# 微服务框架技术选型

## 微服务框架常见使用场景

微服务是一种软件架构风格，它将一个大型应用程序拆分成一组小型、独立的服务，每个服务都可以独立部署、扩展和维护。每个服务都有自己的业务逻辑和数据存储，它们通过轻量级的通信机制（如REST API）相互通信。

对于企业来说，微服务架构可以带来以下价值：

1. 更快的应用程序开发和部署：由于每个微服务都是独立的，开发人员可以同时工作，加快了应用程序的开发和部署速度。
2. 更高的可扩展性和可维护性：微服务架构可以让企业更容易地扩展和维护应用程序，因为每个微服务都是独立的，可以单独扩展或维护，而不会影响整个应用程序。
3. 更好的容错性和可靠性：微服务架构可以让企业更容易地处理故障，因为每个微服务都是独立的，可以在一个服务出现故障时继续运行其他服务。
4. 更好的安全性：微服务架构可以让企业更容易地实现安全措施，因为每个微服务都可以有自己的安全措施，如身份验证、授权等。
5. 技术多样性：微服务架构可以让企业使用不同的技术栈和编程语言来开发不同的服务，这可以根据不同的需求和场景来选择最合适的技术。

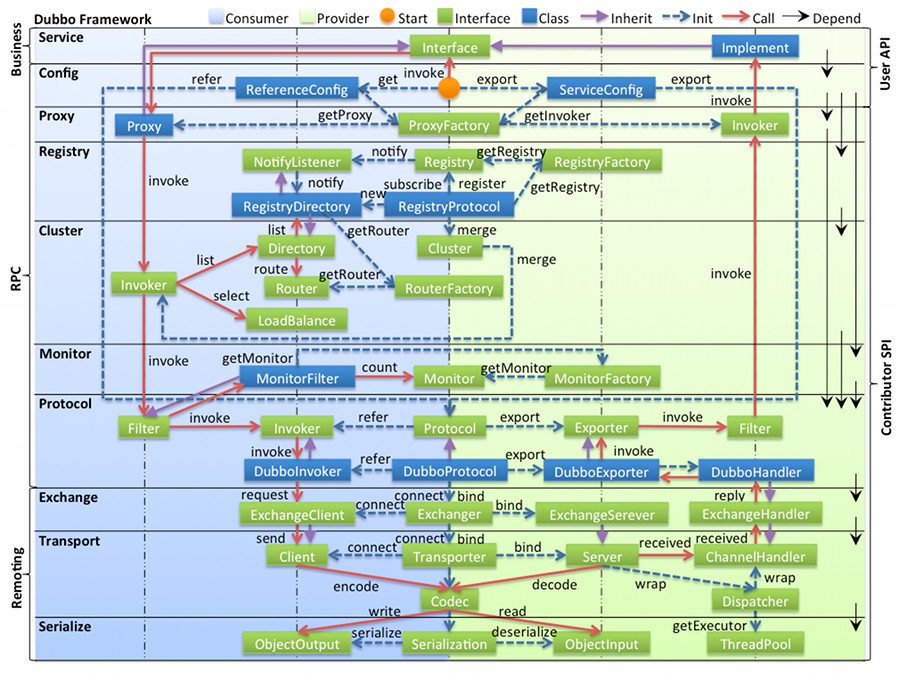
总之，微服务架构可以帮助企业更快、更容易、更可靠地开发和维护应用程序，提高业务效率和客户满意度。

微服务技术体系目前已经是大家开发分布式系统的首选方案，但是常见的微服务框架都是通用的各类单独技术工具，如何整合这些不同的框架，形成一套完善的微服务技术平台，并且有效的支持银行复杂的业务落地，是微服务技术选型的最重要考量点。

