****



**本 科 毕 业 设 计**

院 系

专 业

题 目

年 级 学 号

学生姓名

指导教师 职 称

提交日期

**南京大学本科生毕业论文（设计）中文摘要**

毕业论文题目： 腾讯PowerPlatform自主型广告接入与投放系统的设计与实现

软件学院 院系 软件工程 专业 2014 级本科生姓名： 生宸

指导教师（姓名、职称）： 刘钦 讲师

摘要：

\*\*项目背景(为什么要做这个项目)。

\*\*技术简介(项目开发中使用了哪些技术，有何特点)。

\*\*项目组整体完成了项目中的哪些功能。

\*\*本人在项目中承担了哪些工作。

追求“变现最大化”是所有媒体的终极目标。但市面上媒体多种多样，流量规模、技术能力、商业化现状都参差不齐，流量规模还不错，但商业化现状还不怎么好，且缺乏技术能力的媒体会急需做变现。我们旨在针对这类媒体，提供一整套的流量变现产品技术，最大化帮助媒体做变现。

项目开发阶段使用了微服务的架构以划分多种类业务，并采用了TCC架构以解决微服务架构的强一致性问题，并且采用链路追踪系统来解决微服务架构的定位难问题。主体业务采用Laravel-Lumen开发，涉及高性能服务的模块采用go语言开发并采用http & grpc通信；日志分别存储到文件、数据库和ElasticSearch中；使用ONS名字服务系统进行负载均衡；使用docker gaia进行部署和交付；

项目组主要完成了整体架构的设计以及相关业务的开发，其中包括资源管理、投放管理、客户管理、人群管理、库存管理和效果分析等内容

本人参与了项目整体的体系架构设计，改写Laravel-Lumen的底层框架以提供公共的基础服务，比如ONS名字服务的应用和分库分表的实现；参与了TCC架构的设计与实现；参与了链路追踪系统的部分开发。

关键词：流量变现 微服务 强一致性 链路追踪 分库配置

**南京大学本科生毕业论文（设计）英文摘要**

THESIS：

DEPARTMENT：

SPECIALIZATION:

UNDERGRADUATE:

MENTOR:

ABSTRACT：

KEY WORDS: \*\*项目，\*\*技术(关键名词)1，\*\*技术(关键名词)2，…..

**目 录**

图目录 II

表目录 III

第一章 引言 1

1.1 项目背景 1

1.2 国内(外)工作流建模技术研究现状 2

1.X可补充二级标题 2

1.X.1 可补充三级标题 2

1.X.2 可补充三级标题 2

1.Z 论文的主要工作和组织结构 2

第二章 \*\*技术概述 4

2.1 \*\*技术 4

2.1.1 \*\* 4

2.1.2 \*\* 4

2.2 \*\*技术 5

2.X \*\*本章小结 5

第三章 \*\*系统需求分析与概要设计 6

3.1 \*\*项目整体概述 6

3.1.1 \*\*可补充三级标题 6

3.1.2 \*\*可补充三级标题 6

3.2 \*\*系统的需求分析 6

3.2.1 \*\*可补充三级标题 6

3.2.2 \*\*可补充三级标题 6

3.3 \*\*系统的概要设计 6

3.3.1 \*\*可补充三级标题 6

3.3.2 \*\*可补充三级标题 6

3.X \*\*本章小结 7

第四章 \*\*项目\*\*模块的详细设计与实现 8

4.1 \*\*模块概述 8

4.2\*\*模块的详细设计 8

4.2.1 \*\*可补充三级标题 8

4.2.2 \*\*可补充三级标题 8

4.3 \*\*模块的实现 8

4.3.1 \*\*可补充三级标题 8

4.3.2 \*\*可补充三级标题 9

4.X \*\*本章小结 9

第五章 总结与展望 10

5.1 总结 10

5.2 展望 10

参考文献 11

致谢 12

# 图目录

[图2.1 \*\*图 4](file:///C:\Users\Echo\Desktop\1、本科论文工作手册及学院论文模板\【1】2018届本科毕业论文工作手册\1.本科论文模板_含说明__2017版_20161208.doc#_Toc323067900)

[图2.2 \*\*图 4](file:///C:\Users\Echo\Desktop\1、本科论文工作手册及学院论文模板\【1】2018届本科毕业论文工作手册\1.本科论文模板_含说明__2017版_20161208.doc#_Toc323067901)

[图4.1 MyWorkController类代码 9](file:///C:\Users\Echo\Desktop\1、本科论文工作手册及学院论文模板\【1】2018届本科毕业论文工作手册\1.本科论文模板_含说明__2017版_20161208.doc#_Toc323067902)

# 

# 表目录

[表2.1 \*\*表 5](file:///C:\Users\Echo\Desktop\1、本科论文工作手册及学院论文模板\【1】2018届本科毕业论文工作手册\1.本科论文模板_含说明__2017版_20161208.doc#_Toc323067881)

# 第一章 引言

## 1.1 项目背景

追求“变现最大化”是所有媒体的终极目标。但市面上媒体多种多样，流量规模、技术能力、商业化现状都参差不齐，流量规模还不错，但商业化现状还不怎么好，且缺乏技术能力的媒体会急需做变现。我们旨在针对这类媒体，提供一整套的流量变现产品技术，最大化帮助媒体做变现。如下图所示，我们拟定了一套自主型广告接入与投放系统。

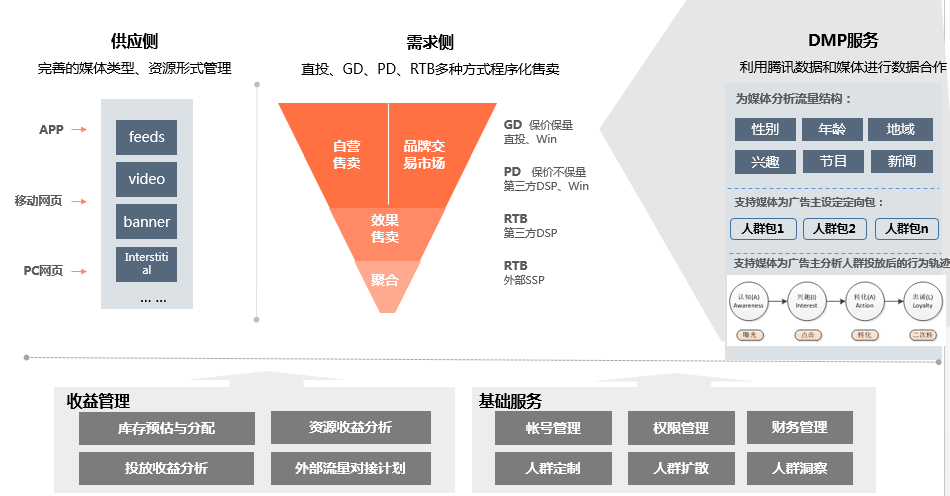


图 PowerPlatform流量销售方案

由于考虑到业务的复杂性，PowerPlatform自主型广告接入与投放系统整体上采用了微服务架构，按业务划分整体系统，降低了开发的复杂度，可以有效地进行自动化部署发布。

然而，微服务架构本身也存在着一些问题：由于业务模块各自有独立的数据访问单元，分布式数据的一致性需要得到保证；网关以及业务模块之间的调用关系错综复杂，发生问题以后难以快速定位；面对高并发，需要同时处理访问高并发和数据高并发。

