收集、分析，并将其存储供以后使用。

kibana 是一个开源和免费的工具，它可以为 Logstash 和 ElasticSearch 提供的日志分析友好的 Web 界面，可以帮助您汇总、分析和搜索重要数据日志。

ELK+Kafka 集成的日志系统：



ELK 能做什么：

1. 分布式日志数据集中式查询和管理
2. 系统监控，包含系统硬件和应用各个组件的监控
3. 故障排查
4. 安全信息和事件管理
5. 报表功能

ELK 的接入方式：

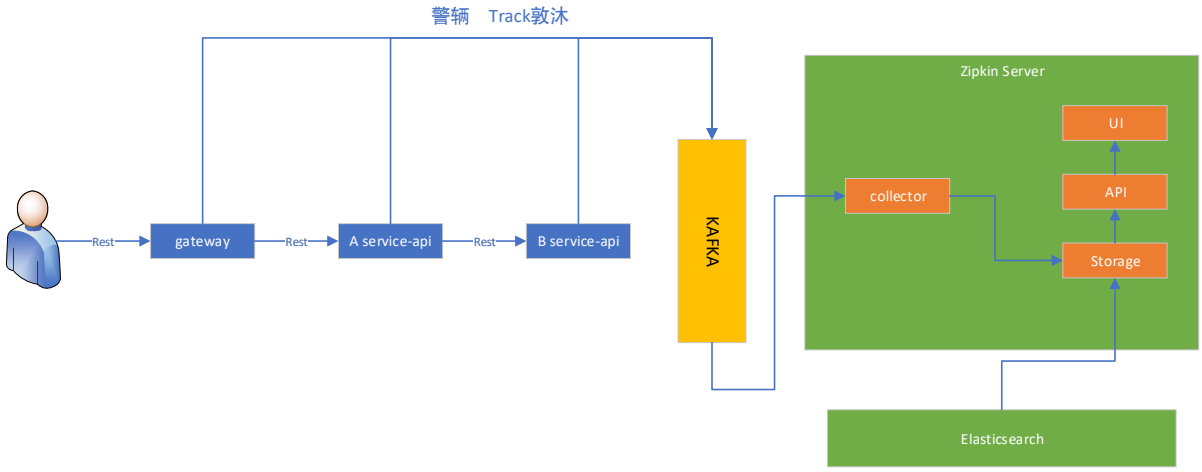
1. 通过 Logback 将日志发布到 Kafka 中
2. ELK 去 Kafka 消费数据并进行存储和展示

4.16. Sleuth + Zipkin

什么是服务追踪 Sleuth
在微服务架构中，要完成一个功能，通过 Rest 请求服务 API 调用服务来完成，整个调用过程可能会聚合多个后台服务器协同完成。在整个链路上，任何一处调用超时
或出错都有可能造成前端请求失败。这时跟踪记录这些请求的调用的情况就要复杂的多，这就需要一个专门的工具来处理，spring cloud sleuth 组件就是用于跟踪记录的工具
Sleuth 就相当于为微服务架构引入了一套记录体系，包含两部分，一个是 trace ID;另一个是 span ID，随时记录一个请求的每一步操作。

什么是日志聚合 Zipkin
zipkin 是 Dpper 的开源实现，支持多种语言。Sleuth 已经将每个请求从开始调用到完成的每一步都进行了记录，但是这些 log 信息会很分散，使用起来不太方便，就
需要有一个工具可以将这些信息进行收集和汇总，并且显示可视化的结果，便于分析和定位。这就需要创建一个 Zipkin Server 用于收集和展示这些调用链路的信息，他
的使用也很方便。

调用链流程图:

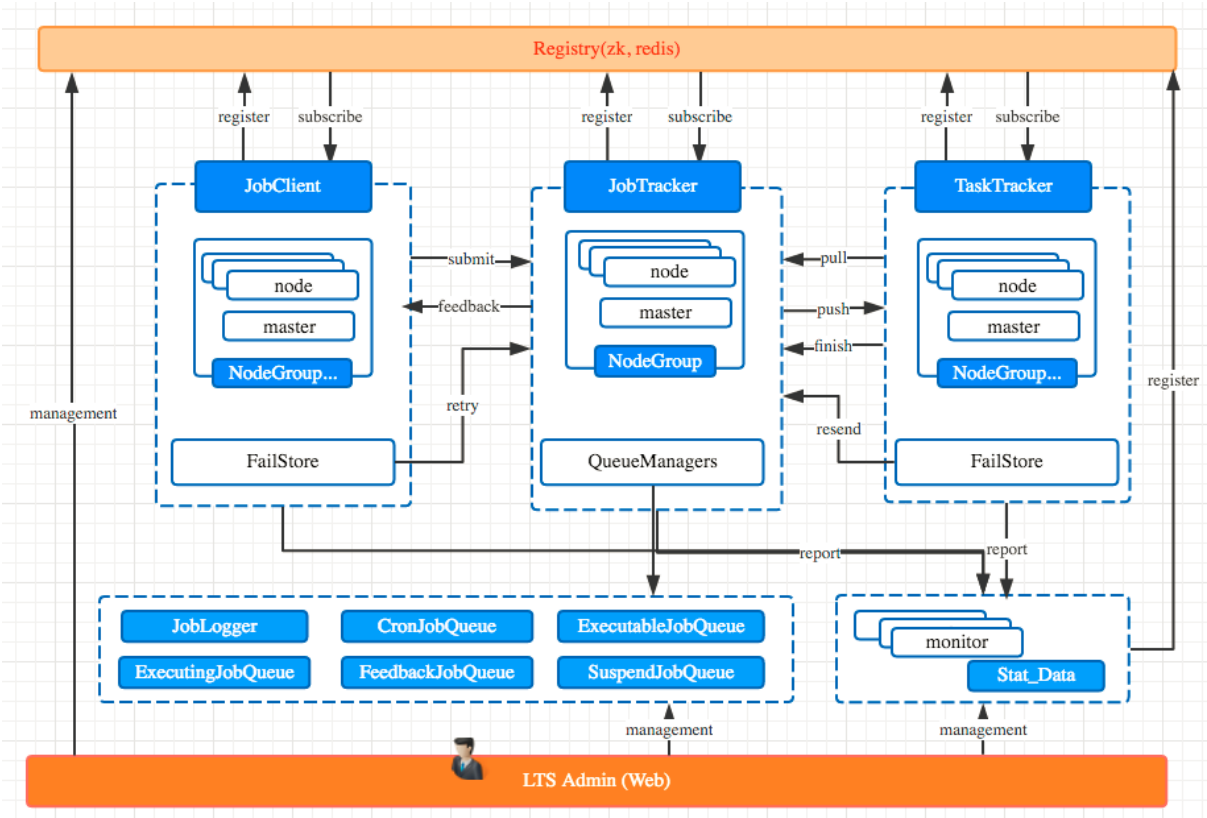


4.17. LTS

LTS(light-task-scheduler)主要用于解决分布式任务调度问题，支持实时任务，定时任务和 Cron 任务。有较好的伸缩性，扩展性，健壮性。

LTS 有主要有以下四种节点：
JobClient：主要负责提交任务，并接收任务执行反馈结果。
JobTracker：负责接收并分配任务，任务调度。
TaskTracker：负责执行任务，执行完反馈给 JobTracker。
LTS-Admin：（管理后台）主要负责节点管理，任务队列管理，监控管理等。

LTS 架构图如下：



4.18. Storm

Storm 是一个免费并开源的分布式实时计算系统。利用 Storm 可以很容易做到可靠地处理无限的数据流，像 Hadoop 批量处理大数据一样，Storm 可以实时处理数据。Storm 简单，可以使用任何编程语言。

Storm 有如下特点：

1. 低延迟和高性能

在一个小集群中，每个节点每秒可以处理数以百万计的消息。

2. 可扩展

在 Storm 集群中主要有三个实体：工作进程、线程和任务。Storm 集群中每台机器上都可以运行多个工作进程，每个工作进程又可以创建多个线程，每个线程可以执行多个任务，任务是真正进行数据处理的实体。

3. 高可靠性

Storm 可以保证 Spout 发出的每条消息都能被完全处理，Spout 发出的消息后续可能会触发产生成千上万条消息，可以形象的理解为一棵消息树，只有当这颗消息树中的所有消息都被处理了才叫“完全处理”，这种特殊的策略会在后面详细介绍。

4. 高容错性

如果消息在处理过程中出现了一些异常，Storm 会重新部署这个出问题的处理单元，Storm 保证一个处理单元永远运行（除非你显式结束这个处理单元）

5. 编程模型简单

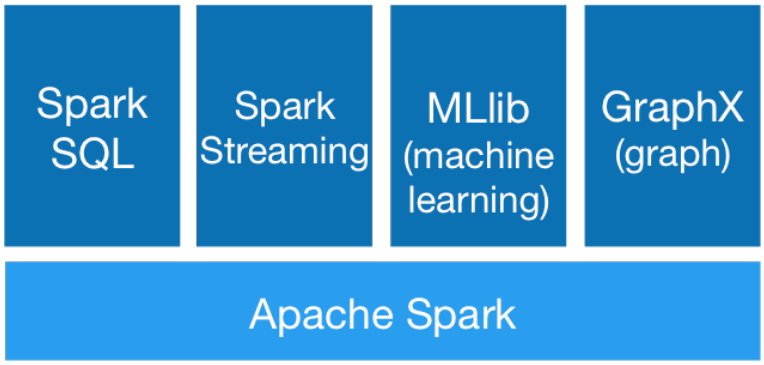
Storm 为大数据的实时计算提供了一些简单优美的原语，大大降低了开发并行实时处理任务的复杂

性。
6. 支持多种编程语言
7. 支持本地模式

4.19. Spark

Spark 是基于内存计算的大数据并行计算框架。Spark 基于内存计算，提高了在大数据环境下数据处理的实时性，同时保证了高容错性和高可伸缩性，允许用户将 Spark 部署在大量廉价硬件之上，形成集群。