## 主流微服务框架

1. **Dubbo**

1）架构图

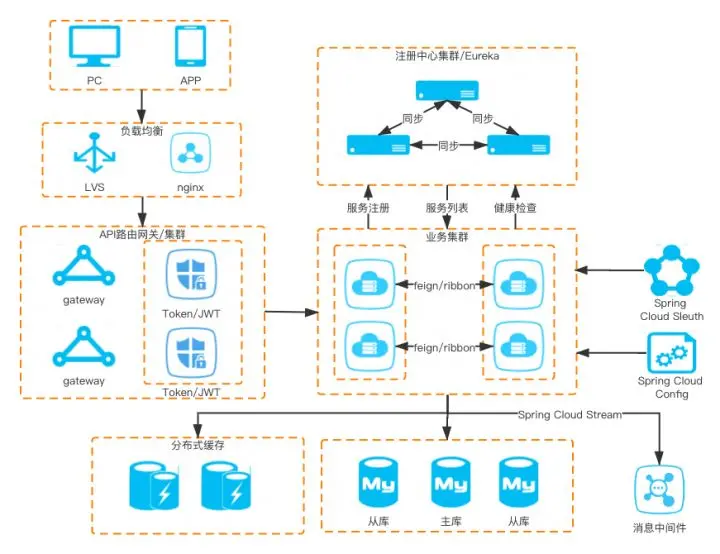


1. 核心功能介绍

[Dubbo](https://so.csdn.net/so/search?q=Dubbo&spm=1001.2101.3001.7020" \t "https://blog.csdn.net/ChenRui_yz/article/details/_blank)是一款Java RPC框架，致力于提供高性能的RPC远程服务调用方案。Dubbo 作为主流的微服务框架之一，为开发人员带来了非常多的便利。

* 高性能RPC调用：提供高性能的基于代理的远程调用能力，为开发者屏蔽远程调用底层细节
* 智能负载均衡：内置多种负载均衡策略，智能感知下游节点健康状况，显著减少调用延迟，提高系统吞吐量
* 服务自动注册与发现：支持多种注册中心服务、服务实例上下线实时感知
* 高度可扩展能力：遵循微内核+插件的设计原则，所有核心能力均被设计为扩展点，平等对待内置实现和三方实现
* 运行期流量调度：内置条件、脚本等路由策略，通过配置不同的路由规则轻松实现灰度发布、同机房优先等功能
* 可视化的服务治理与运维：提供丰富的服务治理、运维工具，随时查询服务元数据、服务健康状态及调用统计，实时下发路由策略，调整配置参数

1. **SpringCloud**
2. 架构图



1. 组件介绍

Spring Cloud是一种最常见的分布式系统模式，为开发者提供了一种简单并且易于接受的编程模型，帮助开发人员构建有弹性的、可靠的、协调的程序软件，轻松实现微服务项目的构建。

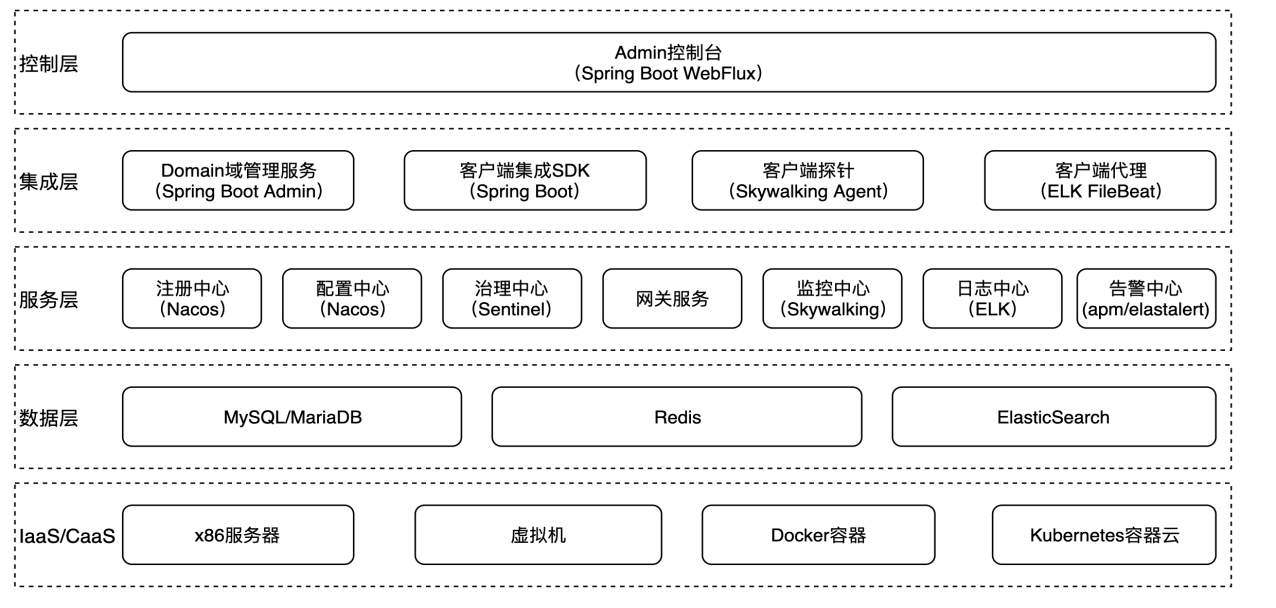
* SpringCloudGateway：网关路由服务，提供代理访问转发，保证服务安全
* SpringCloudEureka：注册中心服务，提供服务注册与发现能力
* SpringCloudFeign：服务代理，提供代理调用服务
* SpringCloudRibbon：负载均衡器，提供多种负载均衡算法
* SpringCloudConfig：服务配置，提供统一配置功能
* SpringCloudSleuth：分布式链路追踪，将一次分布式请求还原成调用链路，进行日志记录，性能监控并将一次分布式请求的调用情况集中展示。
* SpringCloudStream：为微服务应用构建消息驱动能力的框架