作者所在的项目组负责开发公共服务模块，以非侵入性的方式解决微服务架构中存在的问题，协助网关与业务模块之间正常通信。

## 1.2 论文的主要工作和组织结构

在本篇论文中，主要针对解决微服务架构问题的公共服务模块进行研究，其中主要包括如下四个方面：保证微服务强一致性的TCC模式；ONS名字服务系统（服务发现与注册，智能路由和负载均衡）；链路追踪系统的部分实现；按用户分库的自动化配置设计。

第一章：概述和前言部分，主要介绍了项目背景，产品解决方案，并描述了该论文的主要工作。

第二章：从项目使用的主要技术方面，介绍项目开发时遇到的难题，对比解决方案的可行性。

第三章：从项目体系架构和需求分析两个方面，整体上描述项目的模块和框架结构，分析功能需求和非功能需求，通过用例图、活动图和模块图等加以展现。

第四章：从项目的具体模块着手，描述模块内的功能与详细设计，并通过类图、顺序图、实体关系图等加以展现。

第五章：总结项目解决问题采取的措施，分析项目中仍然存在的缺点，提出项目改进的意见

# 第二章 \*\*技术概述

## 2.1微服务架构

在本次自主型广告接入与投放项目中，由于业务较为复杂，开发量比较大，因此项目组决定采用微服务架构。根据业务需求将项目进行拆分，形成多个高内聚、低耦合的子业务模块，每个字模块业务拥有独立的数据库，彼此之间通过网关进行Http或Grpc通信。并且微服务架构适合持续开发，持续部署和发布。

### 2.1.1 技术实践

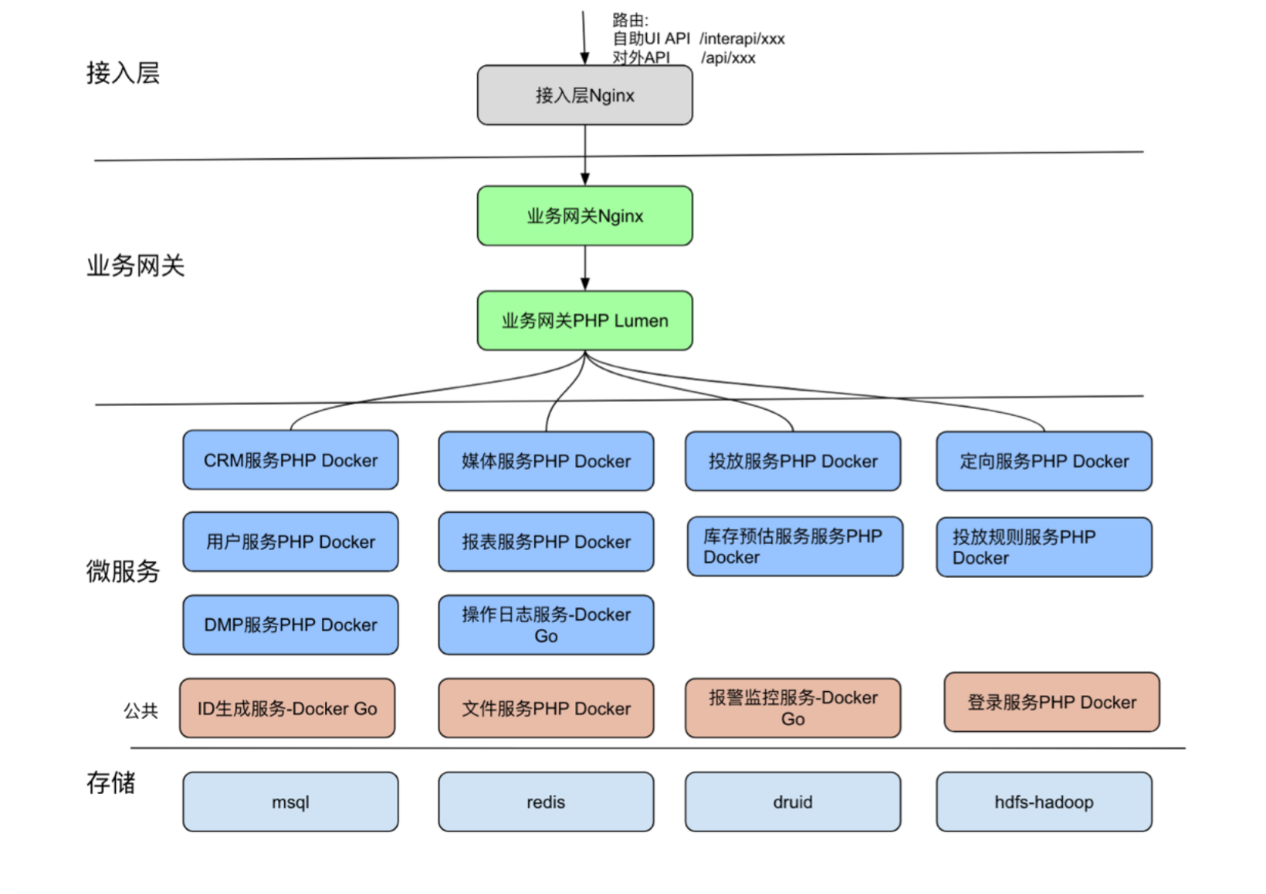


图 自主型广告接入与投放系统-微服务架构图

### 2.1.2 技术优点

1. 将复杂的整体应用分解，分解后的单个模块易于理解、开发和维护
2. 项目易于部署，模块可独立部署，加速部署流程，实现持续部署
3. 由于模块可独立部署，因此不同模块可以使用不同的技术

### 2.1.3 技术缺陷

1. 分布式系统提升了项目的复杂度。比如需要解决模块之间的通信机制问题，分布式数据一致性问题
2. 微服务架构的测试比较困难

## 2.2 TCC框架

为解决微服务架构带来的分布式事务问题，项目组采用了TCC框架，不同于2PC(两阶段提交)，TCC框架属于应用层面，不需要涉及RM(资源管理器)和XA协议，同时TCC框架可以解决2PC种第二阶段出错而引起的问题，因此在项目中对TCC框架进行适当的改进，可以保证微服务的强一致性。

### 2.2.1 分布式事务与2PC

2PC的两阶段：

1. 事务协调者向所有参与者发起prepare操作，要求所有参与者回应
2. 若所有参与者的回应都可以进行任务提交，则事务协调者依次通知各个参与者执行commit操作，否则通知各个参与者执行rollback操作

