**基于微服务架构的在线云盘的设计与实现**

**Design and implementation of an online cloud storage based on microservice architecture**

目 录

[1 绪论 2](#_Toc130130730)

[1.1 选题背景及研究意义 2](#_Toc130130731)

[1.2 国内外研究现状 3](#_Toc130130732)

[1.3 研究内容 3](#_Toc130130733)

[1.4 论文组织安排 4](#_Toc130130734)

[2 关键技术及理论基础 4](#_Toc130130735)

[2.1 微服务架构概述 4](#_Toc130130736)

[2.2 Token校验机制概述 5](#_Toc130130737)

[2.3 FTP服务器概述 6](#_Toc130130738)

[2.4 Redis缓存策略概述 6](#_Toc130130739)

[2.5 Docker概述 6](#_Toc130130740)

[2.6 RESTful概述 7](#_Toc130130741)

[2.7 Eureka注册中心 7](#_Toc130130742)

[2.8 Ribbon负载均衡 8](#_Toc130130743)

[2.9 本章小结 8](#_Toc130130744)

[3 需求分析 8](#_Toc130130745)

[3.1 系统概述 9](#_Toc130130746)

[3.2 功能性需求 9](#_Toc130130747)

[3.3 非功能性需求 11](#_Toc130130748)

[3.4 本章小结 11](#_Toc130130749)

[4 系统设计 11](#_Toc130130750)

[4.1 系统架构设计 11](#_Toc130130751)

[4.2 基础服务设计 12](#_Toc130130752)

[4.2.1 账户管理服务设计 12](#_Toc130130753)

[4.2.2 文件管理服务设计 13](#_Toc130130754)

[4.2.3 公告管理服务设计 13](#_Toc130130755)

[4.3 进阶服务设计 13](#_Toc130130756)

[4.3.1 身份认证服务设计 13](#_Toc130130757)

[4.3.2 文件共享服务设计 14](#_Toc130130758)

[4.3.3 数据分析服务设计 14](#_Toc130130759)

[4.4 数据设计 14](#_Toc130130760)

[4.4.1 关系型数据库设计 14](#_Toc130130761)

[4.4.2 非关系型数据库设计 15](#_Toc130130762)

[4.4.3 FTP服务器设计 16](#_Toc130130763)

[4.5 本章小结 16](#_Toc130130764)

[5 系统实现 17](#_Toc130130765)

[5.1 账户管理服务实现 17](#_Toc130130766)

[5.2 文件管理服务实现 19](#_Toc130130767)

[5.3 公告管理服务实现 20](#_Toc130130768)

[5.4 文件共享服务实现 20](#_Toc130130769)

[5.5 本章小结 21](#_Toc130130770)

[6 系统测试 21](#_Toc130130771)

[6.1 测试概述 21](#_Toc130130772)

[6.1.1 测试目的 21](#_Toc130130773)

[6.1.2 测试背景 21](#_Toc130130774)

[6.1.3 测试范围 21](#_Toc130130775)

[6.1.4 测试点总览 21](#_Toc130130776)

[6.2 系统功能测试（例） 22](#_Toc130130777)

[6.3 系统非功能测试（例） 23](#_Toc130130778)

[6.4 本章小结 25](#_Toc130130779)

[7 系统部署 26](#_Toc130130780)

[7.1 服务器环境搭建 26](#_Toc130130781)

[7.2 服务模块部署 27](#_Toc130130782)

[7.3 数据库部署 28](#_Toc130130783)

[7.4 本章小节 28](#_Toc130130784)

[8 总结与展望 28](#_Toc130130785)

[8.1 全文总结 28](#_Toc130130786)

[8.2 展望 29](#_Toc130130787)

[9 致谢 29](#_Toc130130788)

[10 参考文献 29](#_Toc130130789)

# 绪论

## 选题背景及研究意义

## 国内外研究现状

## 研究内容

近年网络速度大幅提升后，云计算的应用和流行，带动了虚拟化、网络存储、分布式计算、负载均衡、并行计算各领域的发展，引领了应用上云的潮流，甚至推动了边缘计算等前沿领域的研究进展[2]。以云盘为代表的应用型云存储技术，其本质也是云计算技术大类中的子集。云存储方案的落地，需要融合多方软硬件资源，借助网络传输技术、虚拟化技术、文件存储及管理技术等各种技术手段，最终实现文件上云存储、上云管理、方便下载、灵活调整、保证安全等诸多现实需求。目前，众多云存储项目研究，倾向于利用分布式系统框架构建大规模的存储服务器集群，在存储服务器的管理、监控等方面进行设计和改良。

云存储项目方案的构成，集群化存储管理、监控是一个重要的方面，而面向用户层面的应用系统、服务系统也是不可或缺的。应用层面的业务逻辑，对上需要面对高并发的用户请求，对下需要链接底层存储设计并及时建立数据交流，其在性能和拓展性上需要充分考量，做出进一步研究。

## 论文组织安排

本文分为八章，具体结构如下：

第一章为绪论部分，主要介绍论文的研究背景、国内外现状、研究目的。本章作为论文头部，为后文的展开和详细论述指明了主题和方向。

第二章为关键技术及理论基础，主要介绍该课题项目所使用到的设计架构思想、技术解决方案，使项目结构在技术栈的层面上更为清晰明了。

第三章为需求分析，主要介绍了该课题项目需要实现的功能和目的。解释了云盘的用途、给什么人用、如何去用。

第四章为系统设计，主要介绍了在架构设计和详细设计阶段，系统的逻辑结构和层次。在不依赖代码详细实现之前，阐述清楚系统的实现方案。

第五章为系统实现，主要侧重说明在编码阶段的解决方案，其中涉及到详细的模块接口定义和约束条件，能够更为直观地了解到系统的运行逻辑。

第六章为系统测试，主要介绍了在系统开发阶段和试运行阶段涉及到的测试流程、测试策略以及测试结果，分析并说明了系统开发过程中的不足以及改进过程。

第七章为系统部署，主要介绍了系统在测试环节和发布环节所使用的硬件环境和运行状态，能够充分表现系统的当前状态。

第八章为总结与展望，充分概括和总结的该课题项目推进的全流程：从课题立项到需求分析，再到系统设计与实现，最后到测试、部署、运维。此外还对项目的成果做出了理性的评价和概括，同时也指出项目的不足之处以及对未来开发和设计的展望。

# 关键技术及理论基础

## 微服务架构概述

微服务架构作为一种软件体系结构，具有高扩展性、高灵活性、分布式部署等众多优点。我认为其架构思想是对代码单一职责原则的充分体现。微服务架构的项目实施，需要对项目需求和功能模块有较为深入和透彻的认识。在需求分析和设计过程中，把业务逻辑划分为基础性模块和非基础性模块。基础性模块为上游提供基础性服务，模块间尽可能解耦，实现灵活的部署调度和服务支撑。而上游模块在开展业务过程中也应保持较高的内聚状态，避免出现复杂的远程过程调用和依赖关系网。

这种业务工作模式优势非常明显，以下给出3个例子加以说明。

当用户访问量升高时，传统的大后端业务模块很难做出调整，短期内只能依赖硬件资源的性能强行维持服务状态。而基于微服务架构的集群系统，可以根据用户量动态调整业务模块可用数。如登陆模块可以在短时间内横向扩展，以应对流量高峰期，用户的登陆需求。而当用户访问量降低时，又可以按需收缩模块数量，节省了网络资源和服务器硬件资源。这种动态调整和弹性部署的解决方案，需要从项目之初，架构之处，就开始设计和分析。这正是微服务架构的价值之一。

当系统受到非法入侵、恶意破坏或者硬件资源故障时，微服务架构的多点模块可以保证系统的高可用性。该系统如若部署了多个身份认证模块在不同的服务器上，当其中1台或几台服务器故障崩溃时，用户登陆请求可由其他服务器来实现，而不会出现大面积无响应或是拒绝服务的状态。

当系统在迭代过程中，出现了新的需求时，微服务架构又会表现出强大的扩展性和兼容性。新的业务需求和新的服务模块，可以快速融入系统集群。更重要的是，这种业务扩展并不需要对已有服务进行调整，也不会对已经部署的业务模块造成影响，实现了一定程度上的热部署和热更新。

## Token校验机制概述

Token校验机制实际上是一种短期身份认证机制。用户在首次登陆的时候获得系统生成的即时令牌。该令牌存在一定的生命周期，借助系统服务模块来维护。对于微服务架构的系统，各服务模块间需要尽可能降低数据交流，依靠令牌校验识别用户身份，进而提供可靠的服务。当令牌到期时，用户自动进入离线状态，各服务模块拒绝服务，直到再次登陆时，才能获取新的令牌。

Token校验机制还可以有效防止一个账户多点登陆的问题。例如在多个设备或客户端进行相同账户登陆操作，则各端均会生成该账户的身份令牌，而在系统中相同账户的身份令牌会相互覆盖，则最后登陆端获取有效身份令牌。这种解决方案，通过使用简单的校验机制，解决的重复登录和踢出下线的非功能需求。

## FTP服务器概述

FTP协议作为网络应用层中的常见协议，主要提供不同种类主机系统之间的文件传输服务，此外还可以以用户权限管理的方式提供用户对远程FTP服务器上的文件管理能力[王道]。FTP服务器默认使用20、21端口与客户端进行数据交互。其中，21端口用于控制连接，20端口用于数据传输。同时，FTP协议还提供了两种传输模式：主动模式和被动模式。在主动模式中，客户端连接到服务端的21端口，登陆成功后客户端随机开放一个数据传输端口给服务端，服务端通过20端口和客户开放的端口进行数据传输；在被动模式中，由服务端随机选择一个数据传输端口，与客户端连接进行数据传输。

在云盘系统中，最主要、最核心的业务需求就是文件存储，因而需要提供可靠的文件上传、文件下载服务。采用FTP服务器作为文件存储服务器，是一种较为简单和易实现的解决方案。

## Redis缓存策略概述

Redis作为一种非关系型数据库，有着更为灵活简介的数据存储策略。同时，Redis作为一种内存型数据库，有着更高的相应速度和并发负载能力。Redis数据库主要使用键值对（Key-Value）结构进行数据组织和存储。其中，字符串键值是最简单的一种存储方式。除此之外还有Hashes、Lists、Sets以及Sorted Sets等多种复合结构。这些结构加上数据库自身的性能优势，Redis常用作缓存数据和消息队列。

在云盘系统中，使用Redis作为账户数据和文件元数据的缓存数据库，加上服务模块的横向拓展，同时为关系型数据库分担访问压力，可以为系统提供更高的并发负载性能。

## Docker概述

在云计算过程中，虚拟化技术一直是不容忽视的关键内容，整体而言，该技术主要涉及到容器和虚拟机两种实现途径。容器技术因其轻量化的特征，能够充分利用操作系统本身特性，越来越受到市场的欢迎。【合称】