1. **EDSP（长亮科技企业级分布式服务平台）**
2. 核心功能介绍

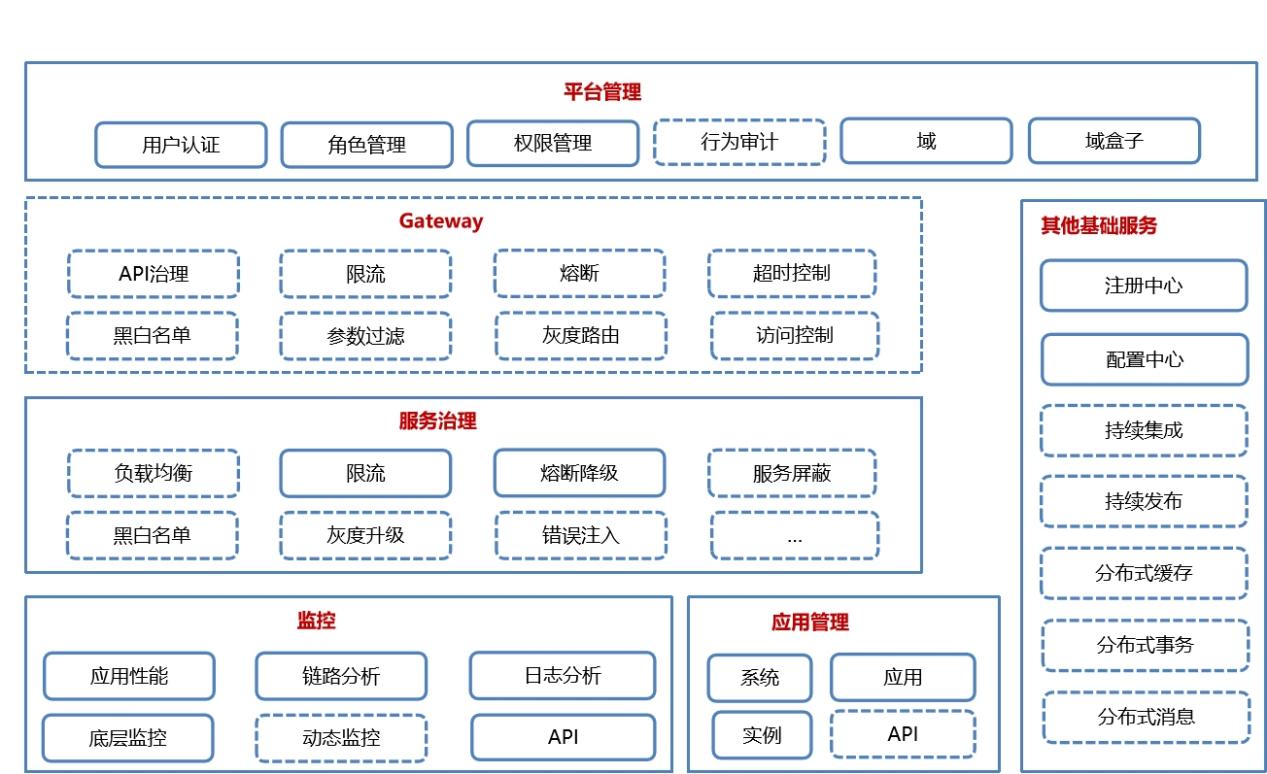
长亮科技微服务框架（EDSP）是一个围绕微服务应用的gPaaS平台，提供应用全生命周期管理、数字化运维、立体化监控、细粒度服务治理等功能；拥抱Spring Boot和Spring Cloud开源社区，帮助企业客户解决传统集中式架构转型的困难，打造大规模高可用的金融级分布式系统架构，实现业务和产品的快速落地。