然而上述的2PC模式在第二阶段存在一些问题：

1. 若事务协调者宕机，则所有的参与者将被卡死在等待执行提交/回滚上
2. 若某个参与者在执行commit操作时发生错误，则其他参与者没有后续处理方案

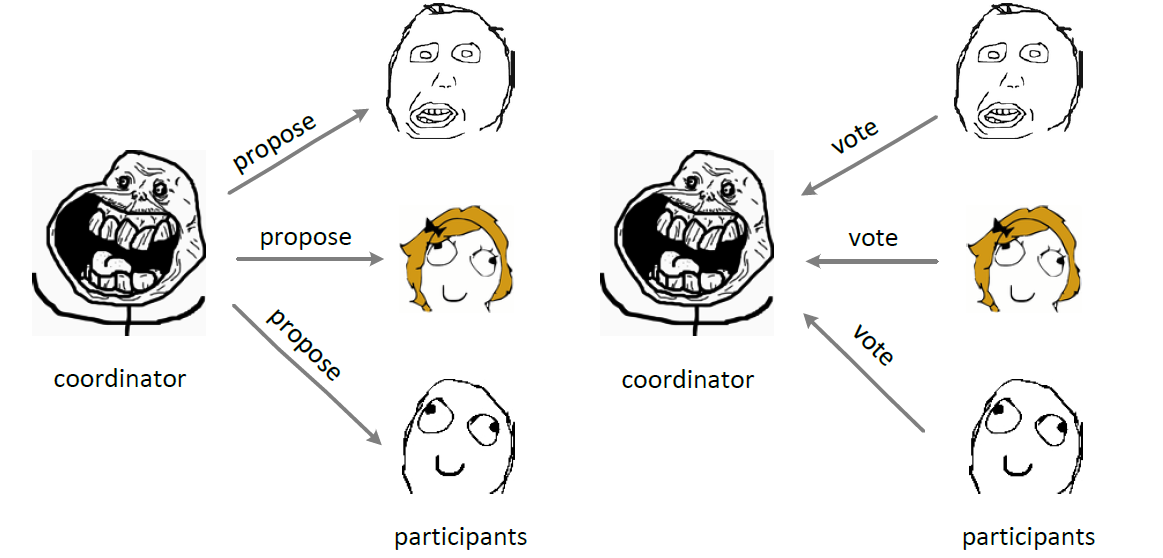


图 两阶段提交模式

### 2.2.2 技术实践

由于TCC框架是从逻辑上保证一致性，所以可以通过流程上的一些改进来弥补漏洞。

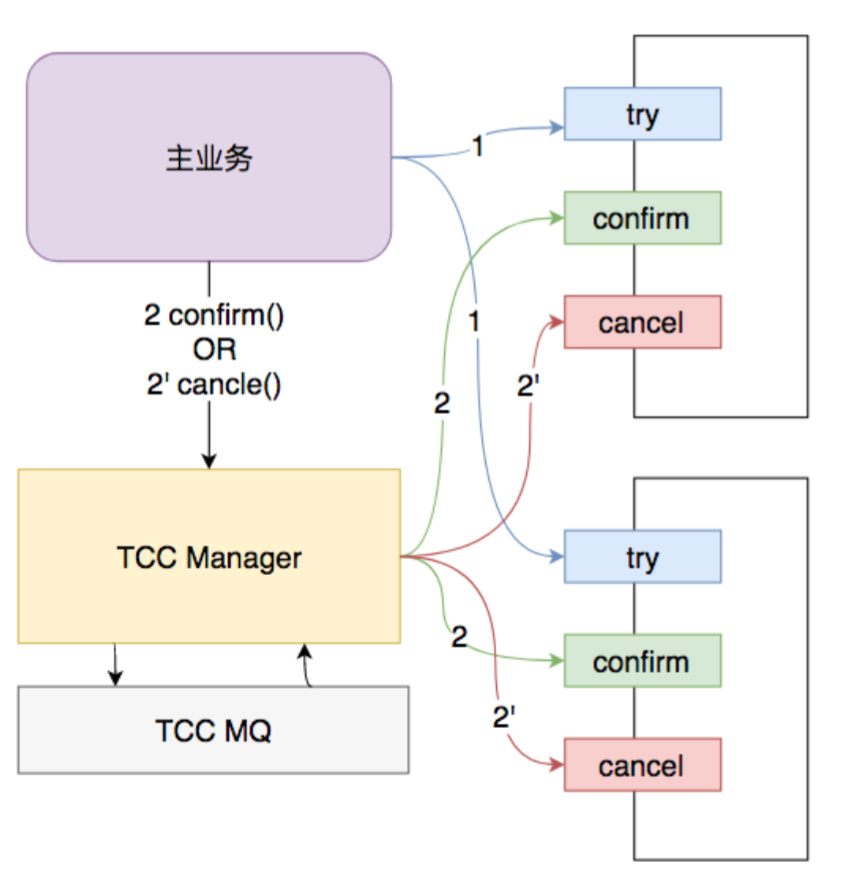


图 TCC模式架构图

try阶段：完成所有业务检查；预留必须的业务资源

confirm阶段：不做任何业务检查；只使用预留的资源；必须是幂等的

cancel阶段：释放预留资源；必须是幂等的

**状态流程：**

1. 主业务在第一阶段同步的执行各个服务的try操作，若出现错误则异步的调用cancel操作释放资源
2. 第二阶段异步的调用confirm/cancel操作

**数据流程：**

1. 在数据记录时，新引入status和sequence两个字段，分别表示当前状态和操作流水标志
2. 更新(Update)操作需要使用不同的sequence

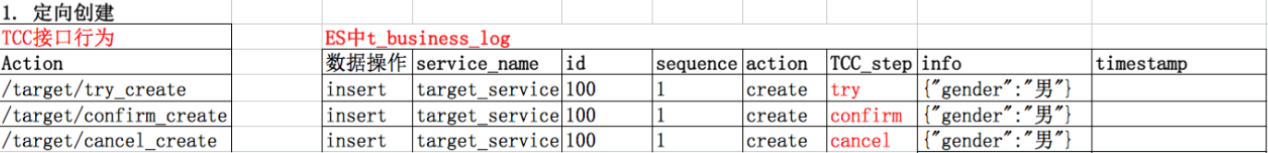


图 定向创建的操作日志记录

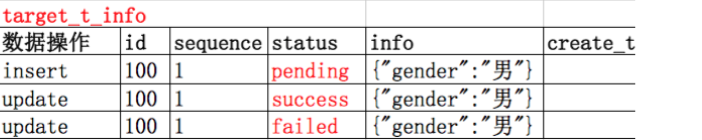


图 定向创建的数据库记录

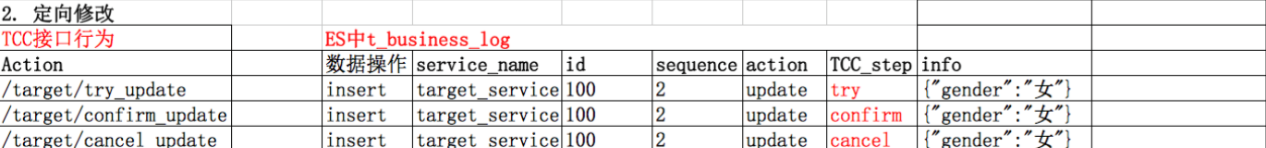


图 定向修改的操作日志记录

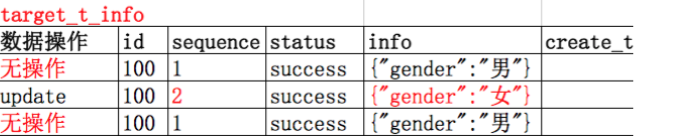


图 定向修改的数据库记录

### 2.2.3 TCC框架的改进

为保证最终一致性，我们对TCC框架做了如下改进：

1. 若第二阶段执行confirm/cancel操作时发生错误，由于confirm/cancel操作的幂等性，TCC MQ会保证在5min内重复执行发生错误的操作。
2. 每隔5分钟进行一次对账操作，扫描距离当前时间[-30min,-5min]的status为try状态且和其同id,sequence的所有记录，理论上相同id和sequence的记录有且仅有两条(要么是[try, confirm]要么是[try, cancel])，对于所有没有闭合的记录，进行报警并通知人工介入排查。

