学校代码：10286
分类号：TP 311
密　级：公开
UDC：004.4
学　号：163699

东南大学

工程硕士学位论文

基于 Ceph 的分布式网盘系统的设计与实现

（学位论文形式：应用研究）

研究生姓名：**胡正东**

导 师 姓 名：**陶军　教授**

金尚　高工

申请学位类别 **工程硕士**　　学位授予单位 **东 南 大 学**

工程领域名称 **软件工程**　　论文答辩日期 **2019 年 05 月 24 日**

研究方向 **软件工程**　　学位授予日期 **20　年　月　日**

答辩委员会主席 **吉 逸**　　评　阅　人 **翟玉庆**

张家凤

20　年　月　日

东南大学

工程硕士学位论文

基于 Ceph 的分布式网盘系统的设计与实现

专 业 名 称： 软件工程

研究生姓名： 胡正东

校 内 导 师： 陶军 教授

校 外 导 师： 金尚 高工

THE DESIGN AND IMPLEMENTATION OF DISTRIBUTED NETWORK DISK SYSTEM BASED ON CEPH

A Thesis Submitted to

Southeast University

For the Professional Degree of Master of Engineering

BY

HU Zheng-dong

Supervised by

Prof. TAO Jun

and

Senior Engineer JIN Shang

College of Software Engineering

Southeast University

May 2019

东南大学学位论文独创性声明

本人声明所呈交的学位论文是我个人在导师指导下进行的研究工作及取得的研究成果。尽我所知，除了文中特别加以标注和致谢的地方外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得东南大学或其它教育机构的学位或证书而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示了谢意。

研究生签名：_____日期：_____

东南大学学位论文使用授权声明

东南大学、中国科学技术信息研究所、国家图书馆、《中国学术期刊（光盘版）》电子杂志社有限公司、万方数据电子出版社、北京万方数据股份有限公司有权保留本人所送交学位论文的复印件和电子文档，可以采用影印、缩印或其他复制手段保存论文。本人电子文档的内容和纸质论文的内容相一致。除在保密期内的保密论文外，允许论文被查阅和借阅，可以公布（包括以电子信息形式刊登）论文的全部内容或中、英文摘要等部分内容。论文的公布（包括以电子信息形式刊登）授权东南大学研究生院办理。

研究生签名：_____导师签名：_____日期：_____

摘要

随着在信息化建设过程中企业对文件数据越来越重视，使用传统的便携式存储设备存储数据在易用性、安全性、成本控制等方面都有明显的不足，网盘的出现解决了这些问题。越来越多的用户把个人文数据存储到网盘中，网盘存储成了一种新的文件存储方式，为用户带来了极大的便利。对于企业用户来说，所存的文件数据可能涉及企业的商业机密等重要信息，而市场上的各类网盘透明度太高，安全事件频发，文件的安全性需要由用户自己负责；同时市面上网盘的传输性能在企业内部环境也不能达到高效的要求；网盘服务存储集群的容灾能力也不可控，对于用户来说文件存储完全透明，一旦服务提供商不再提供存储服务或者存储集群出现不可修复故障的情况，用户存储的文件面临极大的风险。

本文基于 AES 加密算法、MD5 散列算法以及微服务等技术实现了基于 Ceph 的分布式网盘系统。本文的主要工作如下：

（1）分析企业用户对于网盘系统的功能需求和性能需求。根据功能需求梳理出网盘系统的功能模块，分别有存储模块、文件模块、群组/监控模块、通知模块和备份模块。根据需求，制定出系统的测试方案，主要包括各项功能测试以及性能测试，性能测试的指标包括系统响应时间、文件上传下载速率和集群的文件读写速度。

（2）设计基于 Ceph 的分布式网盘系统。从系统的整体架构设计到各个模块中具体功能的设计。在文件模块中，优化了文件的存储结构，提高文件存储的效率。针对不同的文件大小采取不同的文件上传策略，提高文件上传的性能。在安全方面对文件使用非对称加密算法对文件加密，保证文件数据存储的安全性。在团队管理和监控模块中，在基于角色权限的模型上引入对单文件的权限控制，更精细的管理用户的权限。在备份模块中，为了应对系统级故障和灾难，采用增量传输的策略跨机房备份文件。在通知模块中，设计了钉钉通知和邮件通知的功能，满足系统通知功能的及时性要求。