* 拥抱开源：拥抱Spring Boot/Spring Cloud开源社区，提供高可用、可扩展、灵活的微服务技术中台架构支持，降低用户迁移到微服务架构的时间和人力成本。
* 全生命周期管理：提供从创建应用到运行应用的全生命周期管理，支持创建、部署、扩容、回滚、下线、启动和停止，支持虚拟机和容器两种模式，满足不同的客户使用需求。
* 细粒度服务治理：提供服务和API级别的服务治理能力，支持控制台上配置服务路由、限流策略、鉴权规则等，支持分布式配置管理。
* 全面的基础组件：集成了注册中心、配置中心、治理中心、日志中心、链路中心、分布式缓存、分布式消息队列、分布式锁、流量网关、治理网关等微服务架构所需的基础组件，并对应用层保持透明性，解决企业级架构管理问题。
* 全方位运维管理：支持应用和组件级监控告警、应用和组件级日志分析、交易链路和依赖拓扑图等多种维度的运维管理，满足不同维度的运维需求。
* 两地三中心：支持同城跨可用域容灾和就近路由，规避单可用域可能存在的不可抗力风险，提高服务的可用性和容灾能力。

1. 组件架构



1. 功能架构



1. **SOFAStack**

SOFAStack™（Scalable Open Financial Architecture Stack）是一套用于快速构建金融级云原生架构的中间件，也是在金融场景里锤炼出来的最佳实践。

SOFAStack 包含了构建微服务体系的众多组件，包括研发框架、RPC 框架，服务注册中心，分布式链路追踪，Metrics监控度量、分布式事务框架、服务治理平台等，结合社区优秀的开源产品，可以快速搭建一套完善的微服务体系。兼容Dubbo、Spring Cloud等微服务运行环境，助力客户各类应用轻松转型分布式架构。

1. 产品架构



1. 产品功能

2.1中间件

满足传统金融架构的平滑迁移、融合适配，以稳妥应对业务升级变更、服务和数据扩展性、事务一致性、秒级容灾、弹性供给与调度等关键技术挑战。

* 双模微服务

基于SDK和服务网格微服务架构，构建高可扩展、高性能、低成本、轻量无侵入分布式系统。

* 消息队列

基于Apache RocketMQ 构建，提供异步解耦、削峰填谷等能力。

* 分布式事务

蚂蚁金服自主研发的金融级分布式事务中间件，保障在大规模分布式环境下业务活动的最终一致性。

* 任务调度

实现任务的分布式处理，并能规范化、自动化、可视化和集中化对金融企业不同业务系统的任务进行统一的调度和全方位监控运维管理。

* API网关

帮助企业统一管理对内外开放的 API，为网络隔离的系统间提供高性能、高安全、高可靠的通信，同时保障内部系统的安全性。

* 数据访问代理

金融级分布式数据库中间件，提供水平拆分、平滑扩缩容、读写分离在线服务。

2.2运维管理

SOFAStack CAFE（Cloud Application Fabric Engine）云应用引擎，提供应用管理、发布部署、运维编排、监控分析、容灾应急等全生命周期管理的PaaS平台能力，满足金融场景中经典和云原生架构的运维需求，帮助传统架构平滑过渡、保障金融技术风险 。

* 容器应用服务

兼顾标准化K8S能力的同时，提供源自蚂蚁实践的全生命周期应用发布管控能力。

* 业务实时监控

帮助运维、研发、SRE （Site Reliability Engineer）等快速地发现问题、定位问题、分析问题、解决问题，为线上系统可用率提供有效保障。

* 经典应用服务

提供基于VM模式发布管控，满足数据中心级别的应用变更需求和灰度引流能力。

2.3研发能效

源于蚂蚁金服互联网金融领域研发背景和工程实践的深厚沉淀，为行业数字化转型客户提供金融级一站式智能研发平台，提供敏捷交付和稳妥创新兼顾的研发交付、风险防控和质量保障等能力，赋能金融产品高可用和研发效能的持续提升。