## 名字服务

### 遇到的问题

当客户端或者网关需要调用业务模块时，需要知道实例的网络地址。由于服务实例的网络地址会随着扩展、失败、更新等原因自动更新，如何定制一套服务发现系统成了当务之急。

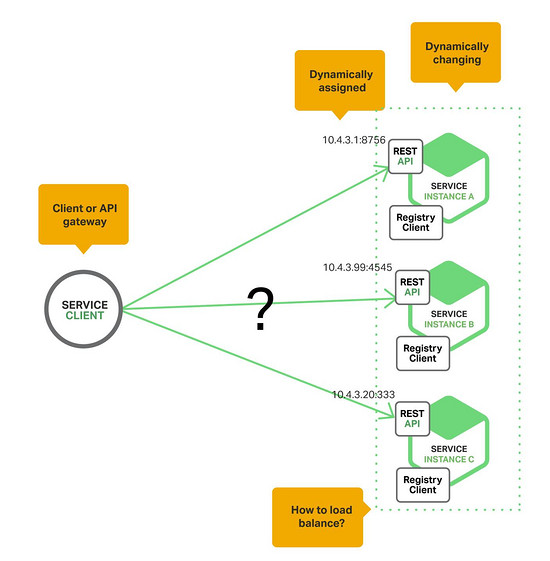


图 客户端/网关调用微服务

### 技术实践

为解决上述问题，我们引入了服务器端服务发现模式。

服务实例可以向服务注册表中注册其网络地址，并通过心跳机制进行检测。客户端向负载均衡器发送请求，负载均衡器再通过查询服务注册表以及负载均衡算法选出网络地址，然后再将请求转发给合适的服务实例。

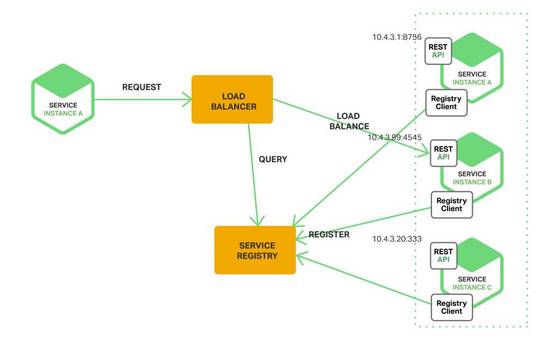


图 服务器端的服务发现模式

## 链路追踪

广告主有权限查看媒体和广告位资源的操作日志，因此需要对所有的操作进行日志记录，在此系统中，项目组使用了链路追踪的方式记录；此外，由于微服务架构存在出现问题难以定位的问题，使用链路追踪，可以通过查询记录日志来快速定位。

### 技术实践

1. 请求开始时，在网关处生成唯一的TraceID，用于标记追踪链
2. 链路追踪开始时，首先在网关处生成一个SpanID，用于记录当前链路的深度，起始处的SpanID为0，若调用一次别的模块，则SpanID长度增加(网关层直接调用A,B两模块，A处的SpanID为01，B处的SpanID为02，若A继续调用模块C，则C的SpanID为011，依次传递)
3. 每进行一次调用时，调用者将TraceID传递给被调用者，被调用者处的SpanID变为X+1位，并生成调用日志信息，通过埋点发送UDP数据给agent，日志信息存储到ElasticSearch中