Docker作为一种快速交付应用、运行应用的技术，可以将项目内容、相关依赖和运行环境进行打包。这种打包形成的镜像具备更高的通用运行性。在不同的系统环境下，利用Docker镜像部署的项目不必担心因为开发环境和生产环境的差异，而产生的问题和潜在的隐患。另一方面，对于相同系统环境下的不同项目，Docker还提供了环境隔离的条件，使用Docker镜像实例化的各容器独立运行，容器内部的程序无法感知到对方的存在，仅通过暴露出的相关接口实现数据交流。这样，Docker实现了类似于虚拟机的运行环境，可以把Docker认为是一种更为轻量级的虚拟机解决方案。

在微服务架构体系下的项目实现，利用Docker进行部署更为简洁方便，更为快速。因为微服务架构下，相同的服务模块可能会横向部署多个在不同的服务器上，使用Docker部署时，省去了配置服务器系统环境的工作，提升了部署运维效率。

## RESTful概述

RESTful可以认为是一种交互风格，全称为Representational State Transfer。其最早由Roy Fielding在2000年提出，主要是指Web程序应用过程中，符合REST设计原则的服务器与客户端以无状态的交互请求为基础，客户端发送信息请求时要附带服务器理解需求所需的信息[36]。而提供接口的服务端往往涉及到诸多内容的封装，比如，算法、数据库记录、应用程序对象等[37]。在资源使用过程中以URI为唯一地址来源，通过采用资源共享的方式，对服务器、客户端内容进行传输，充分利用了现有的HTTP协议，比如DELETE、PUT、GET、POST。【合成的引用】在目前的Web服务交互方案中较为常用，它倾向于使用更为简洁的方法设计来实现数据交互。RESTful作为分层系统的设计原则，对服务的粒度进行了有效的划分和解耦，清晰的接口风格使其在上下层应用的独立性有良好保障的同时，也降低了系统复杂性。在实现RESTful规范的工程中，良好的前后的分离可以使后端专注与业务逻辑优化，性能瓶颈集中在接口数据的处理上，从而使其在降低服务器交互延迟，与此同时，解耦后简洁的前端设计，也能提升系统整体的实现效果。【合成的应用】

## Eureka注册中心

Eureka是Netflix开发的服务发现框架，其本身是基于RESTful的服务，主要用于定位运行在AWS域中的中间层服务，以达到负载均衡和中间层服务故障转移的目的。SpringCloud将它集成在其子项目spring-cloud-netflix中，以实现SpringCloud的服务发现功能。https://blog.51cto.com/u\_14693305/4767027

Eureka包含两个组件：Eureka Server和Eureka Client。其中，Eureka提供注册服务，当各微服务模块上线时，会在Eureka Server上进行服务注册。之后Eureka Server会采用心跳机制保持与各服务模块之间的连接和监测。Eureka Client为服务模块，其又可划分为服务提供方和服务消费方。服务提供方为实际执行具体业务流程的模块，而服务消费方则是通过RPC进行服务请求。

Eureka注册中心提供了一种统一构建和监管服务集群的解决方案，提供可视化的Web管理平台，整合集群资源。

## Ribbon负载均衡

Ribbon是一种为客户端提供负载均衡的开源工具。在SpringCloud中，Ribbon被集成在Netflix发布的中间层，主要通过负载均衡算法对用户的请求进行分流负载。其负载均衡算法包括Round Robin Rule（Ribbon默认使用的负载均衡算法）：对于相同类型不同数量的服务节点，按照轮询机制分配服务任务；Availability Filtering Rule：若服务节点3次连接失败，则将其设置为短路状态，首次短路时间为30秒，后续按几何倍率增加；Weighted Response Time Rule：每个服务节点配置一个权重值，服务节点响应时间越长，权重值就越小，通过动态调整各服务节点的权重值来实现负载均衡；Zone Avoidance Rule：通过划分服务节点所在区域，来实现轮询访问；Best Available Rule：忽略短路的服务节点，选择并发数较低的服务节点；Random Rule：随机选择服务节点；Retry Rule：重试机制选择服务节点。

对于开发者来说，Ribbon已经集成在Netflix之中，在SpringCloud框架实现的项目中配合Eureka注册中心使用。使用时无需进行繁琐的配置工作，如果对负载策略（算法）无特殊的需求，仅需要通过@LoadBalanced注解来使用其负载均衡的服务。

## 本章小结

综上所属，基于微服务架构的在线云盘系统，侧重于后端业务模块的设计与实现。微服务架构的设计理念和优势主要表现在高扩展性、高灵活性方面。根据架构需要，各模块接口均采用RESTful规范实现数据交流，基础配套设施中融入了Eureka注册中心，并通过其内嵌的Ribbon服务实现业务模块负载均衡。在云盘功能实现方面，通过FTP协议保障文件传输，借助Token校验机制和Redis缓存机制提高了系统的安全性和访问效率。在部署运维方面，利用Docker容器化技术实现模块环境隔离，快速部署，减少了微服务项目部署的压力。

# 需求分析

云盘系统开发的目的，是能够组织各种硬件资源、软件资源及网络资源，实现用户文件的扩展存储和同步共享。在此基础上还应确保该系统具备数据安全性、功能拓展性及较高的负载能力。本章主要从功能性和非功能性两个角度，对云盘用户的需求进行挖掘分析。

## 系统概述

云盘是基于云计算技术的在线存储服务。云盘的发展主要体现在以下六个方面：海量存储、多终端支持、安全性提升、集成各种附加功能、智能化、用户隐私保护。

云盘的用户量和文件容量逐渐扩大。可采用微服务架构设计云盘，使其具备较高的灵活性和负载能力。当用户并发量增加时，可以临时部署上线额外的服务模块，以响应更多用户的请求，实现负载均衡；当用户并发量紧缩时，也可适当下线已有的服务模块，以节约成本，提高业务模块利用率。根据用户并发量、文件传输流量等数据反馈，可及时做出部署及运维调整，实现需求与供给的动态平衡。

云盘系统在微服务架构的基础上，前端可对接多平台、多终端，实现跨平台一体化服务。云盘能够在多种平台上使用，如电脑、手机、平板等，而业务实现和数据管理均来自同一大后端体系。

云盘作为用户数据管理平台，安全性的要求也在逐渐提高。应充分考虑数据加密方式及手段，保证数据安全，保护用户隐私。

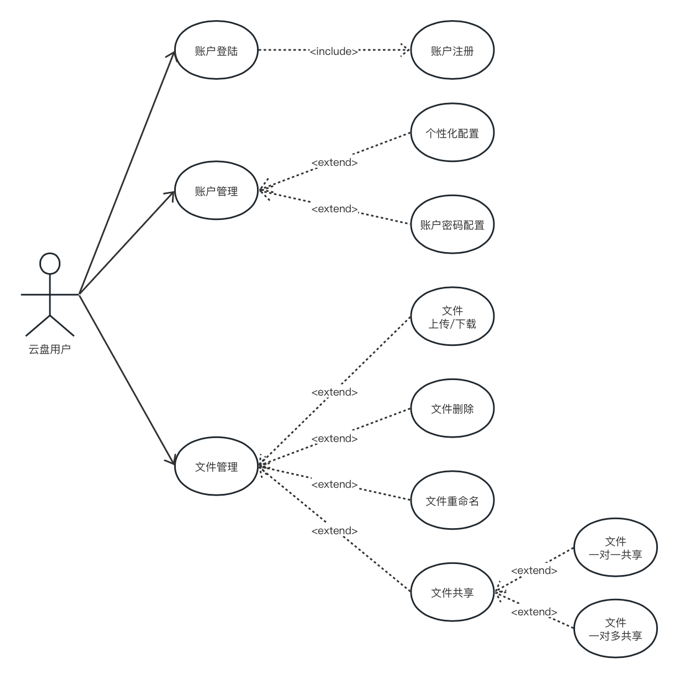
云盘的发展会不断衍生出各种附加功能，如新闻资讯、在线协作、笔记管理等。可采用微服务架构设计云盘，当新的需求出现时，业务模块可做出灵活调整，新的业务模块也可以轻松并入系统集群。

在人工智能的发展浪潮下，云盘也出现了智能化的服务需求，如自动识别文件类型、智能搜索等。充分利用人工智能的应用技术，可为云盘系统提供更为强大的生命力。

## 功能性需求

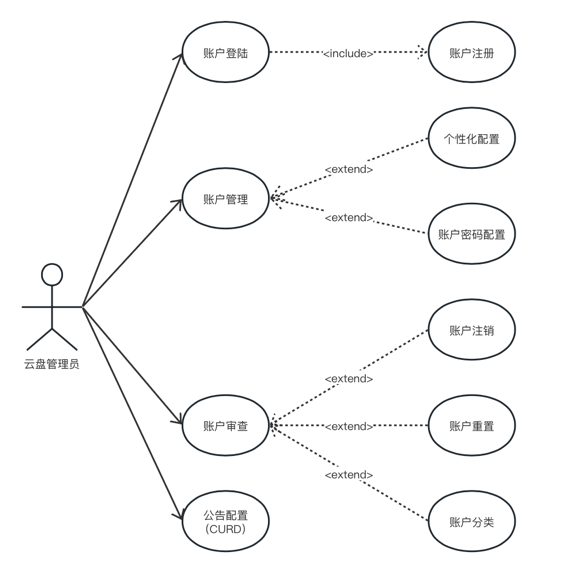
在线云盘需要具备三大基础功能：用户管理、文件管理、个性化配置。用户管理应具备信息管理、身份识别认证和权限管理功能。文件管理应具备文件存储、文件传输和文件共享功能。个性化配置应包括头像配置、昵称配置等用户账户个性化方案。

云盘用户需要进行身份识别认证（账户注册/登入/登出），进入云盘空间后可以实现账户管理（头像/名称的个性化配置、密码管理）、文件管理（CRUD）、文件共享（文件共享链接配置）。



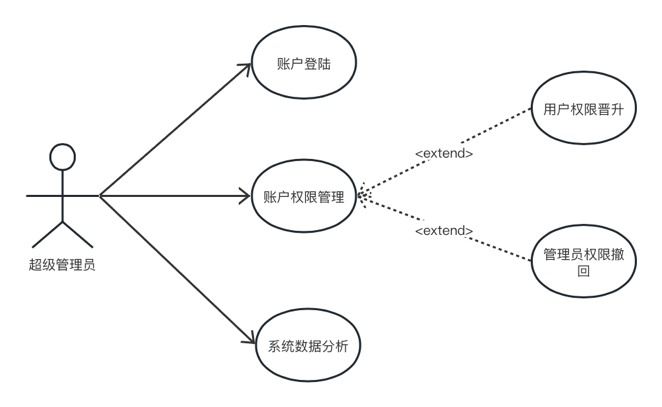
（云盘用户用例图）

云盘管理员需要进行身份识别认证（账户登入/登出），进入云盘空间后可以实现账户管理（头像/名称的个性化配置、密码管理）、用户账户审查（CRUD）、云盘公告管理（CRUD）。