* 持续交付

基于 Gitflow 工作流的最佳实践，通过组件灵活编排，轻松实现代码扫描、代码评审、自动化测试、自动编译部署等核心功能，以持续交付实践不断提高研发效率。

* 项目协作

简洁、易用、整合的一站式研发协作平台，以项目为切入点，多维度集成管理需求、任务、缺陷、迭代、看板，提供轻量级支持敏捷开发等多种项目管理的实践方法。

1. 应用场景

互联网金融中台

基于蚂蚁集团中台战略及架构的最佳实践，将企业级公共能力进行抽象，形成以客户服务、运营服务、分布式架构为基础的业务中台体系，实现开放、可扩展、组件化、分布式的业务架构，支持业务快速、高效、低成本创新，满足互联网场景化快速多变的业务发展需求。



金融级云原生应用

蚂蚁集团自主研发的金融级分布式架构平台，专注为金融用户提供全栈式的基础架构能力，保证风险安全的同时帮助业务需求敏捷迭代，同时满足异地容灾、低成本快速扩容的需求，解决传统集中式架构转型的困难，打造大规模高可用分布式系统架构，支撑金融业务创新。



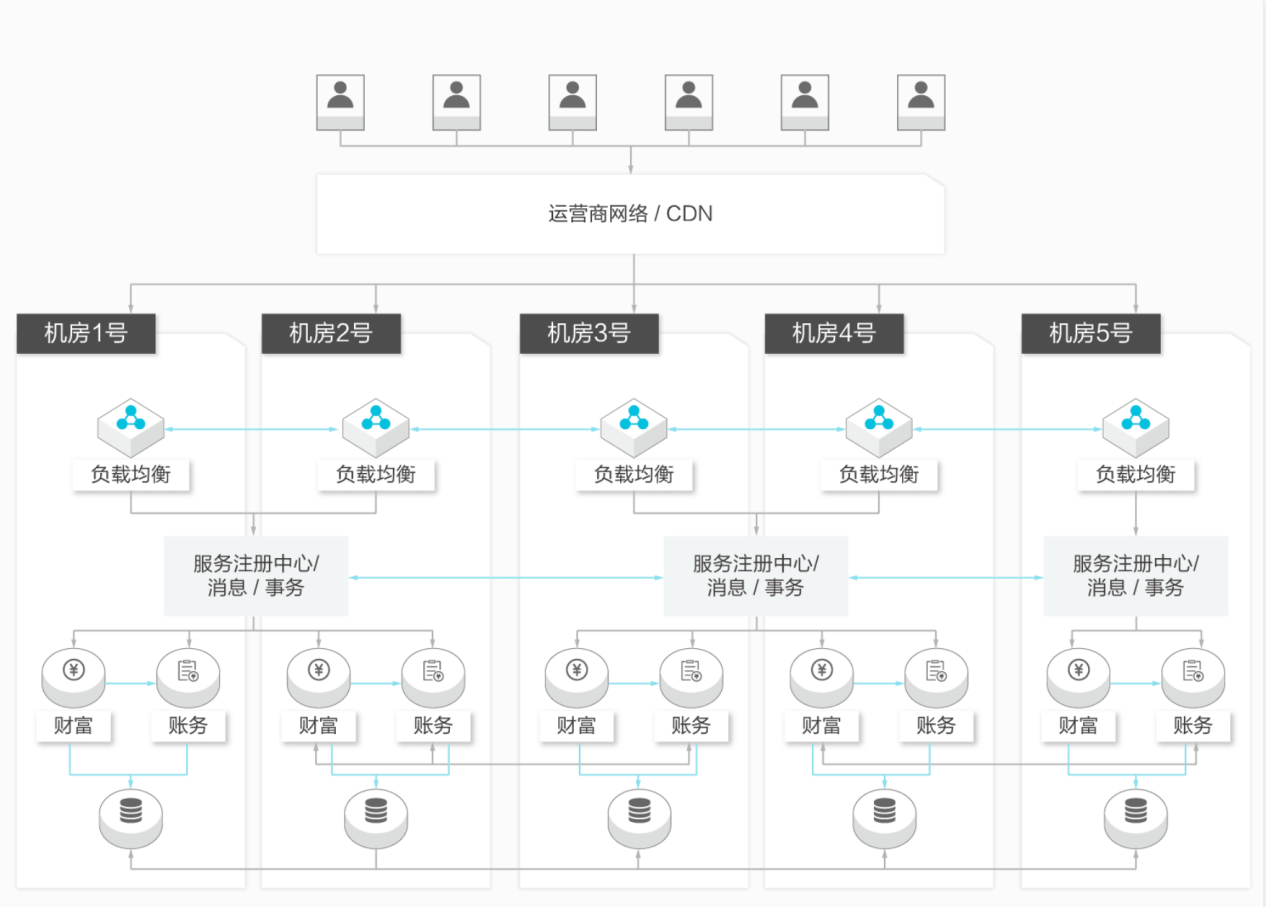
异构应用融合迁移