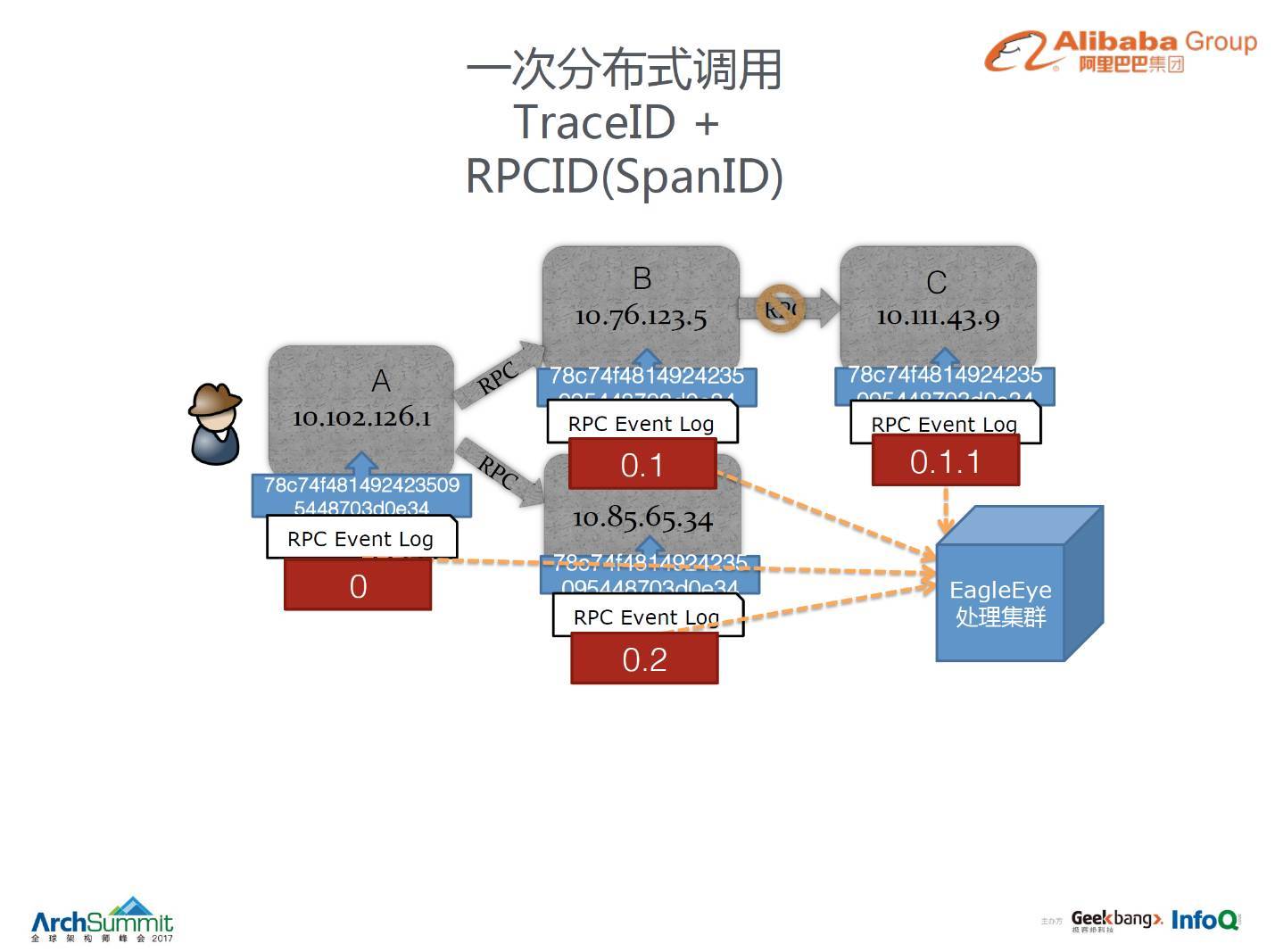


图 链路追踪记录日志

## 2.5 本章小结

在本次项目中，从开发到部署，都采用了当前较为主流的技术方案，并根据项目的实际情况进行相应的调整与修改，提升项目的高性能、可靠性，简化了项目开发的复杂程度

# 第三章 \*\*系统需求分析与概要设计

## 3.1 \*\*项目整体概述

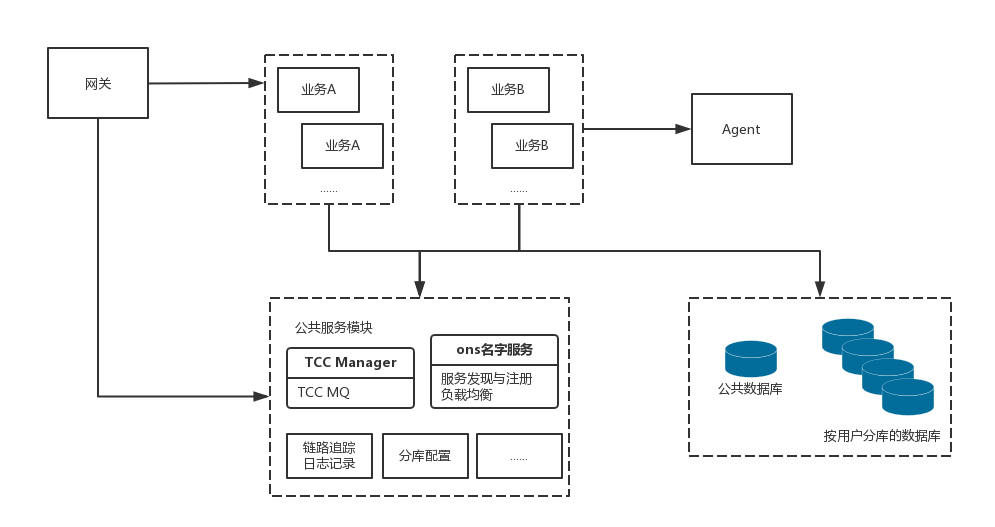
PowerPlatform项目的公共服务模块以composer的方式被各个业务模块以及网关所依赖，再以非侵入的形式为业务模块提供公用服务。

服务注册中心需要在网关以及各个微服务模块启动时进行服务发现并注册，通过心跳检测机制定时监控服务的健康程度。通过智能路由表，构建服务名称-服务地址的映射，并在请求访问某服务时，根据负载均衡算法选出当前最佳的服务地址并访问。公共模块需要负责在启动时为各个业务模块进行注册，并提供个性化的配置选项。

在网关与业务模块以及业务模块之间发生调用关系时，需要进行详细的日志记录，以便于进行链路追踪。链路追踪系统中，一次完整的链路调用共用一个相同的TraceID，并且以SpanID来表示此链路的调用栈结构。公共模块需要在各业务模块中埋点，在业务模块将日志信息以UDP包发送时捕获，并将日志信息转发给ElasticSearch存储。

在TCC模式中，公共模块需要负责提供调用接口，以便对一次请求中涉及的多个微服务进行注册和调用，并在ElasticSearch中记录操作的流水日志，利于对账，保证微服务的最终一致性。