（云盘管理员用例图）

超级管理员需要进行身份识别认证（账户登入/登出），进入云盘空间后可以实现账户权限管理（用户账户/管理员账户）、系统数据分析。



（超级管理员用例图）

## 非功能性需求

身份识别认证服务，利用token机制生成身份标识符。其它服务模块提供服务前，使用token认证代替登陆时的账户认证，使得服务模块与账户数据库解耦，利用Redis缓存机制实现用户身份数据高效访问，提升数据交互效率，进而降低关系型数据库在用户高并发访问时的压力。

文件管理模块将文件数据与文件元数据分离，利用文件控制块（FCB）检索文件存储位置。FCB可进一步优化，分离出检索关键词（借鉴UNIX文件管理中的i节点），提升文件检索性能。

文件存储服务器与业务服务器分离。前后端采用Ajax实现数据交流，后端业务服务器与文件服务器采用FTP协议建立传输通路实现数据交流。

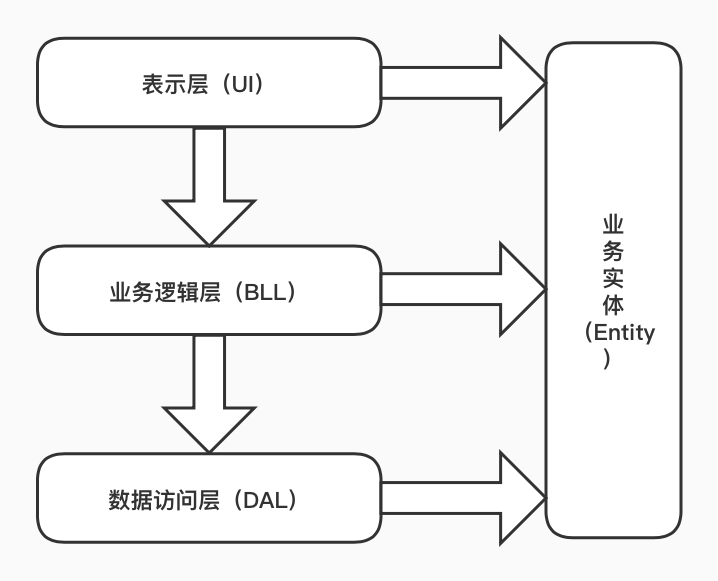
## 本章小结

本章主要介绍了云盘产品主要的用途和用户群体按不同层次的划分。其中，普通用户是云盘系统的最大最主要用户群体，而文件上传下载又是云盘系统最主要最核心的服务方案。此外，用户群体还扩展出管理员以及超级管理员两种角色，服务方案则拓展出文件共享和账户个性化配置等多种特色方案。在功能需求得到满足的同时，还需要考虑存储效率、系统安全等多方面因素，由此产生了多种非功能需求，服务于功能性需求。非功能需求和功能需求的结合使能，才能使得系统更加丰富和完善。

# 系统设计

## 系统架构设计

云盘系统前后端分离，项目前端主要包括UI交互服务，作为表示层；后端采用微服务架构设计，架构内部分为业务逻辑层和数据访问层，共同实现业务模块和存储模块。前端根据用户操作，向业务逻辑层发出数据请求，业务服务根据请求类型和功能点的不同，向数据库进行数据交互并对数据进行处理，结果反馈给前端。



（后端模块架构图）

业务（服务）模块包括：账户管理服务、文件管理服务、公告管理服务、身份认证服务、文件共享服务、数据分析服务。前三者为基础服务，后三者为进阶服务，即后三者依赖于前三者并融入了自己的独有的业务逻辑。

存储模块包括：关系型数据库（MySQL）用于存储账户数据、文件元数据、公告数据、数据分析数据；非关系型数据库（Redis）用于存储token令牌、账户数据（即时）、文件元数据（即时）；FTP服务器用于存储用户文件数据。

## 基础服务设计

### 账户管理服务设计

账户管理属于基础服务，主要负责维护MySQL和Redis中的账户数据。该服务利用账户控制块（ACB）管理账户数据。ACB对象包括以下属性：账户ID、密码、昵称、头像、权限、有效位、账户容量。

具体服务包括：创建账户（账户ID、密码、权限、有效位、账户容量为非空属性）、修改账户（密码、昵称、头像、权限、有效位、账户容量其中之一）、查看账户（所有属性其中之一）。上述各操作应提供独立的对外接口（API）且各接口间仅可存在对账户ID的依赖关系。该服务提供的接口可能会被前端请求直接调用，或是被后端其它服务模块组合调用。

创建账户时，ACB对象同时录入MySQL及Redis。查询账户时，优先查询Redis，Redis未命中时再查询MySQL（同时将该数据录入Redis）。修改账户时，若查询Redis命中则仅修改Redis中的相关数据，若查询Redis未命中则立即查询MySQL，将相关数据修改后录入Redis。

显然，该服务还需要考虑Redis-MySQL置换算法。当Redis账户数据容量上限时，采用LRU置换算法进行ACB置换。若Redis中的ACB被修改过，则应当在置换前将其写入MySQL，确保数据一致性。

### 文件管理服务设计

文件管理属于基础服务，主要负责维护MySQL和Redis中的文件元数据及FTP服务器中的文件数据。该服务利用文件控制块（FCB）管理文件元数据。FCB对象包括以下属性：文件ID、文件名、文件大小、文件类型、更新日期、文件下载次数、文件路径。

具体服务包括：创建FCB（所有属性均为非空属性）、删除FCB、修改FCB（文件名、文件类型、更新日期、文件下载次数、文件路径其中之一）、查看FCB（所有属性其中之一）。上述各操作应提供独立的对外接口（API）且各接口间仅可存在对文件ID和文件名的依赖关系。该服务提供的接口可能会被前端请求直接调用，或是被后端其它服务模块组合调用。

创建文件时，FCB对象同时录入MySQL及Redis。查询文件时，优先查询Redis，Redis未命中时再查询MySQL（同时将该数据录入Redis）。修改文件时，若查询Redis命中则仅修改Redis中的相关数据，若查询Redis未命中则立即查询MySQL，将相关数据修改后录入Redis。

该服务考虑的Redis-MySQL置换算法可类比上文账户管理服务。

在FTP服务器中，按预设的文件系统存放文件数据。此时，该文件在FTP服务器内的路径地址称为“FTP偏移地址”，而该FTP服务器的IP地址称为“FTP地址”。该服务建立传输通路，使用的地址（URL）即“FTP地址+FTP偏移地址”。所以，文件上传时，该服务为待传文件创建FCB并将新的URL即文件元数据写入FCB中，然后再将文件传输到指定地址（URL）进行存储；文件下载时，该服务通过检索待下载文件的FCB（或i节点），获得文件存储地址（URL），再建立地址与服务间的传输通道，进行文件调取。

### 公告管理服务设计

公告管理属于基础服务，主要负责维护MySQL中的公告数据。该服务利用公告控制块（NCB）管理公告数据。NCB对象包括以下属性：公告ID、标题、公告内容、修改人、更新日期、有效位。

具体服务包括：创建NCB（所有属性均为非空属性）、修改FCB（标题、公告内容、修改人、更新日期、有效位）、查看FCB（所有属性）。上述各操作应提供独立的对外接口（API）且各接口间仅可存在对公告ID的依赖关系。该服务提供的接口可能会被前端请求直接调用，或是被后端其它服务模块组合调用。

## 进阶服务设计

### 身份认证服务设计

身份认证服务属于进阶服务，依赖于账户管理服务。其主要负责用户登陆环节的身份认证，生成token并辅助Redis维护token生命周期。

该服务需要调用账户管理服务的相关接口（API），获取相关账户数据，进行数据比对，完成身份认证。认证成功，则使用MD5算法对账户数据及相关参数（日期，盐值）进行单向加密，生成token发放给用户并存入Redis中的token记录表；认证失败，则拒绝发放token并给出反馈。不同权限的账户token生命周期不同，该服务需要在Redis中设定token生命周期（有效期）。

### 文件共享服务设计

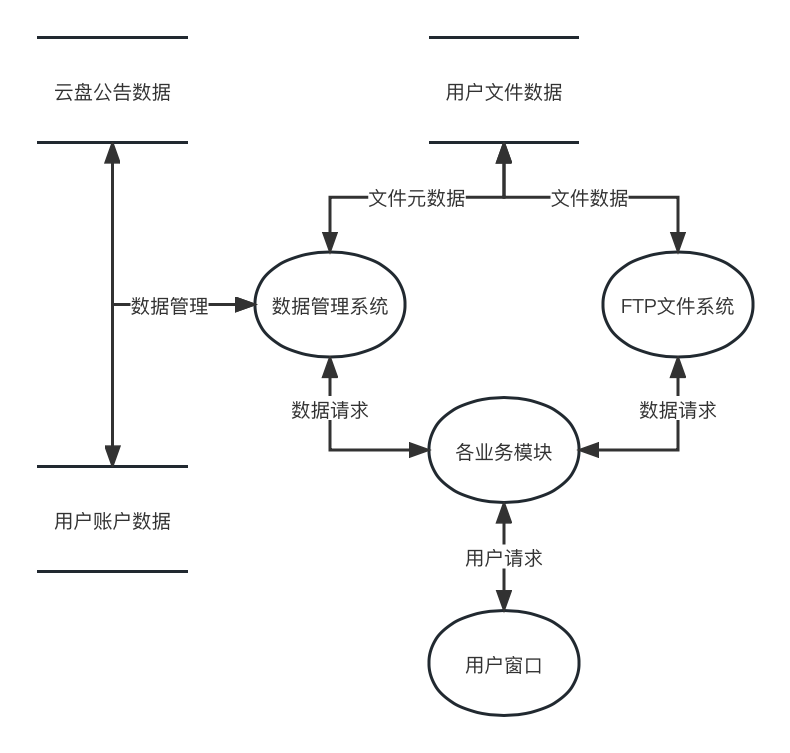
文件共享服务属于进阶服务，依赖于文件管理服务。其主要负责根据待共享文件生成共享码、识别共享码解析出共享文件。该服务需要设定共享文件与共享码构成的映射表，并存放于Redis数据库中。Redis根据共享形式（只读/可写）的不同分别建立不同的映射表。若为只读共享，该服务仅提供文件下载接口；若为可写共享，该服务还会提供修改文件（包括文件重命名、文件覆盖上传）接口。上述共享码在Redis中的生命周期决定了共享文件的时限，因此该服务还需维护共享码的生命周期。