微服务平台通过 SOFA 微服务和 Service Mesh 微服务，提供了既支持 SOFA 框架又支持 Service Mesh 架构的微服务管理和治理能力，解决用户在技术转型期间与未改造的遗留系统相互之间打通和过渡问题，帮助金融机构平稳的从传统的集中式、微服务架构演进到云原生架构。



异地多活单元化架构

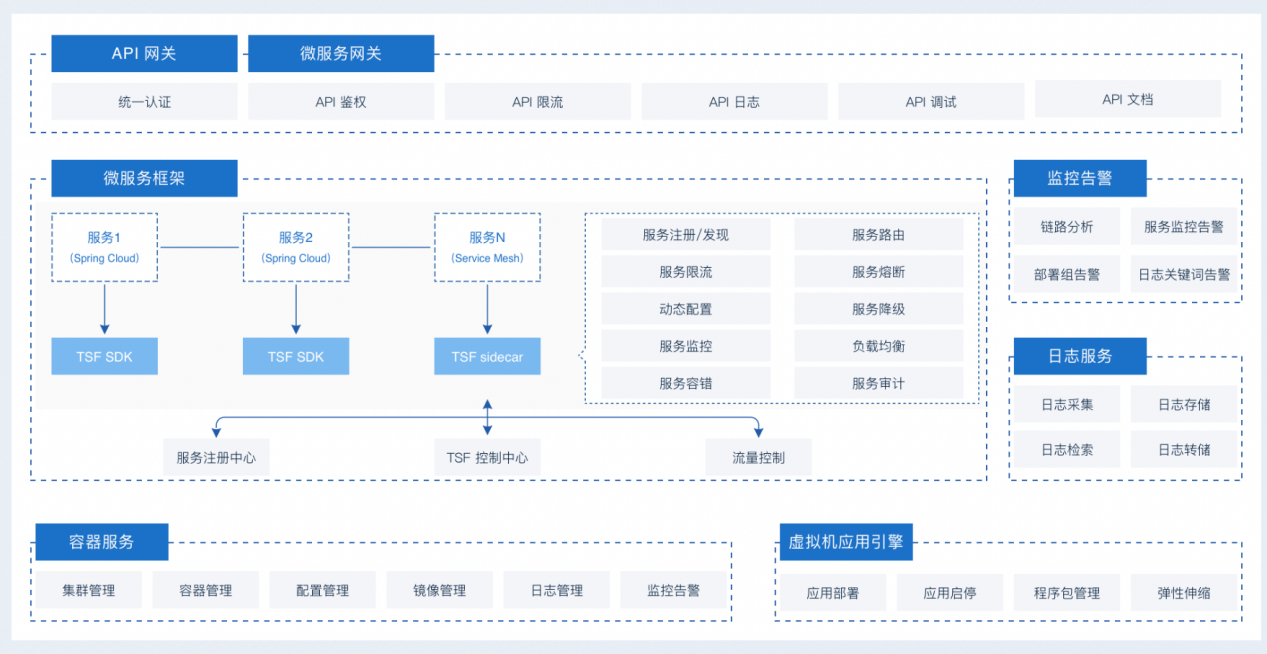
该架构解决方案下，可以避免跨机房、跨城市访问的延迟，真正实现异地多活部署，不但消除了传统“两地三中心”架构中的单独冷备中心，并提升了灾备高可用能力，无论在成本还是在伸缩性、高可用方面，都带来了巨大的优势。



1. TSF

微服务平台（Tencent Service Framework，TSF）是一个围绕着应用和微服务的 PaaS 平台，提供应用全生命周期管理、数据化运营、立体化监控和服务治理等功能。TSF 拥抱 Spring Cloud 、Service Mesh 微服务框架，帮助企业客户解决传统集中式架构转型的困难，打造大规模高可用的分布式系统架构，实现业务、产品的快速落地。针对原生 Spring Cloud 应用与 Mesh 方式零成本接入。TSF 以腾讯云中间件团队多款成熟的分布式产品为核心基础组件，提供秒级推送的分布式配置服务、链路追踪等高可用稳定性组件。此外，TSF 与腾讯云 API 网关和消息队列打通，帮助企业轻松构建大型分布式系统。