在分库配置时，公共模块需要根据用户id自动化地为数据库的连接进行配置，例如数据连接地址、用户名和密码，以及连接的主库等。



图公共模块整体描述

## 3.2 \*\*系统的需求分析

### 3.2.1用例图

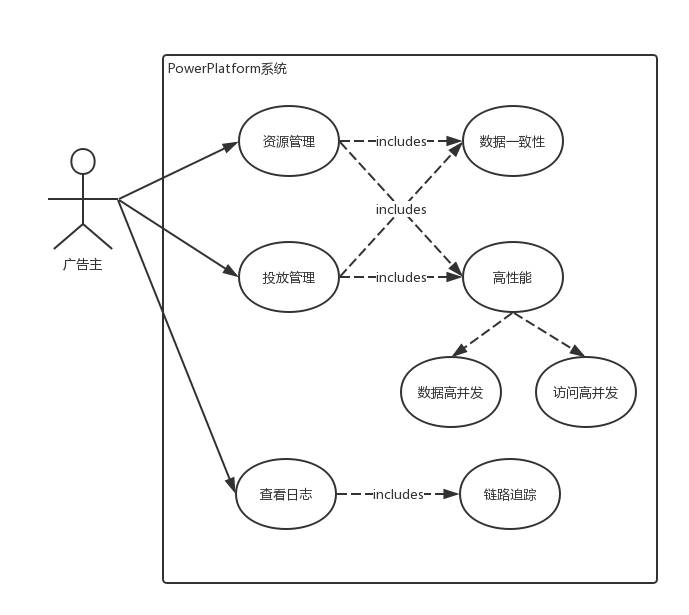


图 用例图

### 用例描述

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | 1 | 名称 | TCC保证微服务一致性 | 优先级 | 高 |
| 参与者 | API网关，各个微服务模块 | | | | |
| 触发条件 | 网关在一次请求中调用多个微服务模块处理业务 | | | | |
| 前置条件 | 无 | | | | |
| 后置条件 | 保证这一次请求的生命周期内，不会出现数据的不一致性 | | | | |
| 正常流程 | 1. 网关接收到请求，需要调用A,B两个模块 2. 网关向A业务模块发送资源尝试请求(try信号)，等待A模块响应 3. 网关向B业务模块发送资源尝试请求，等待B模块响应 4. 待A,B同步响应后，异步向A,B两个模块发送资源确认请求(confirm信号)，等待A,B模块响应 5. 待A,B异步响应后，将处理结果返还 | | | | |
| 扩展流程 | 2a. A模块尝试分配资源时出错   1. 网关异步向A,B两个模块发送资源取消请求(cancel信号) 2. 记录异常并进行报警   3a. B模块尝试分配资源时出错   1. 网关异步向A,B两个模块发送资源取消请求(cancel信号) 2. 记录异常并进行报警   4a. A模块在执行confirm操作时出错   1. 任务队列在5min中内重复执行confirm操作 2. 若5min内confirm操作仍未成功执行，则将被标记为未闭合状态，报警并通知人工干预   4b. B模块在执行confirm操作时出错  同4a  4c. 网关在接收到A,B模块的try响应后崩溃   1. 由于5min后A,B模块的try操作都处于未闭合状态，系统将会报警并通知人工干预 | | | | |
| 业务规则 | Try, confirm, cancel信号都由TCC MQ发起  Confirm, cancel操作都是幂等的  由一个每隔5min执行的对账任务检查所有未闭合的记录 | | | | |
| 特殊需求 |  | | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | 2 | 名称 | 数据分库配置 | 优先级 | 高 |
| 参与者 | 微服务模块，Mysql数据库 | | | | |
| 触发条件 | 微服务模块对数据进行增删改查操作 | | | | |
| 前置条件 | 数据库分为公共数据库和按照用户分库的个人数据库 | | | | |
| 后置条件 | 配置好正确的数据库连接 | | | | |
| 正常流程 | 1. 微服务模块接收到请求，需要对某数据库内容进行操作 2. 若只需要配置公共数据库连接，读取模块内的配置信息，进行一次名字服务，生成ip+port等连接信息 | | | | |
| 扩展流程 | 2a. 由于个人数据是按照用户分库的(即不同用户的数据库之间结构相同，只有数据内容有差异)，所以查询时需要进行分库配置   1. 根据请求的用户id，查询内存中是否有对应的用户id和ip+port映射，如果存在，直接生成连接信息 2. 若内存中不存在这样的映射，则查询redis，如果redis中存在，则记录到内存中，并且生成连接信息 3. 若redis中仍然不存在这样的映射，则查询公共数据库中的用户数据库表，将连接信息记录到redis中，并生成连接信息 | | | | |
| 业务规则 |  | | | | |
| 特殊需求 | 若需要获取所有用户的某个共同数据，系统会先得到所有用户的ip+port，在连接池中建立不同的连接(若ip+port相同 只是库不同 则视为同一个连接)，查询后返回collection | | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | 3 | 名称 | 名字服务 | 优先级 | 高 |
| 参与者 | 网关，微服务模块 | | | | |
| 触发条件 | 网关需要向微服务模块发起一次请求 | | | | |
| 前置条件 | 微服务模块已经向服务注册表zk\_host注册了当前的ip+port信息 | | | | |
| 后置条件 | 无 | | | | |
| 正常流程 | 1. 网关根据业务需求，获取到需要交互的微服务模块的名字 2. 根据上述的名字，查询名字服务系统 3. Zk\_host根据负载均衡算法，选择一个最优的服务器，并返回ip+port 4. 网关根据获取到的ip+port进行访问 | | | | |
| 扩展流程 |  | | | | |
| 业务规则 |  | | | | |
| 特殊需求 |  | | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | 4 | 名称 | 链路追踪 | 优先级 | 高 |
| 参与者 | 网关，微服务模块，agent | | | | |
| 触发条件 | 网关接收到请求后，调用各个微服务模块 | | | | |
| 前置条件 | 在各个微服务模块上埋点 | | | | |
| 后置条件 | 将模块之间的调用关系以链路的方式进行记录 | | | | |
| 正常流程 | 1. 网关接收到请求，生成唯一TraceID， 标记调用链路；生成SpanID，记录调用链的深度，初始为0 2. 网关调用模块A，并将TraceID传入，SpanID变为01，由于模块A已被埋点，生成此次调用的日志记录 3. 模块A继续调用模块C，将TraceID传入，SpanID变为011，生成调用记录 4. 网关调用模块B，并将TraceID传入，SpanID变为02，生成调用记录 5. 完成此次请求后，将以UDP的形式将调用记录传递给agent，最终记录将存储到ElasticSearch中 | | | | |
| 扩展流程 |  | | | | |
| 业务规则 | 调用日志中每条记录包含了：调用方和被调用方的ip，port，调用的耗时，调用结果的返回值、状态码等信息 | | | | |
| 特殊需求 |  | | | | |