Spark 特点：
1. 更快的速度
内存计算下，Spark 比 Hadoop 快 100 倍。
2. 易用性
Spark 提供了 80 多个高级运算符。
3. 通用性
Spark 提供了大量的库，包括 SQL、DataFrames、MLlib、GraphX、Spark Streaming。 开发者可以在同一个应用程序中无缝组合使用这些库。



4. 支持多种资源管理器
Spark 支持 Hadoop YARN，Apache Mesos，及其自带的独立集群管理器

5. 非功能设计

5.1. 性能

影响系统性能主要瓶颈在 I/O，包括数据库，socket，网络通信，文件等，

具体可用的设计策略有:
缓存以及缓存层设计
多线程并发设计
负载平衡结构设计
数据库优化设计
文件系统优化设计

代码性能设计

5.2. 可靠性

通过分布式集群结构的设计，保证系统的可靠性。
通过 Hystrix、Ribbon 等中间件对系统进行熔断、重试等处理，保证系统出的可靠性。

5.3. 可扩展性

使用微服务服务注册发现机制
使用分布式的系统设计
使用无状态的设计思想

5.4. 可维护

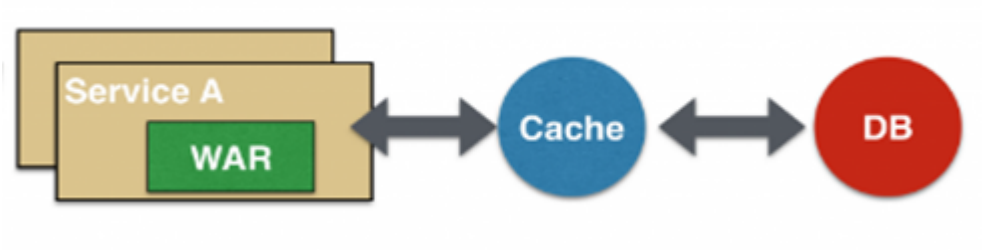
结合 ELK、CAT、Zipkin 等中间件等的监控功能，对系统进行 24 小时的监控、跟踪及报警，并结合服务的注册发现机制随时动态进行服务的治理。

6. 数据存储设计

6.1. 数据分类与存储方式

数据分类	存储方式
车型，用户，配置等基础数据	关系数据库 MySql
车况数据	非关系数据库 HBase
二进制数据。例如用户头像等	分布式文件系统 FastDFS
频繁读取数据	缓存 Redis

6.2. 服务的数据存取架构

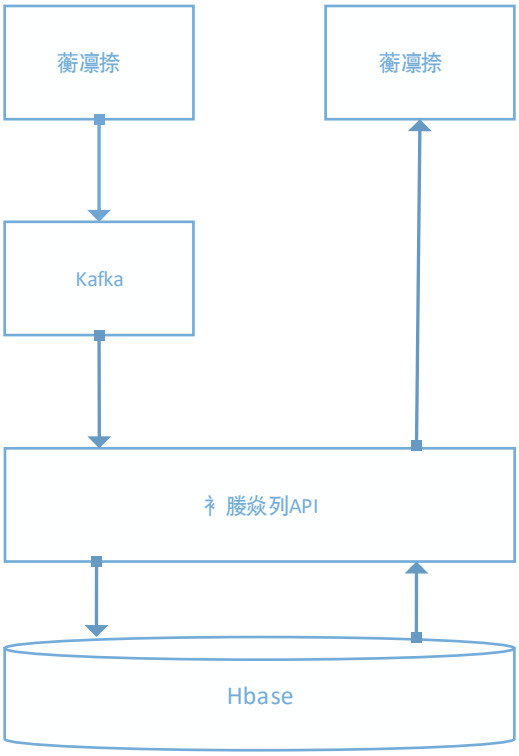


对于大部分服务来存取架构如上图所示：

- 数据读取：服务会先到缓存中查看数据是否被缓存。如果有，则从缓存获得数据，如果没有，则访问 DB，并将数据缓存。
- 数据写入：会将数据写入到 DB 中，同时清除对应的缓存数据。

7.3 HBase 存储

HBase 适用于频繁读写，数据量大，实时要求较高的应用场景，在车联网场景中 HBase 存储主要用来存储车况信息，车俩轨迹等数据，用于车俩诊断，数据统计等。主要涉及 HDFS，HBase，Kafka，其中，HDFS 用于 HBase 的底层的存储，HBase 用于数据的读写，Kafka 用来数据的采集和接收。架构图如下：



数据写入：平台层调用 kafka 生产者接口，把数据存入 kafka 的 brokers 中，HBase 服务端，通过 kafka 消费者接口，通过主题 Topic 订阅消费相应的数据，然后调用数据接口 API 把数据插入 Hbase 中。

数据读取：平台应用通过数据接口 API 查询数据，数据接口 API 对外提供数据查询，数据更新

等操作，其中数据查询包括指定 rowkey，rowkey 范围，和 scan 等。