1. 产品架构



1. 产品功能

服务注册与发现

|  |  |
| --- | --- |
| 功能 | 详细说明 |
| 金融级高可用注册中心 | 提供金融级高可用的服务注册中心，数据多副本，支持服务自动注册和发现，无须配置注册中心地址即可使用。 |
| 健康检查 | 支持健康检查，如果出现宕机或服务不可用时，注册中心自动剔除不可用实例。 |
| 毫秒级推送 | 客户端和服务注册中心建立长连接，任何服务注册信息变更，立即推送。 |
| 服务本地缓存 | 客户端 SDK 拥有内存和文件级别缓存，当访问注册中心失败的时候会自动启用缓存数据，保证服务发现高可用。 |

细粒度的服务治理

|  |  |
| --- | --- |
| 功能 | 详细说明 |
| 服务鉴权 | 为服务提供安全的访问机制，支持黑白名单鉴权方式，支持系统和业务标签参数进行鉴权。 |
| 服务路由 | 用户可以通过配置、权重标签的形式进行细粒度的流量控制，实现灰度发布、就近路由、部分账号内测、流量限制、访问权限控制等功能。 |
| 服务限流 | 保障业务不被突发流量击垮，提高系统问题稳定性。支持服务和接口级限流配置和监控。 |
| 服务熔断 | 当下游的服务因为某种原因导致服务不可用或响应过慢时，上游服务为了保证自己整体服务的可用性，不再继续调用目标服务，直接返回。当下游服务恢复后，上游服务会恢复调用。 |
| 服务容错&降级 | 支持 failfast、failover 和 forking 容错策略和 fallback 降级方法。 |

全面的应用生命周期管理

|  |  |
| --- | --- |
| 功能 | 详细说明 |
| 多种应用托管方式 | 支持虚拟机和容器部署方式。使用虚拟机部署，应用可以独占资源；使用容器部署，可灵活分配资源实现资源共享。 |
| 应用全生命周期管理 | 提供从创建应用到运行应用的全程管理，功能包括创建、删除、部署、回滚、扩容、下线、启动和停止应用。支持变更记录查询。 |
| 版本管理 | 支持软件仓库和镜像仓库管理程序包版本和镜像版本，支持自定义软件仓库对接用户 COS。 |

高可靠的配置管理

|  |  |
| --- | --- |
| 功能 | 详细说明 |
| 可视化配置管理 | 用户可以在控制台上管理配置，支持多版本管理，支持将配置发布到应用部署单元（部署组）或者命名空间范围。 |
| 配置动态推送，实时生效 | 支持配置动态推送，服务从配置中心读取到更新后的配置进行逻辑处理，支持配置回调方法。支持查看部署组上已发布的配置，支持配置回滚操作。 |
| 推送记录查看 | 支持按照部署组或者配置的维度查看推送的配置记录，支持配置文件的导入和导出。 |

可视化应用运维

|  |  |
| --- | --- |
| 功能 | 详细说明 |
| 服务监控 | 支持服务和接口的成功率、调用量、耗时、异常次数等多维度监控和告警。 |
| 服务依赖拓扑 | 支持查看服务之间的依赖关系，了解系统瓶颈服务和链路并进行针对性的服务优化。支持服务与 API 网关、消息队列、数据库等上下游组件的链路查看。 |
| JVM 监控 | 支持查看 JVM 内存分布、线程、堆栈、火焰图。 |
| 日志服务 | 提供日志采集、日志存储、日志检索，日志关键词告警等功能。支持日志与调用链联动排查线上问题。 |
| 全链路灰度发布 | 在发布过程中，将具有一定特征或者比例的流量分配到需要被验证的版本中，用来观察新的验证版本的线上运行状态。当线上调用链路较为复杂时，全链路灰度发布可以将线上的各个服务隔离出一个单独的运行环境。 |
| 弹性伸缩 | 支持根据预先设定的弹性伸缩规则，动态增加或者减少部署组的实例数。 |