### 系统顺序图

图一展示了微服务中的名字服务和分库配置

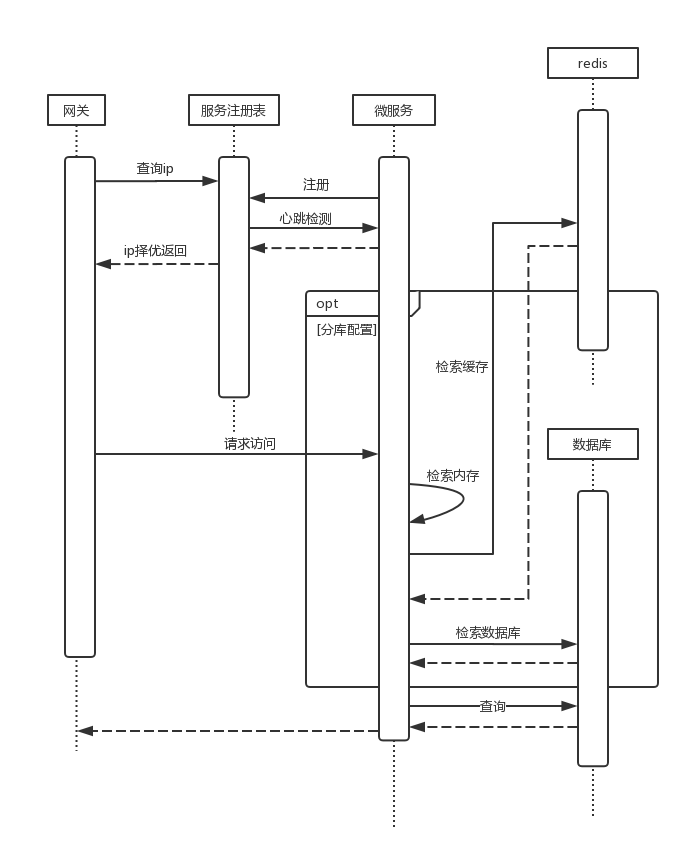


图 系统顺序图-1

图二展示了TCC模式以及链路追踪

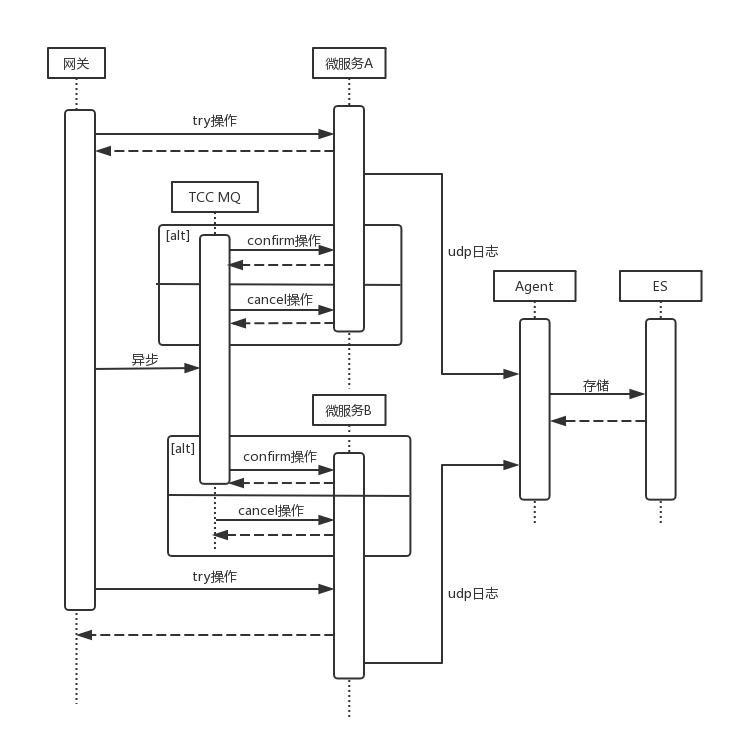


图 系统顺序图-2

### 实体关系图

由于作者参加的模块大多为公共服务开发，因此涉及业务层面的实体关系较少，只有在分库配置中存在用户和库配置一对一的关系表，结构如下。

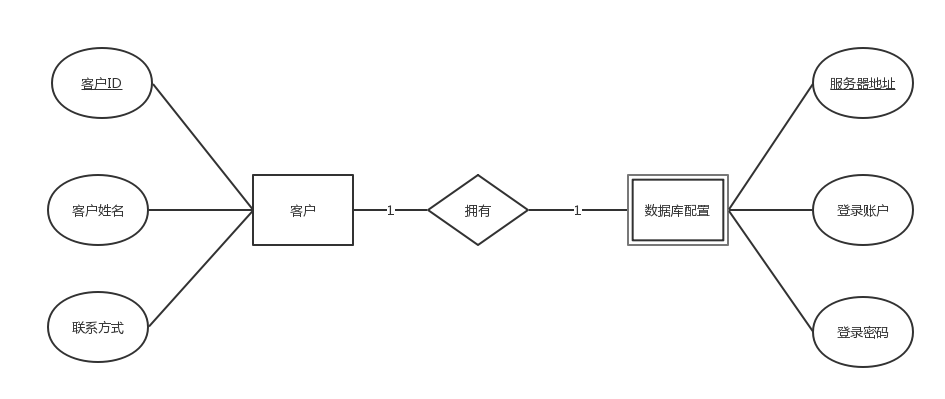


图 分库配置-实体关系图

## \*\*系统的概要设计

### 模块图

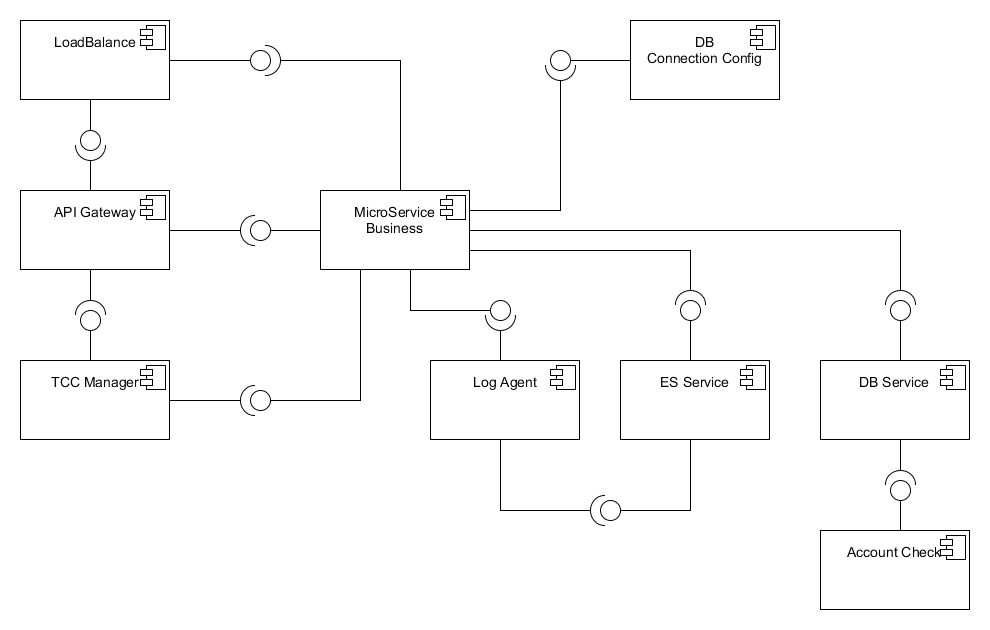


图 开发模块图

### 接口定义

|  |  |  |  |
| --- | --- | --- | --- |
| ID | | | 1 |
| 模块名 | | | LoadBalance |
| 供给接口 | | | |
| 接口ID | 注册 | 功能描述 | 提供服务注册接口，以便业务模块注册ip+port |
| 语法 | ServerConfig serverRegister(ServerConfig sc) |
| 前置条件 | 业务模块已经激活并可以正常使用 |
| 后置条件 | 无 |
| 不变量 | 无 |
| 择优 | 功能描述 | 根据服务名字，通过负载均衡算法选择最佳的服务地址 |
| 语法 | ServerConfig serverSelect(ONSName name) |
| 前置条件 | 已注册的名字匹配的服务器数量至少为1个 |
| 后置条件 | 无 |
| 不变量 |  |
| 需求接口 | | | |
|  | | | |

|  |  |
| --- | --- |
| ID | 2 |
| 模块名 | API Gateway |
| 需求接口 | |
| ServerConfig serverSelect(ONSName name)  OperationRecv(TCCJob job)  BusinessProcess(HttpRequest req) | |

|  |  |  |  |
| --- | --- | --- | --- |
| ID | | | 3 |
| 模块名 | | | TCC Manager |
| 供给接口 | | | |
| 接口ID | TCC  处理 | 功能描述 | 接收TCC模式的操作请求，再以同步或异步的方式处理 |
| 语法 | void OperationRecv(TCCJob job) |
| 前置条件 | 网关在一次请求中需要访问多个业务模块，并且保证一致性 |
| 后置条件 | 根据业务模块的状态，相应的执行confirm/cancel操作 |
| 不变量 |  |
| 需求接口 | | | |
| Bool TCCOperate(TCCJob job) | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| ID | | | 4 |
| 模块名 | | | MicroService Business |
| 供给接口 | | | |
| 接口ID | 业务  处理 | 功能描述 | 接收来自网关的请求，进行业务处理后返回数据 |
| 语法 | BusinessProcess(HttpRequest req) |
| 前置条件 | 无 |
| 后置条件 | 无 |
| 不变量 |  |
| TCC处理 | 功能描述 | 提供try,confirm,cancel接口用于资源分配、锁定、释放 |
| 语法 | Bool TCCOperate(TCCJob job) |
| 前置条件 | 业务模块接收到来自TCC MQ的请求信息 |
| 后置条件 | 对业务资源进行相应的操作 |
| 不变量 |  |
| 链路  追踪 | 功能描述 | 记录每一次业务模块的调用信息，以及模块间的通信信息 |
| 语法 | TraceInfo recordTrace(TraceInfo info, target, source) |
| 前置条件 | 无 |
| 后置条件 | 以UDP的形式将调用的日志信息发送给Agent |
| 不变量 |  |
| 需求接口 | | | |
| ServerConfig serverRegister(ServerConfig sc)  DBConfig connConfig(bool isPublic, int userId)  Query ACID(SQL sql)  void stateRecord(TCCJob job) | | | |

|  |  |
| --- | --- |
| ID | 5 |
| 模块名 | LogAgent |
| 需求接口 | |
| TraceInfo recordTrace(TraceInfo info, target, source)  void logRecord(TraceInfo info) | |

|  |  |  |  |
| --- | --- | --- | --- |
| ID | | | 6 |
| 模块名 | | | DB Connection Config |
| 供给接口 | | | |
| 接口ID | 分库  配置 | 功能描述 | 在业务模块访问数据前，对数据库连接进行配置 |
| 语法 | DBConfig connConfig(bool isPublic, int userId) |
| 前置条件 | 无 |
| 后置条件 | 根据业务模块的需求，修改连接配置信息 |
| 不变量 |  |
| 需求接口 | | | |
|  | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| ID | | | 7 |
| 模块名 | | | DB Service |
| 供给接口 | | | |
| 接口ID | 数据  操作 | 功能描述 | 普通的数据增删改查 |
| 语法 | Query ACID(SQL sql) |
| 前置条件 | 无 |
| 后置条件 | 无 |
| 不变量 |  |
| 需求接口 | | | |
| checkResult accountCheck() | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| ID | | | 8 |
| 模块名 | | | ES Service |
| 供给接口 | | | |
| 接口ID | 日志  记录 | 功能描述 | 接收来自Agent的链路追踪的日志，之后以KannBoard图形化显示 |
| 语法 | void logRecord(TraceInfo info) |
| 前置条件 | Agent接收一次请求中所有埋点的发包信息，整合后发送给ES |
| 后置条件 | 无 |
| 不变量 |  |
| TCC  记录 | 功能描述 | 记录TCC模式下的状态信息 |
| 语法 | void stateRecord(TCCJob job) |
| 前置条件 | TCC MQ执行某个TCC Job |
| 后置条件 | 将此job的执行状态信息存储在ES中 |
| 不变量 |  |
| 需求接口 | | | |
|  | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| ID | | | 9 |
| 模块名 | | | Account Check |
| 供给接口 | | | |
| 接口ID | 对账 | 功能描述 | 每隔5分钟执行一次对账任务，扫描[-30min, -5min]内容曾经有过try状态的记录，并判断它们是否都闭合了 |
| 语法 | checkResult accountCheck() |
| 前置条件 | 无 |
| 后置条件 | 对于没有达到闭合状态的记录，进行报警，并通知人工干预 |
| 不变量 |  |
| 需求接口 | | | |
|  | | | |