（3）实现并测试了分布式网盘系统。根据各个模块的设计实现各项功能，并对网盘系统进行功能测试和性能测试，给出各项测试的结果。

本文设计与实现的基于 Ceph 的分布式网盘系统已经在企业内部上线使用，目前满足了用户对于企业网盘的功能需求和性能需求，在企业网络环境下的实际应用效果高于市场上的其他网盘。

关键词：Ceph 文件系统，分布式存储，网盘，企业信息化

ABSTRACT

With the increasing emphasis on file data in the process of enterprise information construction, the use of traditional portable storage devices has obvious deficiencies in use, security, and cost control. And the emergence of cloud storage solves these problems. Network disk storage has become a new way of storing files, which brings great convenience to users. For enterprise users, various types of network disks in the market is too pellucidity, the transmission performance of the network disk cannot meet the requirements, and the security of files and the disaster tolerance of storage clusters are uncontrollable.

This thesis implements a Ceph-based distributed network disk system based on AES encryption algorithm, MD5 hash algorithm and micro-service Architecture Development Method. The main contents of this thesis are summarized as follows:

(1) Thoroughly Study Ceph's distributed file system, and analyze enterprise users' functional requirements and performance requirements for the network disk. According to the functional requirements, the function modules of the network disk system are respectively combed 6 modules. There are a storage module, a file module, a group master/monitoring module, a notification module and a backup module. According to performance requirements, draw up system test indicators, including system impact time and file upload and download indicators.

(2) Design a distributed network disk system. Designs are from the overall architecture design of the system to the design of specific functions in each module. In the file module, the storage structure of the file is optimized, and the efficiency of file storage is improved. Different file upload strategies are adopted for different file sizes to improve the performance of large file uploads. And encrypt the file to ensure the security of the file. In the team management and monitoring module, control of individual files is introduced on the model of role permissions. In the backup module, in order to deal with the system and faults and disasters, the incremental transmission strategy is used to back up files across the equipment room. In the notification module, ding notification and mail notification is designed.

(3) Implemented and tested the distributed network disk system. According to the function design of each module realize all functions. According to Perform functional tests and performance tests on the network disk system, give the results.

The Ceph-based distributed network disk system designed and implemented in thesis has been used online in the enterprise. At present, it satisfies the user's functional requirements and performance requirements for the enterprise network disk. The actual application effect in the enterprise network environment is significantly higher than that of other network disk.

Keywords: Ceph file system,distributed storage, network disk, enterprise information

目录

缩略词表

缩略词	英文全称	中文
LGPL	Lesser General Public License	宽通用公共许可证
IDC	Internet Data Center	互联网数据中心
RPC	Remote Procedure Call	远程过程调用
MD5	Message Digest Algorithm MD5	消息摘要算法
MDS	Metadata Server	元数据服务器
CRUSH	Controlled Replication Under Scalable Hashing	受控复制的分布式哈希算法
OSD	Object-based Storage Device	对象存储设备
DES	Data Encrytion Standard	数据加密标准
3DES	Triple DES	三重数据加密算法
AES	Advanced Encryption Standard	高级加密标准
IO	Inout/output	输入输出
CPU	Central Processing Unit	中央处理器
HTTP	HyperText Transfer Protocol	超文本传输协议
TCP	Transmission Control Protocol	传输控制协议
QPS	Query Per Second	每秒查询率
API	Application Programming Interface	应用程序编程接口
LADP	Lightweight Directory Access Protocol	轻量目录访问协议
UUID	Universally Unique Identifier	通用唯一识别码
CFB	Cipher feedback	密码反馈
FTP	File Transfer Protocol	文件传输协议
IOPS	Input/Output Operations Per Second	每秒读写操作的次数