### 数据分析服务设计

数据分析服务属于进阶服务，依赖于上述所有服务。其主要负责收集分析云盘使用数据。该服务根据时间段归纳总结使用数据，生成数据分析控制块（DCB），存入MySQL数据库，供管理员参考分析。具体来说，DCB主要包括记录ID、记录日期、云盘容量、剩余空间、文件数、用户数、访问量。总的来说，数据分析服务可看作云盘日志，且具备较高的自由度。记录内容及分析策略可根据管理员需要做出灵活调整，也可融入数据监控的功能，实现云盘资源自动化管理及动态调整。

## 数据设计

系统数据流主要有三个方面：用户账户数据、用户文件数据、云盘公告数据。从上述系统架构可以看出，用户账户数据流和云盘公告数据流仅存在于业务模块和关系型数据库之间，而文件数据流还会存在于业务模块、关系型数据库和FTP文件服务器之间。

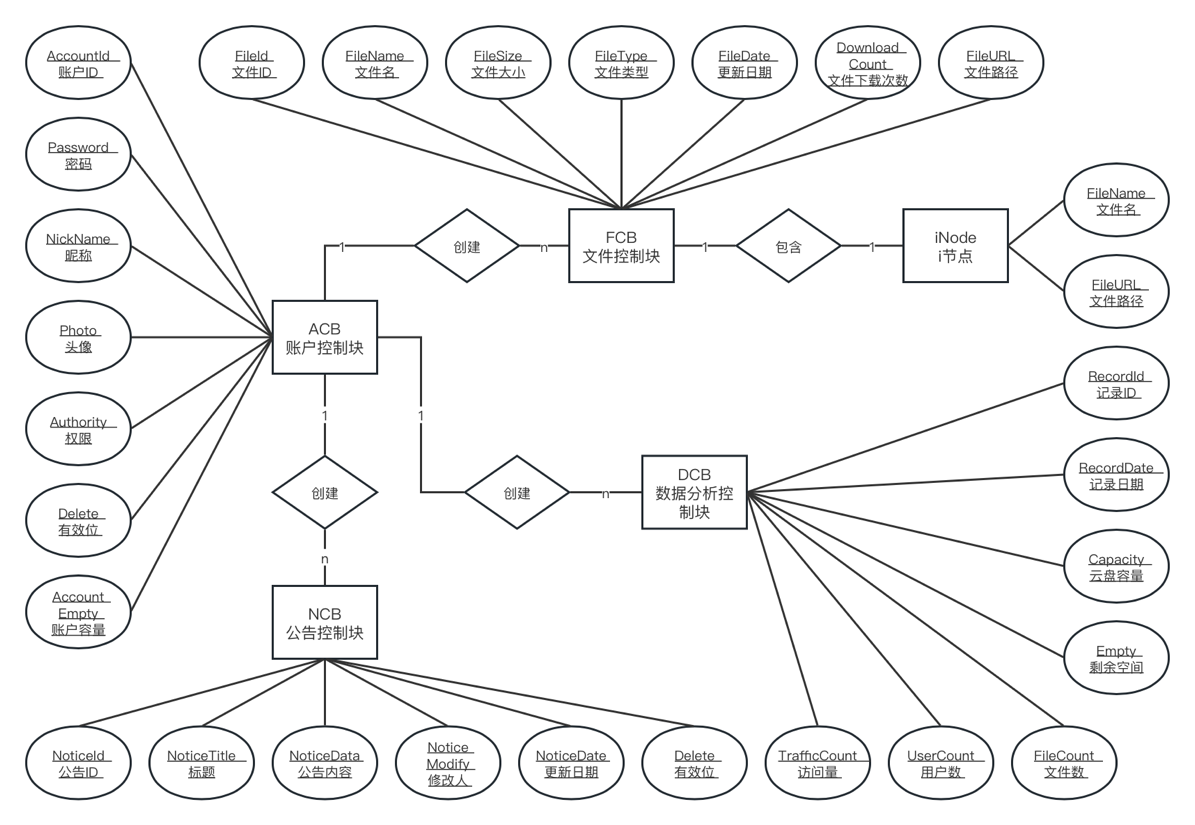


（数据流图）

### 关系型数据库设计

#### 概念模型设计

主要数据对象：账户控制块（ACB）、文件控制块（FCB）、i节点、公告控制块（NCB）、数据分析控制块（DCB）。



(实体-关系图)

#### 逻辑模型设计

根据上述E-R图的描述，可将现有的实体关系转换为对应的逻辑模型。

账户控制块（ACB）可独立构建成账户数据表，表内以账户ID作为关系表主键。

文件控制块（FCB）依据账户数据表而构建，形成文件元数据表，表内以文件ID和文件名共同作为关系表主键，而文件ID作为外键参照账户数据表中的账户ID。

公告控制块（NCB）依据账户数据表而构建，形成公告数据表，表内以公告ID和公告名共同作为关系表主键，而公告ID作为外键参照账户数据表中的账户ID。

数据分析控制块（DCB）依据账户数据表而构建，形成数据分析表，表内以记录ID作为关系表主键，同时作为外键参照账户数据表中的账户ID。

关系表结构

### 非关系型数据库设计

该系统采用Redis作为系统缓存，用于存储账户认证token、高频访问的账户数据及文件元数据。存储对象为token和账户ID构成的键值对、文件控制块（ACB）、文件控制块（FCB）。其中，ACB和FCB采用Hashes数据结构，其结构细节见下：

ACB::[AccountId]

Password [value]

NickName [value]

Photo [value]

Authority [value]

Delete [value]

AccountEmpty [value]

LRU [value]

Change [value]

FCB::[FileId]

FileName [value]

FileSize [value]

FileType [value]

FileDate [value]

DownloadCount [value]

FileURL [value]

LRU [value]

Change [value]

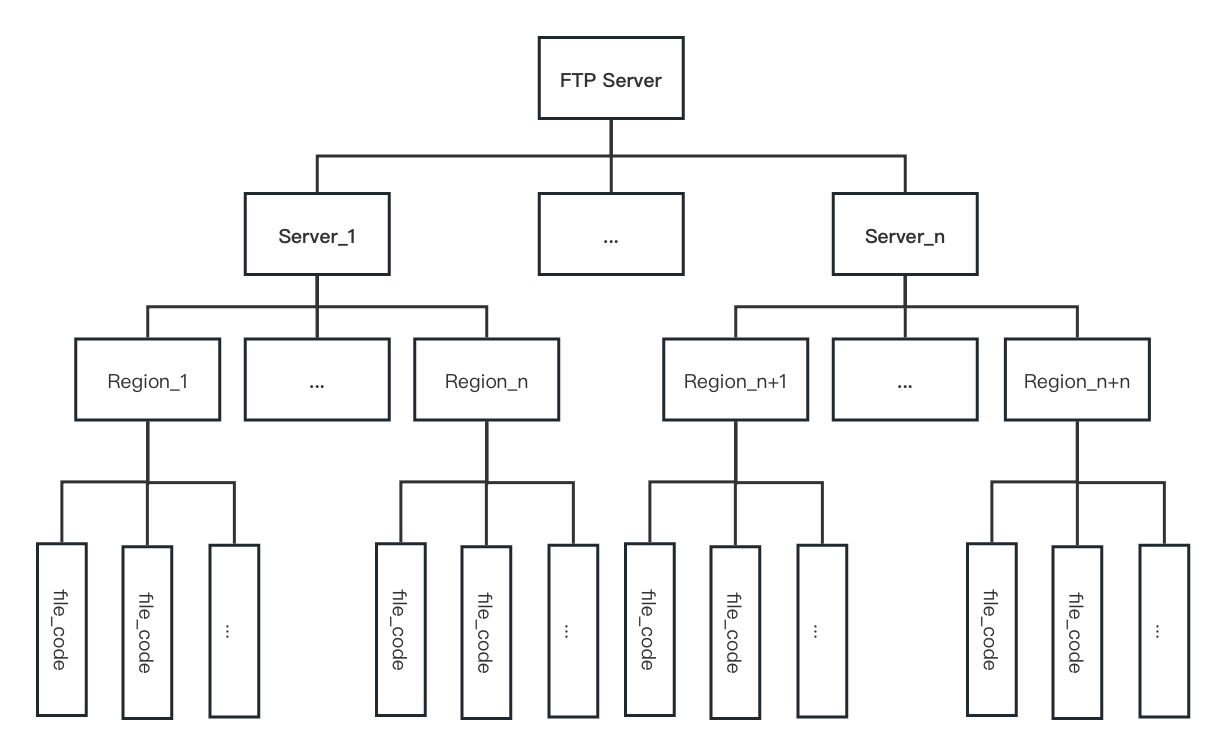
redis 1:ACB缓存 2:token缓存 3:FCB缓存

上述数据均存在特定的生命周期，根据服务模块提供的参数而定。

### FTP服务器设计

文件路径（URL）：FTP服务器IP地址+文件系统内址（偏移地址）

文件系统内址：/cloudbox/region\_[value]/[file\_code]



（FTP服务器文件管理图）

## 本章小结

系统设计从架构到模块，粒度由粗到细，进行了详细的说明。

在服务模块中，调用可能是前端用户发起的直接调用，也可能是后端服务模块发起的依赖调用。若是前端的用户调用，模块在服务前均需要核验其附带的token，而后端模块间的相互调用则不需要进行身份核验。

在存储模块中，MySQL作为持久化数据的主导角色，而Redis作为数据缓存的辅助角色，二者的数据交互逻辑可类比计算机组成中的Cache和主存的关系，同时也充分借鉴的主存使用的页面置换算法和Cache写策略。FTP服务器作为文件数据的存储核心，其结构上充分借鉴了操作系统中文件系统的目录结构以及使用FCB管理文件的策略（在MySQL中还引入了UNIX文件系统中的i节点优化策略），使得文件存储更合理，文件管理更高效。

# 系统实现

系统各模块均是基于SpringBoot框架实现，在框架内集成Mybatis-plus和Redis相关依赖，用于分别对接后台MySQL和Redis数据库，进行数据操控。模块内采用分层设计实现，具体包括：entity实体类、Service业务类、Controller接口类。

entity实体类作为容器接收MySQL关系表数据，类成员变量与关系表中的属性一一对应，必要之处会使用相关注解进行数据约束校验，以满足关系表内的数据结构要求。

Service业务类抽象出业务逻辑接口，并用相应的实现类去编写业务逻辑实现，各接口中所定义的形式参数均按照业务逻辑的需要，利用@Validated注解设定不同的数据格式检查。

Controller接口类利用@PostMapping或@GetMapping注解设定接口URL，接收前端用户的请求和其他模块的远程调用，并利用@ResponseBody注解反馈Json格式数据。

系统各服务模块均遵循RESTful规范，进行数据交流。各服务模块暴露出的接口，共有三种层级：普通用户级接口、管理员级接口、超级管理员级接口。接口层级存在包含关系（即高层级用户可调用低层级接口，低层级用户不可调用高层级接口）。