### 系统框架结构图

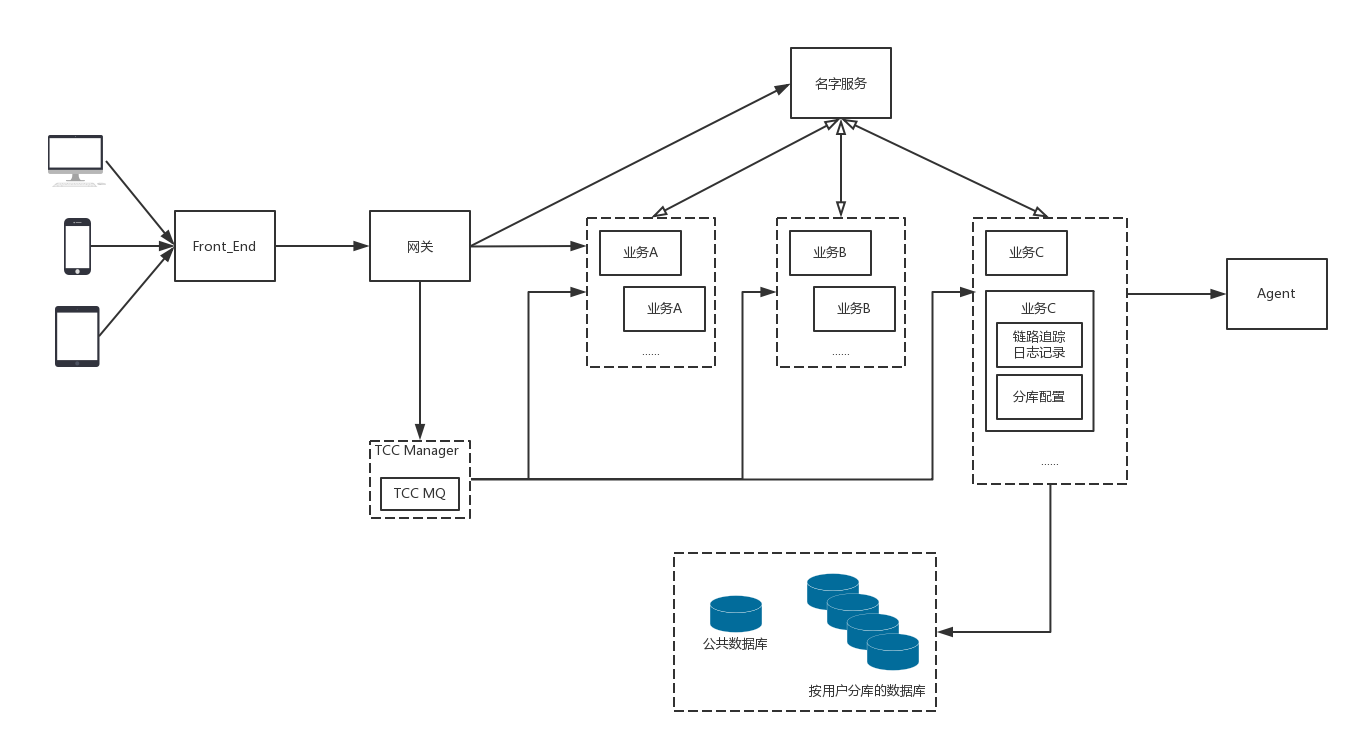


图 系统框架结构

## 3.4 本章小结

# 第四章 \*\*项目\*\*模块的详细设计与实现

## 4.1 \*\*模块概述

## 4.2\*\*模块的详细设计

### **4.2.1 \*\*可补充三级标题**

### **4.2.2 \*\*可补充三级标题**

## 4.3 \*\*模块的实现

### **4.3.1 \*\*可补充三级标题**

在实现部分，可以有少量关键性的代码，代码的排版形式，代码贴入表格框(1行\*1列)，设置为可以跨页的形式，且代码中的字体使用“Arial/五号/单倍行距”，对该表格框标记为图x.y，并给出以该段代码的用途给出图名。示例如下：

|  |
| --- |
| public String signAndUnsignList() throws Exception {  String resultStr = SIGN\_UNSIGN\_LIST\_RESULT;  Loginer loginer = this.getLoginer();  Page page = null;  CriteriaSetup criStp = new CriteriaSetup();  try {  //设置界面不分页  this.ecLimit = ExtremeTablePage.getLimit(this.getHttpServletRequest(), 999999);  page = this.myWorkextSvc.FindTasksOfActorId(loginer, new Long(ProcessVariable.JBPM\_FLOW\_TASK\_ONLINE),  this.ecLimit, criStp);  } catch (Exception ex) {  Log.debug(ex);  throw new SysException(SysExcepType.BUG\_UNKNOWN\_RUNTIME\_EXCEPTION, ex);}  List list = page.getResult();  //排序  Comparator comp = new Comparator(){  public int compare(Object o1,Object o2) {  TaskBillVO v1=(TaskBillVO)o1;  TaskBillVO v2=(TaskBillVO)o2;  if((v1.getDuration()==null || v1.getDuration().equals("")  || v1.getDuration().equals("null")) && (v2.getDuration()!=null  && !v2.getDuration().equals("") && !v2.getDuration().equals("null")))  return 1;  else if((v2.getDuration()==null || v2.getDuration().equals("")  || v2.getDuration().equals("null")) && (v1.getDuration()!=null  && !v1.getDuration().equals("") && !v1.getDuration().equals("null")))  return 0;  else  return 1;  }  };  Collections.sort(list,comp);  this.setTaskbillVOs(list);  getCaseNum(list);  this.getHttpServletRequest().setAttribute("totalRows", page.getTotalCount());  return resultStr;} |

图4.1 MyWorkController类代码

### **4.3.2 \*\*可补充三级标题**

## 4.X \*\*本章小结

# 第五章 总结与展望

## 5.1 总结

## 5.2 展望

# 参考文献

[1] 作者,译者,书名(版本),出版地:出版社,出版时间,引用部分起止页.

[2] 作者,译者,文章题目,期刊名,年份,卷号(期数):引用部分起止页.

[3] 作者,学位论文名,本科/硕士/博士论文,大学/机构名,年份.

[4] 网页的主题,URL.

# 致谢

感谢指导老师……

感谢参与本项目的同组的其他同学……

感谢其他……