第一章 绪论

1.1 研究背景

在互联网技术快速发展的时代，随着人工智能技术在各行各业有越来越多的场景应用，而这些大数据和人工智能应用依赖和产生的数据呈现出爆发增长的态势，数据的存储就成了束缚业务发展的桎梏，同时个人用户存储的文件数据在时代背景下也越来越多。传统的个人存储主要使用 U 盘，TF 卡以及移动硬盘等实体存储介质，这种物理介质存储具有携带不方便、价格不菲和容量可扩展性差等缺点，这些已经不能满足互联网时代人们对存储快、容量大以及即买即用的要求。所以存储的发展已经慢慢从购买存储设备发展到购买存储服务[1]的方式。云存储正是解决数据存储服务问题的一项技术，在用户使用存储服务的过程中，存储设备对于用户而言是完全透明的，用户不会感知到底层实际存储设备的存在即可使用数据的存储管理功能。对于企业用户来说，企业日常的生产运营活动都不可避免的需要用到文件数据管理的功能，目前也有一些存储服务提供商比如阿里云存储、百度云存储以及腾讯云存储等厂商提供存储服务，但是这些服务提供商并不能提供完全适合企业的服务，比如腾讯云存储和阿里云存储只能提供纯后端的存储服务，而百度云存储提供的服务又不能完全满足企业的需求。基于现有的情况，企业致力于构建一套安全可靠的分布式网盘系统，以提高生产和管理效率[2]。

本课题的项目背景是基于所实习企业信息化建设的过程中，出于安全和协同工作的需求，构建了一套基于 Ceph[3]存储的分布式网盘系统，充分发挥了 Ceph 作为存储的高性能、高可用和去中心化的特点，使得网盘底层的存储系统健壮高可用。同时在此基础上引入微服务架构技术、加密技术和限流技术等实现了一套安全性高、可扩展性好、成本低的企业网盘系统[4]，降低了使用传统存储中的硬件成本以及出现安全性问题的可能性，助力企业日常运营发展数据的安全。

1.2 国内外研究现状

1.2.1 云盘的发展和研究现状

目前云计算技术日益完善，市面上网盘的快速发展正是得益于此。国内外对网盘进行了广泛的研究，根据过去的网盘速度慢、营运成本高、恢复能力和抗灾能力低以及安全性差等瓶颈的存在，文献[5]讨论了网盘系统的主要组成结构以及其关键技术的研究。文献[6]中则集中介绍了网盘中运用到的云存储技术，并且在数据安全方面介绍了对数据的控制技术，包括对用户的权限控制技术、对存储数据的加密技术和底层的资源隔离技术等。

网盘[7]以存储设备为中心，通过引入分布式文件系统、大规模集群等互联网技术，结合大量的存储设备，使得这些设备以集群的方式组建起来对外提供数据存储服务。

用户可以在客户端上随时随地的通过网络访问自己存储的数据。

与传统的设备存储方式相比，网盘存储系统是由底层存储设备、服务器、系统应用软件以及用户访问客户端等多个组件构成的复杂系统。目前网盘的存储架构主要有四层[8]。首先是存储层，存储层是实际存储数据的物理设备，设备的数量较大的同时分布也比较广泛，由存储设备管理系统统一的管理；然后是基础管理层，基础管理层是系统的基础和核心，该层通过集群和分布式文件系统，实现存储层中的设备统一协同工作，以整体的方式对外提供统一的服务，外部访问的系统对于集群的构建方式和规模无感知。接着是应用接口层，为不同的服务对象提供存储服务，满足不同用户的个性化需求。最后是访问层，在该层次提供用户的认证和权限控制等服务，经过用户认证和权限校验之后，用户即可登陆到网盘系统中，使用网盘提供的数据存储和管理服务。总的来说网盘是利用分布式文件系统、集群应用技术以及网络技术，协调网络中多种类型的存储设备在软件的指挥下协同工作完成任务，并向外输出业务访问功能及提供数据存储软件系统[9]。