系统各服务模块间均需要配置特定的访问端口和数据库连接，通用配置参数设置见下图。



（通用配置参数图）

## 账户管理服务实现

该模块用于对账户数据进行管理，共设置3个Controller接口类：Basic Account Controller类、Senior Management Account Controller类、Management Account Controller类以及3个Service业务类（细节从略）：Account Management类、Account Redis Util类、Token Redis Util类。其中，Account Management类负责实现全体账户业务逻辑、Account Redis Util类负责实现账户控制块（ACB）在Redis服务器上的缓存管理、Token Redis Util类负责账户令牌（token）在Redis服务器上的映射和缓存管理。

Basic Account Controller类作为基础账户接口，为云盘普通用户提供账户服务，共提供7个接口（详情见下表）：账户创建、账户登出、账户注销、账户密码修改、账户昵称修改、账户头像修改。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 接口名称 | URL | 请求参数 | 参数格式 | 返回值 | 接口层级 |
| 账户创建 | /createAccount | box | Map<String,String> | Object | basic |
| 账户登出 | /logoutAccount | token | String | Object | basic |
| 账户注销 | /deleteAccount | token | String | Object | basic |
| 修改密码 | /updatePassword | box | Map<String,String> | Object | basic |
| 修改昵称 | /updateNickname | box | Map<String,String> | Object | basic |
| 修改头像 | /updatePhoto | box | Map<String,String> | Object | basic |
| 临时身份认证 | /tempLogin | result | Map<String,String> | Object | basic |

Management Account Controller类作为管理员账户接口，为云盘管理员用户提供账户服务，共提供4个接口（详情见下表）：基础账户列表查看、基础账户密码重置、基础账户容量重置、基础账户注销。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 接口名称 | URL | 请求参数 | 参数格式 | 返回值 | 接口层级 |
| 基础账户列表查看 | /selectBasicList | token | String | Object | Management |
| 基础账户密码重置 | /updateBasicPassword | token、id | String、String | Object | Management |
| 基础账户容量重置 | /updateBasicEmpty | token、id、empty | String、String、int | Object | Management |
| 基础账户注销 | /deleteBasicAccount | token、id | String、String | Object | Management |

Senior Management Account Controller类作为超级管理员账户接口，为云盘超级管理员用户提供账户服务，共提供4个接口（详情见下表）：管理员账户列表查看、管理员账户密码重置、管理员账户注销、账户权限修改。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 接口名称 | URL | 请求参数 | 参数格式 | 返回值 | 接口层级 |
| 管理员账户列表查看 | /selectManagementList | token | String | Object | Senior |
| 管理员账户密码重置 | /updateManagementPassowrd | token、id | String、String | Object | Senior |
| 管理员账户注销 | /deleteManagementAccount | token、id | String、String | Object | Senior |
| 账户权限修改 | /updateAuthority | token、id、authority | String、String、String | Object | Senior |

## 文件管理服务实现

该模块用于对文件数据和文件元数据进行管理，共设置1个Controller接口类：File Controller类以及4个Service业务类（细节从略）：File Management类、File Redis Util类、Token Redis Util类、Ftp Client Util类。其中，File Management类负责实现全体文件业务逻辑、File Redis Util类负责实现文件控制块（FCB）在Redis服务器上的缓存管理、Token Redis Util类负责账户令牌（token）在Redis服务器上的映射和缓存管理、Ftp Client Util类负责对接FTP文件服务器并实现文件数据的上传下载。

File Controller类作为文件管理接口，为云盘用户提供文件管理服务，共提供6个接口（详情见下表）：文件创建、文件名修改、文件类型修改、文件下载、个人文件列表查看、文件删除。guan li

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 接口名称 | URL | 请求参数 | 参数格式 | 返回值 | 接口层级 |
| 文件创建 | /createFile | token、fileNmae、file | String、String、MultipartFile | Object | basic |
| 文件名修改 | /updateFileName | box | Map<String,String> | Object | basic |
| 文件类型修改 | /updateFileType | box | Map<String,String> | Object | basic |
| 文件下载 | /downloadFile | token、fileName、response | String、String、HttpServletResponse | Object | basic |
| 个人文件列表查看 | /readFileList | token | String | Object | basic |
| 文件删除 | /deleteFile | box | Map<String,String> | Object | basic |

## 公告管理服务实现

该模块用于对公告数据进行管理，共设置1个Controller接口类：Notice Controller类以及2个Service业务类（细节从略）：Notice Management类、Token Redis Util类。其中，Notice Management类负责实现全体公告业务逻辑、Token Redis Util类负责账户令牌（token）在Redis服务器上的映射和缓存管理。

Notice Controller类作为公告管理接口，为云盘用户提供公告管理服务，共提供4个接口（详情见下表）：公告广播、公告创建、公告修改、公告有效位修改。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 接口名称 | URL | 请求参数 | 参数格式 | 返回值 | 接口层级 |
| 公告广播 | /readNoticeAll | / | / | Object | basic |
| 公告创建 | /createNotice | token、noticeTitle、noticeData | String、String、String | Object | basic |
| 公告修改 | /updateNoticeData | token、noticeTitle、noticeData | String、String、String | Object | basic |
| 公告有效位修改 | /updateNoticeDeleted | token、noticeTitle、noticeDeleted | String、String、boolean | Object | basic |

## 文件共享服务实现

该模块用于实现用户文件共享功能，共设置1个Controller接口类：Share Controller类以及2个Service业务类（细节从略）：Share Management类、Token Redis Util类。其中，Share Management类通过RPC实现文件共享服务、Token Redis Util类负责账户令牌（token）在Redis服务器上的映射和缓存管理。

Share Controller类作为文件共享接口，为云盘用户提供文件共享服务，共提供2个接口（详情见下表）：主动共享、被动共享。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 接口名称 | URL | 请求参数 | 参数格式 | 返回值 | 接口层级 |
| 主动共享 |  |  |  |  | basic |
| 被动共享 |  |  |  |  | basic |

## 本章小结

系统各业务模块的实现，均严格遵照上述需求分析和设计方案进行。项目开发阶段，每一个模块都作为独立的项目进行初始化创建和依赖环境配置，并配置了通用的数据库连接方案。各模块都遵照单一职责原则进行代码实现，项目结构较为清晰和简单，但是考虑到系统大环境中对于用户的身份认证需求，所以每个模块项目中均配置了独立的身份认证接口，该接口的代码实现存在一定的冗余度。

总的来说，微服务架构的系统实现，均会从在一定的设计冗余性和代码冗余性，借此换取模块的扩展性、灵活性以及低耦合。

# 系统测试

## 测试概述

### 测试目的

本测试报告为“在线云盘”项目的测试用例设计及执行结果，目的在于分析项目的功能性需求和非功能性需求，描述系统是否达到需求分析中的详细要求和结果。预期参考人员包括模拟用户、测试人员、开发人员、项目管理者等。

### 测试背景

“在线云盘”项目开发初步完成的功能有：用户账户注册、登陆、身份校验等；文件上传、下载、共享、重命名等；云盘公告的新建、查看、修改、删除等。

上述功能的交互和使用都基于浏览器UI接口，本次测试任务也将以Web应用测试为主。

### 测试范围

本次测试任务覆盖账户管理模块、文件管理模块、公告管理模块、文件共享模块、数据分析模块。测试用例设计方法采用**等价类划分**以及**边界值分析**的综合运用，受时间因素限制，测试用例的设计和测试计划的展不能完全覆盖项目所有功能点，下文仅展示最为重要且核心的功能点测试结果。

### 测试点总览

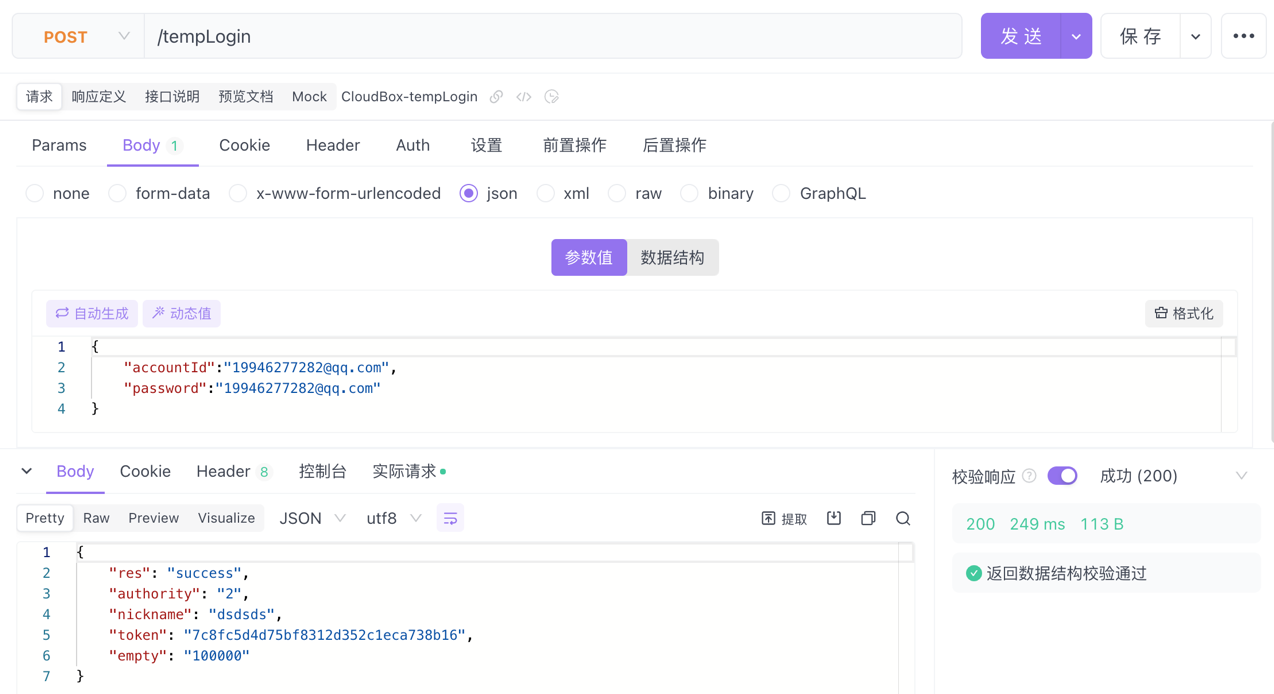
|  |  |  |  |
| --- | --- | --- | --- |
| **编号** | **测试点概述** | **用例个数** | **测试点类型** |
| **1** | 账户管理 | 18 | 功能性 |
| **2** | 文件管理 | 4 | 功能性 |
| **3** | 公告管理 | 13 | 功能性 |
| **4** | 文件共享 | 6 | 功能性 |
| **5** | 身份认证 | 2 | 功能性 |
| **6** | 数据分析 | 5 | 功能性 |
| **7** | Token映射 | 18 | 非功能性 |
| **8** | Redis缓存 | 4 | 非功能性 |
| **9** | Eureka注册中心 | 2 | 非功能性 |
| **10** | Ribbon负载均衡 | 1 | 非功能性 |