微服务网关

|  |  |
| --- | --- |
| 功能 | 详细说明 |
| 请求转发 | TSF 中的微服务网关可以通过页面配置灵活管理需要被转发请求的微服务 API。微服务网关会及时从注册中心感知后端服务节点健康状况，保证在后端服务节点变动情况下请求不中断。 |
| API 管理 | 微服务网关集中管理了所有需要对外暴露的 API，帮助用户进行 API 的生命周期管理。 |
| API 治理 | 支持 API 级别的限流、路由等能力，支持用户绑定系统插件或自定义插件。 |

## 各微服务选型对比

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **核心要素** | **Dubbo** | **SpringCloud** | **EDSP** | **SOFAStack** | **TSF** |
| 开发难度 | 高 | 低 | 低 | 低 | 低 |
| 开源 | 开源 | 开源 | 可交付源码、可定制化、满足自主可控 | 基于蚂蚁自研、可交付源码、可定制化、满足自主可控 | 基于腾讯云自研、可交付源码、可定制化、满足自主可控 |
| 成熟度 | 高 | 高 | 高 | 高 | 高 |
| 权限控制 | 无 | 无 | 完善的用户权限体系、细粒度权限控制、操作审计 | 完善的用户权限体系、细粒度权限控制、操作审计 | 完善的用户权限体系、细粒度权限控制、操作审计 |
| 通讯协议 | RPC | HTTP/REST | HTTP/REST | RPC、HTTP/REST | HTTP/REST |
| 服务网关 | 无 | Zuul/Gateway | edsp-gateway | 自研API统一网关 | 基于zuul或者scg增强 |
| 观测性 | 无，一般需要手工集成第三方工具 | Sleuth集成Zipkin进行链路展示 | 基于skywalking增强，定制化链路、性能分析，与日志平台高度联动 | 自研RMS实时监控服务。 | 基于spring-sleuth增强 |
| 统一管控 | 管控功能较弱 | 管控界面松散，体验性差 | 统一管控，体验性强 | 统一管控 | 统一管控 |
| 银行案例 | 邮储银行深度定制 | 未知 | 交通银行信用卡核心、兴业银行、渤海银行、民生银行，邮储直销银行 | 南京银行、中国人保、网商银行、天弘基金等 | 中国银行 |
| 售后服务 | 企业运维团队支持 | 企业运维团队支持 | 提供稳定及时的售后服务，强大的技术、运维团队支持 | 售后团队支持 | 售后团队支持 |
| 技术成熟度 | 成熟 | 成熟 | 成熟/完全兼容SpringCloud体系 | 成熟/完全兼容SpringCloud和Dubbo体系 | 成熟/完全兼容SpringCloud体系 |

## 微服务框架选型建议

基于以上分析对比，常见的各类微服务框架技术汇总如下：

1）Dubbo作为国内微服务框架开源的代表作，使用方便上手简单。以及早期的金融微服务场景，多用于互联网行业，存量用户较多。国内银行领域使用的案例较少。

2）SpringCloud作为后起之秀，目前在新的微服务项目中使用的比较多，但是其作为一系列零散的通用框架组合，缺少一整套的微服务解决方案和最佳落地实践支持。

3）长亮企业级分布式服务平台EDSP作为一套完全兼容SpringCloud的微服务技术体系，提供应用全生命周期管理、数字化运维、立体化监控、细粒度服务治理等功能；帮助企业客户解决传统集中式架构转型的困难，打造大规模高可用的金融级分布式系统架构，实现业务和产品的快速落地。在过去的几年，ESDP拥有大量的银行客户案例，成功帮助了几十家银行实现数字化转型，完成核心系统的分布式微服务技术实施，深得广大用户的一致好评。目前在邮储直销中已经使用了EDSP技术平台。

4） SOFAStack功能体系较为完善，也有银行使用案例，且对Dubbo和Spring Cloud体系都能较好的兼容。作为阿里/蚂蚁出品的微服务体系，其在技术先进性和成熟度都较高。不过需要注意的事：其可观测性相关工具采用的自有技术，而不是开源组件。且现有系统如果改造成SOFAStack技术栈，整体的改动工作量非常大。

5）TSF是腾讯基于SpringCloud体系打造的企业级微服务架构框架，功能完善，对单元化架构支持得非常好。目前银行的案例除中行外还不多。

综上所述，**微服务框架选型建议采用EDSP（长亮企业级分布式服务平台）。**