目前国外比较典型的的网盘[10]产品有 Google 的 Google Drive、微软的 OneDrive、亚马逊的 Amazon S3、苹果的 iCloud 和 Dropbox 等。现在国内已经有腾讯微云、新浪微盘、百度网盘、360 网盘盘、坚果云网盘等个人网盘服务。目前出现的网盘应用虽然给用户带来了极大的便利，但是也存在一些问题。其中国外的网盘产品访问速度较慢，不适合国内用户使用。其他网盘系统一般出现的问题包括数据存储和传输过程中的安全性、文件丢失等问题。企业和个人用户的关键数据在数据存储和传输过程中的安全性不能得到保证，个人隐私数据和企业的商业机密极有可能遭到泄漏。如果网盘服务提供商停止提供服务或者由于不可抗因素导致存储服务出现意外，那么用户文件丢失极有可能造成巨大的损失。对于企业用户来说，目前的公有网盘传输速度[11]不够快，如果需要提高速度则需要支付巨额费用，对于企业来说这是一项巨额的成本。同时网盘不能和企业的流程与组织架构紧密结合，在不能有效提高效率的同时还需面对极大的风险对于企业来说是一项沉重的负担。

1.2.2 Ceph 的发展和研究现状

Ceph 是一个起源于学术研究课题的开源项目，它是加州大学 Santa Cruz 分校的 Sage Weil 专为其博士论文设计的新一代开源的分布式文件存储系统。Ceph[12]作为开源项目，其遵循 LGPL 开源协议，基于 C++语言开发。目前 Ceph 已经成为最受欢迎的全球开源软件定义存储项目，知名互联网 IT 厂商致力于一起对其开发和发展做出贡献，Ceph 在不同的领域不同的行业均得到了广泛的应用。Ceph 的设计目标就是解决数据存储问题中的规模过大和存储节点的分布式问题。大规模的含义是指能够供 PB 级别的数据的写入和读取，分布式是指存储集群中有成千上万数量的节点协同工作提供服务。如今云计算、大数据和人工智能在中国发展得如火如荼，国内企业存储数据早已进入 PB 量级，数据时代已经来临。

1.3 论文研究目的及意义

对于企业来说，文件数据的重要性无需多言，诸多内部的重要数据资料都有可能涉及企业的业务发展甚至是存亡。使用传统的便携式存储设备[15]的存储方式资料易泄漏、易丢失，资料的传播性不够及时，这种相对于云存储而言落后的方式不适合企业使用。所以，在企业的信息化过程中，面对企业中各种各样的文件数据，如何统一的管理起来是一个问题。在解决该问题的基础上并且能有效的支持协同工作，对于提升企业员工之间的沟通效率、个人的工作效率以及保证企业的文档数据安全对企业是一项极具价值的工作。本课题研究的基于 Ceph 的分布式网盘系统就是为了方便企业用户在公司内部存储、查看、分享和管理文件的系统，同时也是为了解决公司核心数据资产的丢失问题，提供数据云备份[16]的能力。对于文件存储服务来说，服务器可能出现宕机和硬盘故障等问题，导致文件存储服务器无法工作，用户的文件就会出现丢失的情况，那么此时就需要使用分布式的文件存储系统[17]，当某台文件服务器出现故障的情况下，其他文件服务器能够在单节点服务器异常时完成异常机器的摘除[18]，实现高可用的文件存储服务。

1.4 本文主要工作

本课题针对企业的实际需求，设计并实现一种可靠、安全和高性能的分布式网盘系统。本文旨在针对企业文件存储要求的安全可靠、高资源的利用率和文件存储服务的高可用要求，设计一种基于 Ceph 分布式的文件存储解决方案，从而解决企业日常生产运营过程中文件存在哪里的问题，也确保存在网盘系统中的文件的安全性和可靠性。本文的主要工作如下：

（1）总结 Ceph 的发展状态和历程，分析现有的一些网盘的缺点，如文件的安全加密对于用户来说是不可见，服务提供商的灾情防备能力（IDC 机房故障）用户不可控，文件传输速度和效率低等。分析 Ceph 的无中心模式的存储系统，避免存储系统出现单点故障。利用 Ceph 提供的技术特性如集群可靠性，集群可扩展性，数据安全性，接口统一性等来保证存储服务的高可用以及易接入性。

（2）应用微服务软件架构方法[19]，将分布式网盘系统分解为存储模块，备份模块，权限模块，以及业务操作模块（文件的分享，管理，下载等）。模块之间使用 RPC 技术通信，提升整个分布式网盘服务系统的通信性能。