## 系统功能测试（例）

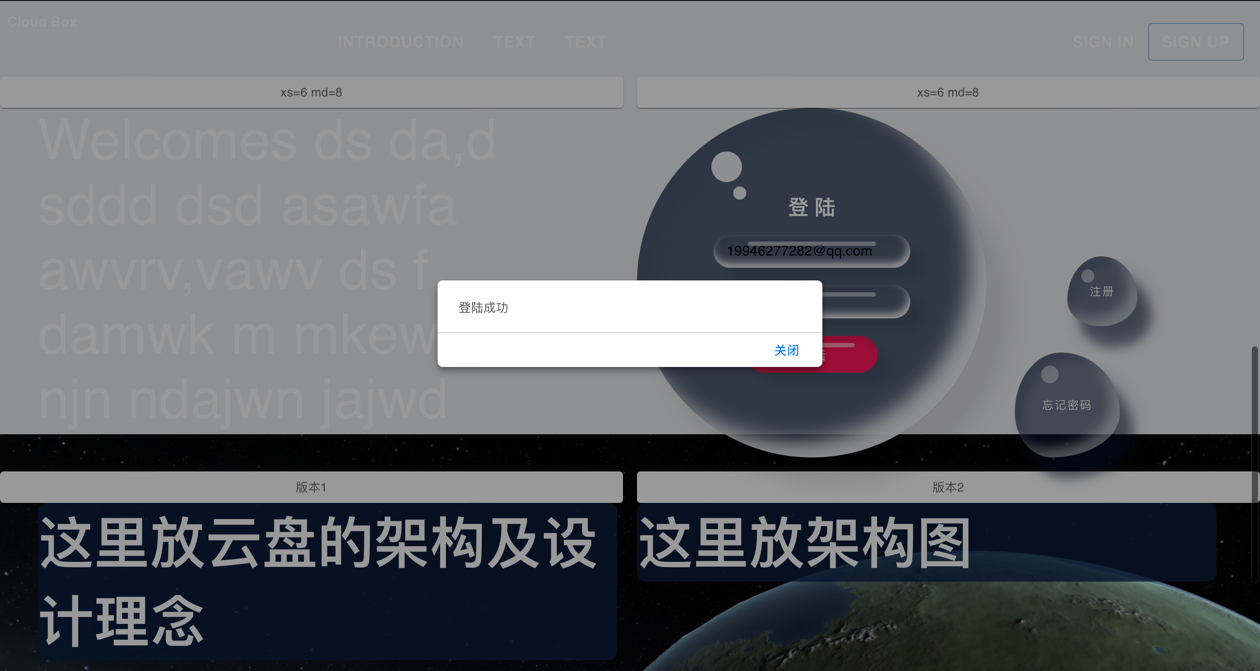
测试用例的设计主要包括9个属性字段（详情见下表）。“用例编号”用于统计测试用例数目以及方便整理归档；“测试点编号”用于标注该测试用例隶属于哪一个测试点；“用例名称”简单概括测试内容；“前置条件”表示在开始本测试之前需要完成的工作；“测试环境”表示测试需要在什么环境条件下进行；“输入参数”表示测试的输入；“预期结果”表示正常情况下的系统反馈；“实际结果”表示根据实际输入而得到的系统反馈；“备注”用于表示测试的额外信息和测试说明。

|  |  |
| --- | --- |
| 用例编号 | 1 |
| 测试点编号 | 1 |
| 用例名称 | 用户账户登陆 |
| 前置条件 | 云盘系统账户已注册；打开浏览器进入云盘系统登陆界面 |
| 测试环境 | Safari、Chrome |
| 输入参数 | UserId=274429919@qq.com；Password=274429919@qq.com |
| 预期结果 | 登陆成功，跳转云盘个人空间 |
| 实际结果 | 登陆成功，跳转云盘个人空间 |
| 备注 | 预期一致，测试通过 |

由于项目使用前后端分离的开发策略，对于功能点的测试，按照前后端数据接口而划分为2个部分：前端测试和后端测试。后端通过接口测试工具完成对后端接口的数据交互测试，保障后端业务逻辑的完整和数据传递格式的规范。前端通过Web界面测试实际的用户操作情况，保障项目最终落地时的使用效果。



(后端接口测试图)



（前端Web界面测试图）

## 系统非功能测试（例）

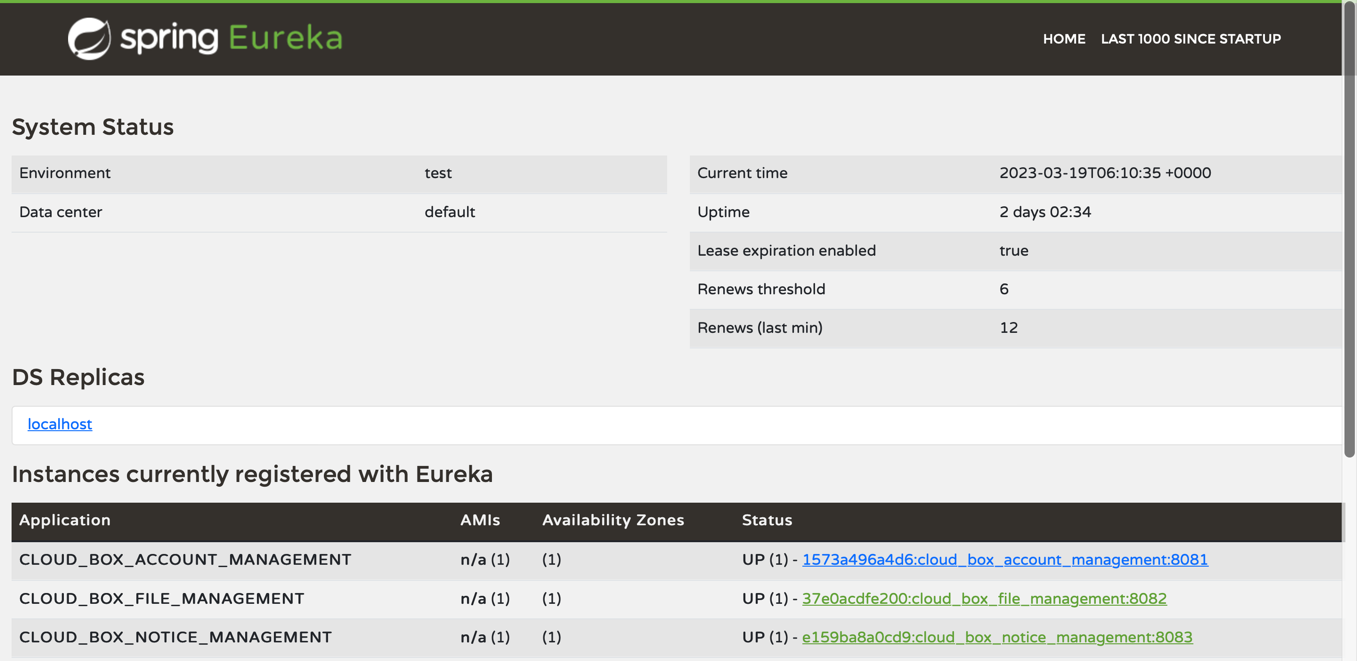
非功能测试的内容主要集中在项目后端，测试内容通过服务端控制台、业务模块输出日志等方式进行反馈。以下通过2个例子分别说明Redis缓存的测试结果和Eureka注册中心的测试结果。

|  |  |
| --- | --- |
| 非功能测试编号 | 1 |
| 测试概述 | Redis缓存账户控制块 |
| 测试详情 | 1. 检查Redis账户缓存库为空  2. 对特定账户进行登陆操作  3. 再次检查Redis账户缓存库发现该账户数据缓存  4. 等待10分钟后执行步骤3，账户数据缓存已清除 |
| 测试结果 | 符合实际预期，测试通过 |

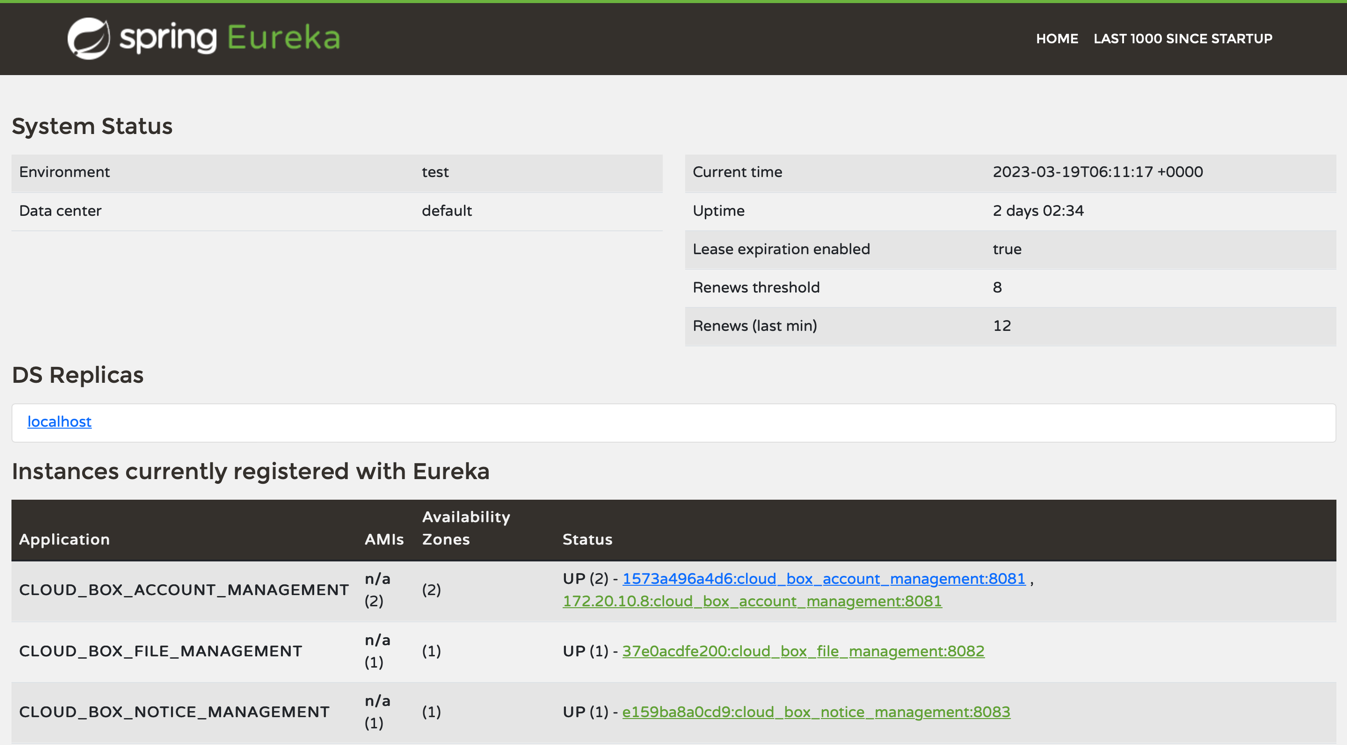


(Redis账户数据缓存测试图)

|  |  |
| --- | --- |
| 非功能测试编号 | 2 |
| 测试概述 | Eureka注册中心：发现服务 |
| 测试详情 | 1. 检查注册中心当前服务类型及数目  2. 重新部署新的账户管理服务  3. 对比注册中心服务类型及数目 |
| 测试结果 | 符合实际预期，测试通过 |



（注册中心图1）



（注册中心图2）

## 本章小结

本次测试计划覆盖了全部的功能性需求点和部分非功能需求点，其中对于微服务架构系统的负载均衡的相关压力测试，因时间限制和进度要求并未展开进行。对测试计划及其结果做出以下分析。

功能性测试方面：用户账户管理的诸项业务，在功能层面上均能很好的完成，不会出现因用户输入非法参数而造成系统报错、崩溃的情况，但是仍有不足，具体体现在对与非法输入提示框消息固定，不具备良好的引导性。用户文件管理及共享业务，在功能层面上均已实现，并未出现文件管理错误或意外，但是在前端UI交互界面上不是很友好，使用体验欠佳。

非功能性测试方面：用户token数据验证服务，在功能层面上来说，始终存在于系统后端，对于用户来说感知不到它的存在，并且在基本测试中也能完成任务，并未发现bug。Redis缓存能够根据账户数据和文件元数据的使用频度，进行缓存并维持一定的生命周期，在生命周期内的相关数据请求操作，可直接由Redis缓存服务器完成，无需MySQL介入。Eureka注册中心能够根据初始化配置发现并记录服务模块的上线，并通过心跳机制感知各服务模块的即时状态，保证服务的可用性。Ribbon提供的负载均衡策略保证服务请求能够较为平均地分配到服务模块，在用户层面透明，而在后端观察者层面，能够及时有效地监控并分析当前系统不同服务模块的负载情况。

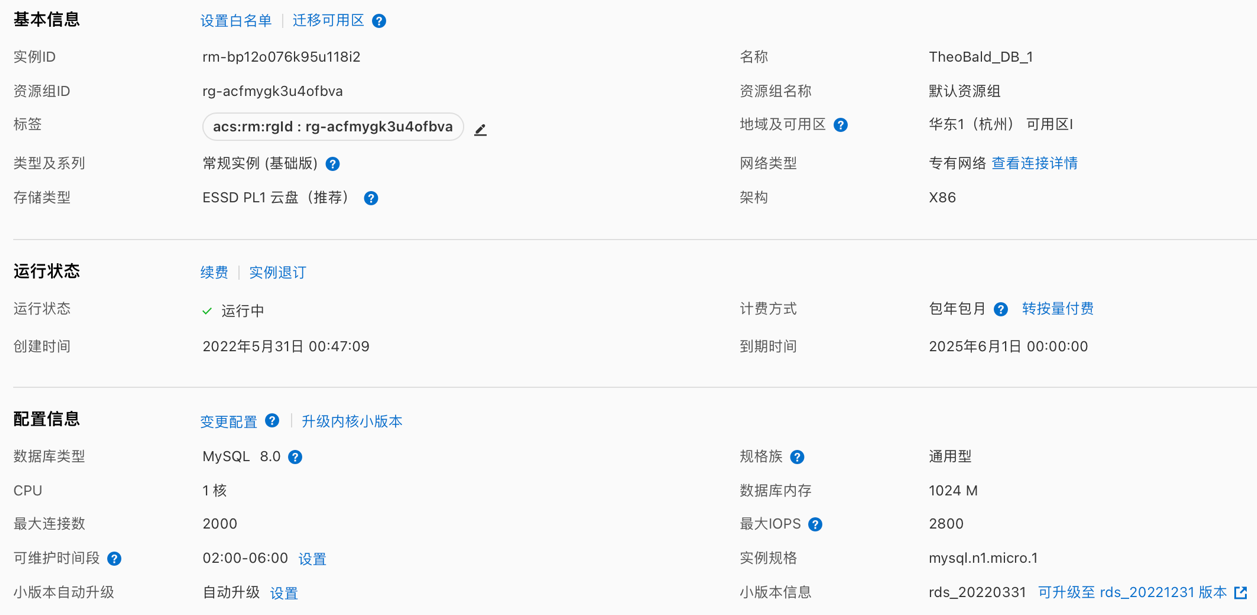
# 系统部署

## 服务器环境概述

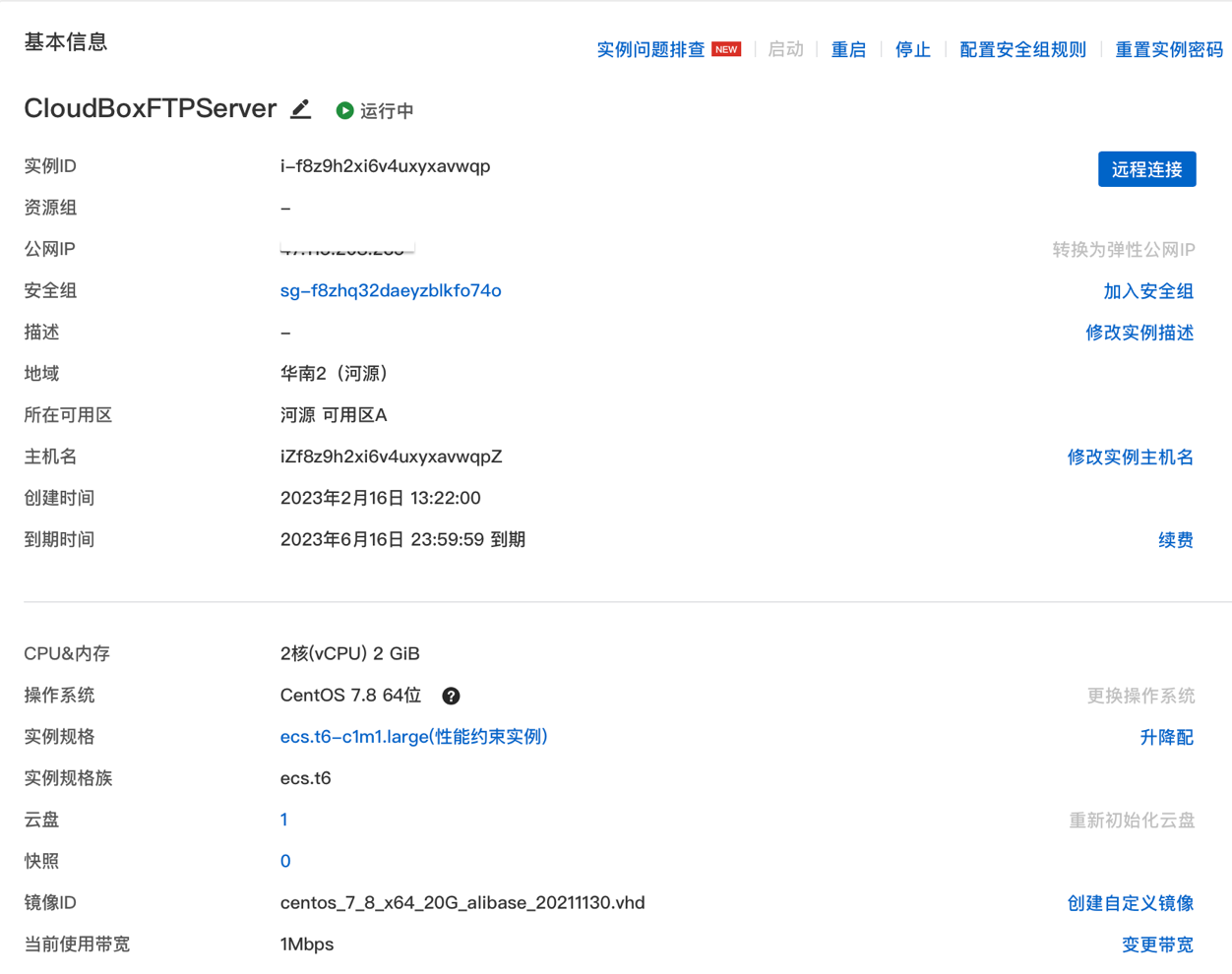
本课题项目计划为每个服务模块至少配置2台服务器，来实现微服务集群的协同工作，但是由于作者经费有限且该项目没有盈利计划，所以目前只有3台云服务器供项目使用。其中，包括一台阿里云RDS服务器；一台阿里云ECS服务器；一台腾讯云轻量级应用服务器。在相同服务器上部署多个项目模块时，均使用配置好的Docker镜像生成各实例容器独立运行，实现环境隔离，以达到逻辑架构上的分布式环境。