（3）分析并应用文件传输和存储技术。鉴于文件传输的过程中会出现网络抖动或者网络故障等不稳定的因素，采用断点续传技术保障文件传输的效率和完整性。优化文件的存储结构，将文件的目录结构和磁盘实际存储结构分离，减少磁盘在定位具体资源时的损耗。同时在异地备份传输过程中，采用基于文件信息摘要 MD5 算法[20]的增量备份技术，节省传输时间和带宽消耗。

（4）分析并应用限流技术。接口限流是在执行备份功能的时候应用，跨机房备份

对带宽需求较大，为了不影响其他业务服务的正常使用，通过限流机制来保证整个服务器的可用状态。本文系统通过使用基于令牌桶算法的限流技术，把发送端的发送请求限制在一个可控的范围内，控制系统的负载，避免因为负载过高而导致的系统问题。

1.5 论文组织结构

本文一共分为七章，以实现基于 Ceph 的分布式网盘系统为目标，根据本文研究内容，组织结构如下：

第一章为绪论。首先从研究背景以及国内外研究现状出发，对目前的网盘和存储方案进行了分析。之后说明研究分布式网盘对于企业的信息化建设具有重要意义，阐释课题的目的与研究意义，最后讲述本文的主要工作。

第二章介绍分布式网盘的理论基础与相关技术。首先介绍 Ceph 的理论基础；其次介绍现有的各种加密技术以及他们的使用场景；再次，主要介绍令牌桶限流算法；然后介绍微服务架构的特征和优点；接着介绍本文系统使用的服务之间调用方式 gRPC 技术；最后介绍本文系统使用到的 Web 应用框架 Gin 数据访问框架 dbr。

第三章主要内容是分析系统的需求。首先对课题进行研究，介绍了本文系统的整体需求目标。其次分别对各个模块的需求进行分析，确定需求边界。最后分析介绍系统的性能需求。

第四章是系统的概要设计。主要介绍了系统的整体架构设计以及各个模块各个功能的概要设计。

第五章是系统的详细设计与实现。首先是实现 Ceph 存储集群的搭建，然后分别设计与实现文件模块、群组/监控模块、通知模块、备份模块。

第六章是系统测试。从功能测试和性能测试两个方面分别对本文系统进行评估验收，最后得出测试结论。

第七章是总结本文的主要内容以及系统的工程价值，并针对目前存在的问题指出了后续研究工作的方向。

第二章 相关理论基础与技术

第一章对课题背景进行了阐述并分析课题研究目的和意义，最后介绍了本文的主要工作。本章主要对网盘系统涉及到的相关技术及其理论进行介绍，对使用该技术的原因和优势进行分析。

2.1 Ceph 简介

Ceph[21]是一款开源统一的分布式文件存储系统，起源于 Sage Weill 在博士期间的工作，随后 Sage 博士发表论文并将 Ceph 项目贡献给了开源社区。Ceph 项目团队致力于将 Ceph 打造成一款高性能、可扩展的、可靠的分布式文件存储系统[22]。

Ceph 在统一的系统中为用户提供了对象（Object）、块（Block）接口、文件（Files）三种接口[23]，这三种接口分别对应着对象存储服务、块存储服务、文件存储服务。

Sage 博士在他的论文[24]中提出了 RADOS 的概念，提出了基于 RADOS 服务实现大规模的、可靠的、可无限伸缩的 Ceph 存储集群。Ceph 的基本架构如图 2-1 所示。