放系统部署图

阿里云RDS、ECS作为持久化数据库部署中心，设置了白名单访问策略，仅供腾讯云服务器访问，实现网络环境隔离。

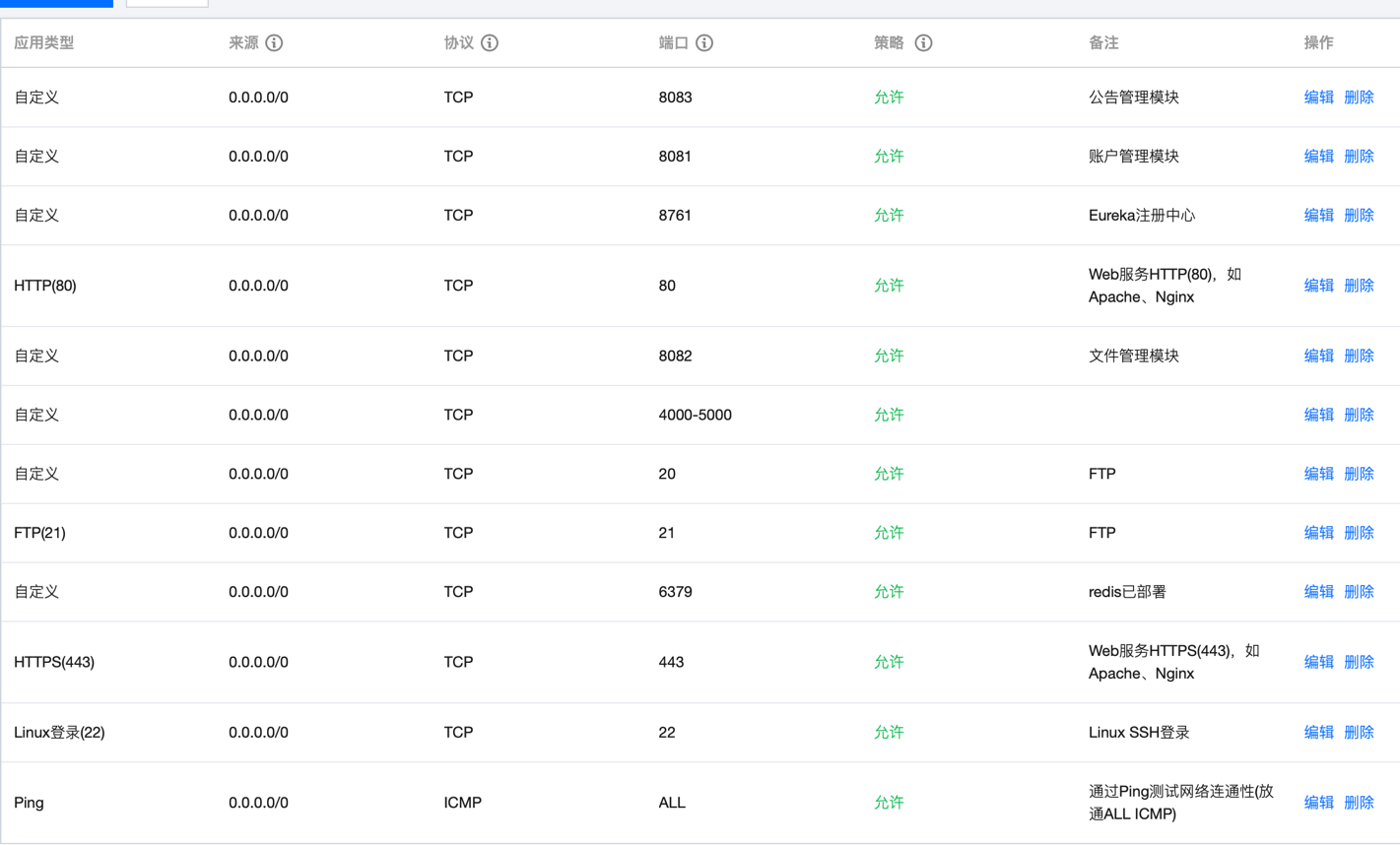


（阿里云RDS实例图）



（阿里云ECS实例图）

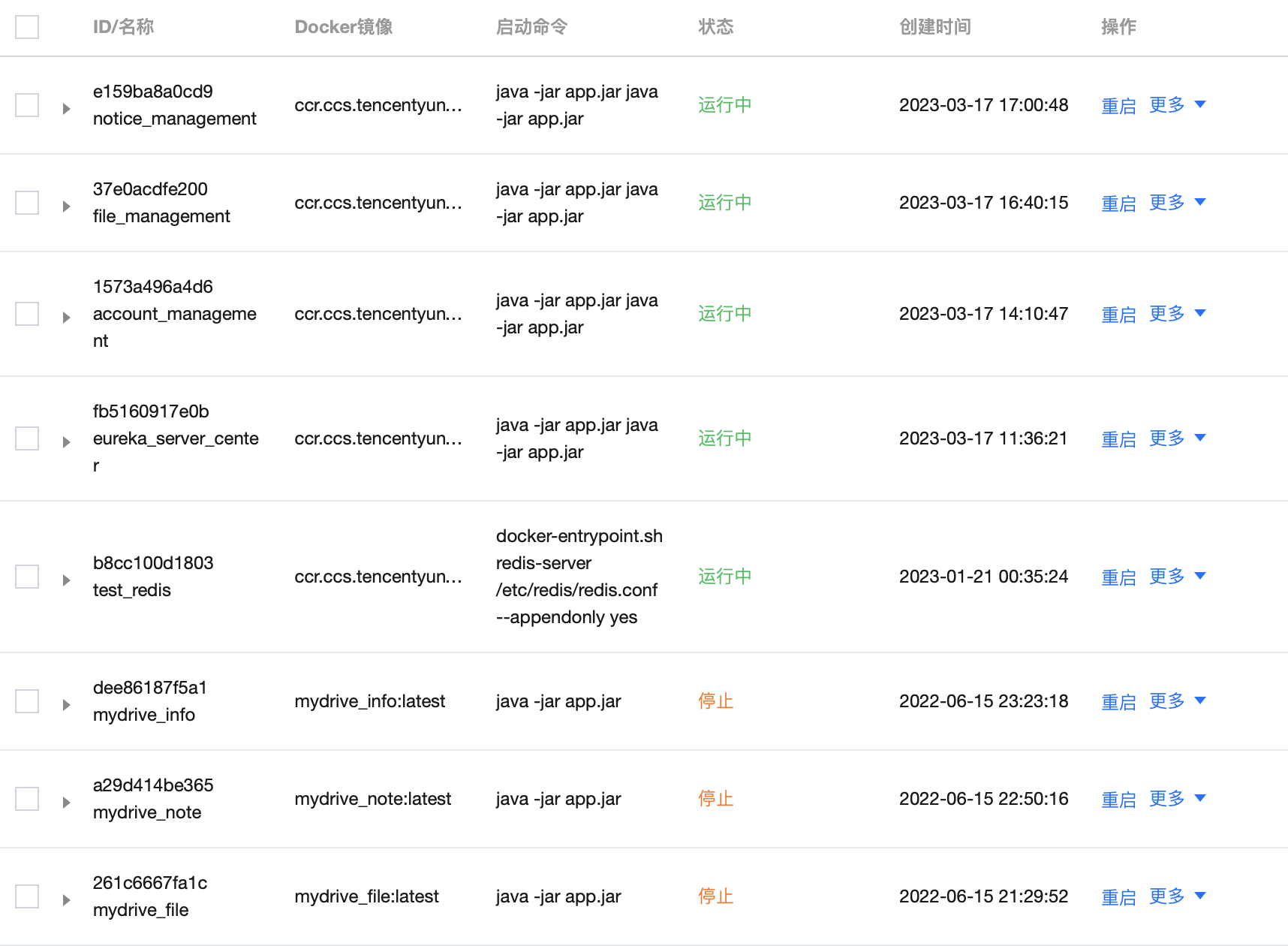
腾讯云轻量级应用服务器作为业务服务部署中心（兼Redis缓存），开放IP地址访问但约束访问端口，开放端口见下图。



（腾讯云轻量级应用服务器端口开放图）

## 服务模块部署

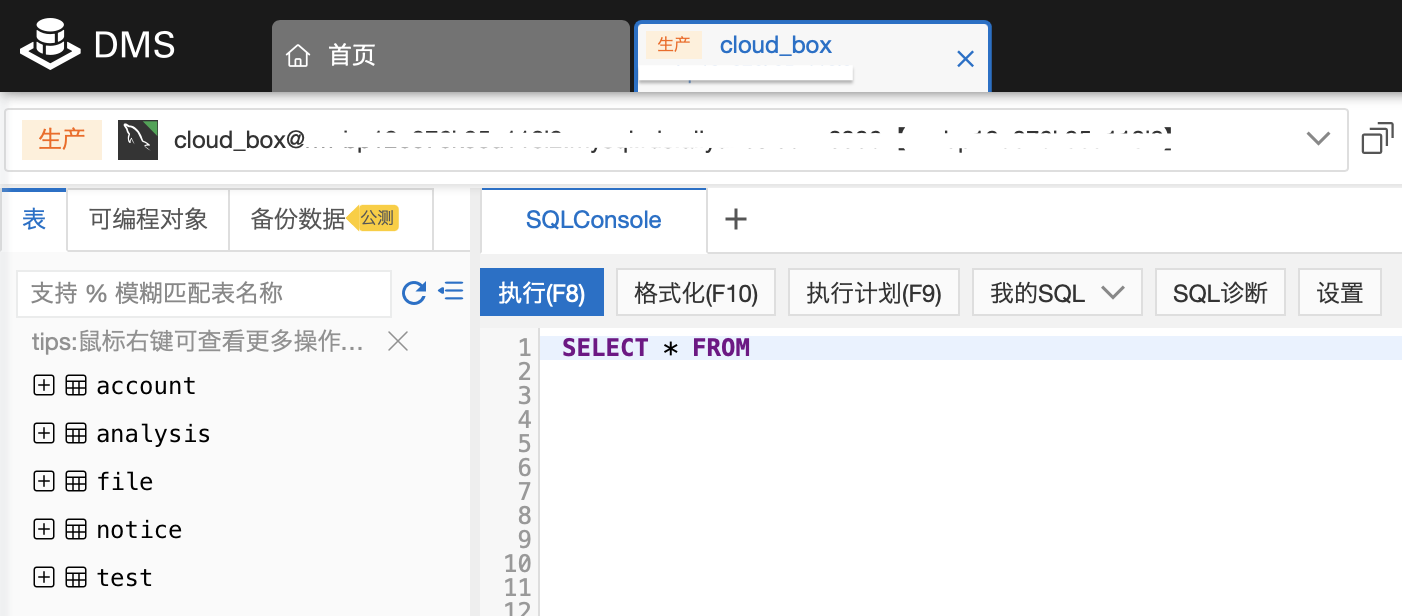
各服务模块主节点，均部署在腾讯云轻量级应用服务器上。受限于硬件资源数量，备用节点可能在本地计算机上灵活补充。云服务器上主要部署：一个Redis缓存数据库、一个账户管理服务模块、一个文件管理服务模块、一个公告管理服务模块、一个Eureka注册中心。



（服务模块部署图）

## 持久化数据库部署

本项目使用到的持久化数据库包括两个：关系型数据库MySQL和FTP服务器数据库。关系型数据库MySQL独占阿里云RDS服务器，使用云服务器预设的MySQL管理系统进行部署。FTP服务器使用阿里云ECS服务器进行搭建，使用vsftp进行文件数据管理和访问控制。



（MySQL数据库管理系统图）



（FTP服务器-文件数据管理图）

## 本章小节

本章主要介绍了项目部署时，所使用的硬件资源和部署策略。项目使用3台云服务器作为后端服务集群。业务服务器上利用Docker容器化部署，来实现逻辑结构上的多节点集群。服务器环境根据实际情况，通过访问白名单和端口约束，保障服务安全。最后通过测试，证明了各单点服务模块能够正常工作，注册中心能够顺利发现集群服务，并对其进行检测管理。

# 总结与展望

## 全文总结

本次课程设计设计并开发了一个“在线云盘”系统，能够具备基本的账户数据管理、文件数据管理、文件共享、公告数据管理、系统数据分析等功能。云盘系统采用前后端分离和微服务架构的设计策略，基于SpringBoot、React、Mybatis、MySQL、Redis、FTP等技术栈结合的综合解决方案，实现了课题项目既定的功能需求和非功能需求。在开发过程中遵循前后端RESTful数据交互规范，进行前后端数据交互。

## 展望

# 致谢

# 参考文献