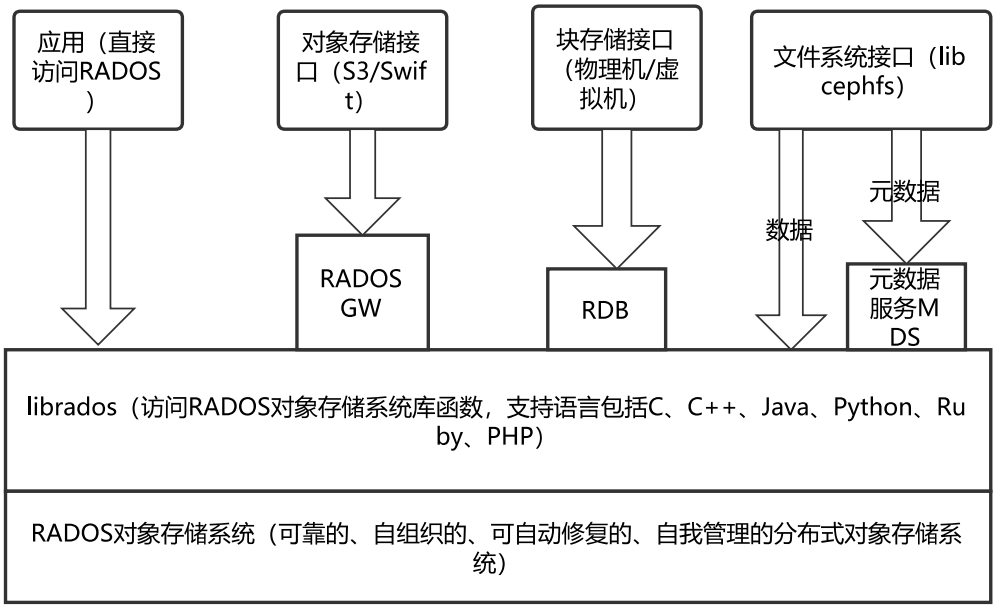

图 2-1 Ceph 架构图

从图 2-1 的 Ceph 架构图可以看出 Ceph 支持的三种接口的都是由底层的 RADOS 对象存储系统提供支持，它是 Ceph 存储集群[25]核心的地方。RADOS 是由多个 OSD 节点和少量的管理这些 OSD 节点的 monitor 节点组成的。monitor 节点通过管理集群运行图管理所有的 OSD 存储节点，每个存储节点和 Ceph 客户端上都有集群运行图的副本。OSD 存储节点的变化都会引起集群运行图变化。集群运行图是由 monitor map、OSD map、PG map、CRUSH[26] map 以及 Mds map 组成的集群拓扑。在 Ceph 存储集群中有两类守护进程：

（1）Ceph 监视器进程，该进程的主要作用是维护集群运行图的主副本，实时更新最新的集群运行图，确保 Ceph 存储集群的客户端能够获得最新的集群运行图。

（2）Ceph OSD 守护进程，该进程主要负责管理集群中所有磁盘，负责添加和删除磁盘、磁盘分区以及提供磁盘数据的可复制性。并且负责对 OSD 的状态进行检查，向监视器发送 OSD 状态报告[27]。

Ceph 通过 CRUSH 算法实现数据的分布。CRUSH 算法是一种伪随机算法，它解决了元数据服务单点故障的问题。它可以有效的将数据对象映射到存储设备上而不依赖于一个中心的目录控制。而单点故障问题存在于 HDFS 这样的分布式文件系统[28]中，由于 HDFS 中的 NameNode 服务存储了文件系统的元数据信息和操作日志，而 NameNode 又是单点提供服务的。一旦 NameNode 服务宕机就会影响整个存储系统的使用。

CRUSH 算法将数据映射到对象存储系统中的过程如图 2-2 所示。

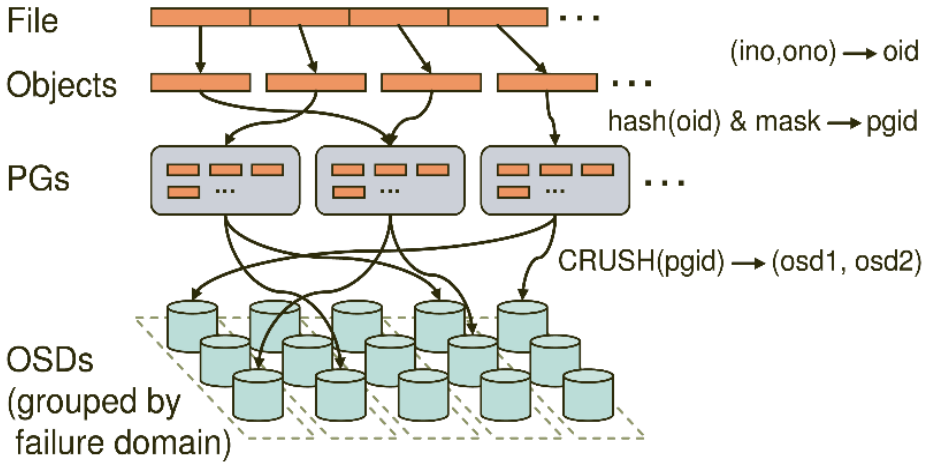

图 2-2 Ceph 数据分布示意图

由图 2-2 可以看出 Ceph 实现对象存储需要经过三个主要步骤：

（1）Ceph 客户端将文件条带化分成等分数据，得到对象 id（oid）；

（2）对 oid 做一次哈希计算获得归置组 id（pgid）；

（3）经过一致性哈希算法 CRUSH 将文件映射到具体的 OSD。

经过以上的数据分布过程，数据就能以文件的形式存储到硬盘上了。

Ceph 提供的文件系统服务 Ceph FS 与 POSIX 接口协议兼容，它基于底层的 RADOS 对象存储系统，CephFS 内的文件被映射到 Ceph 存储集群中的对象，CephFS 客户端可以把存储集群的文件系统挂载在用户空间的文件系统上。CephFS 文件系统服务需要使用到 Ceph 存储集群中的元数据服务器 MDS。元数据 MDS 服务器的作用是把文件系统中所有的元数据存在内存中，为文件的访问和定位提供服务。CephFS 在结构上把文件数据和元数据信息分离，数据的分离帮助 OSD 存储集群减轻负载压力，专注于提供文件数据服务。元数据服务器也可以像 OSD 存储集群一样按照集群的方式部署，保障元数据服务的高可用性。与常见的 HDFS 相比，CephFS 是一个通用的实时存储系统。HDFS 对于交互式应用比如网盘这样的低延迟应用支持一般，适用于计算型的大文件存储。大批量的小文件会带来大量的元数据信息，而 HDFS 中存储元数据的 NameNode 服务是单

点部署的，访问量过大的情况下 NameNode 服务会产生单点故障，整个存储服务将会宕机。而 CephFS 是可以配置多元数据的节点，不会有 HDFS 类似的问题。

Ceph 提供了重平衡[29]的策略，重平衡策略在 OSD 存储节点增加或者删除的时候生效。重平衡机制的示意如图 2-3 所示。

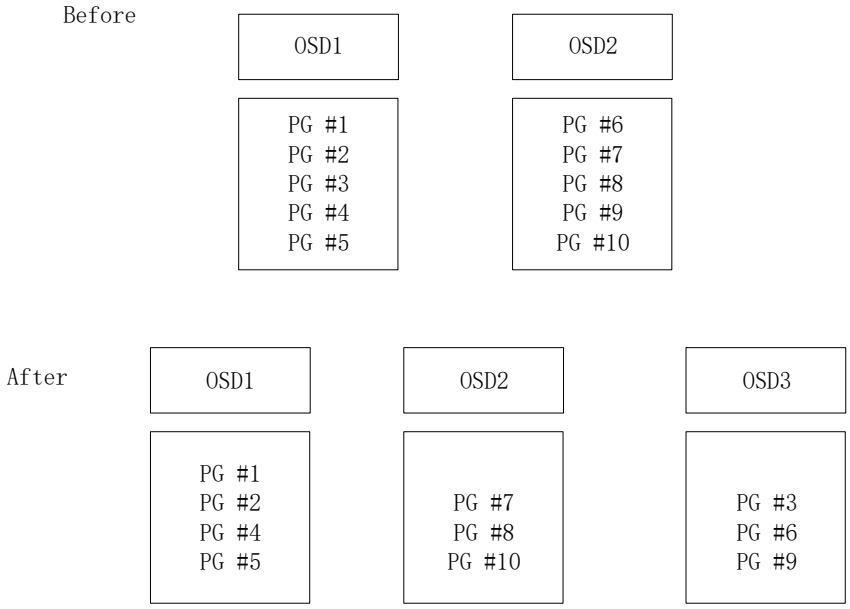

图 2-3 重均衡策略原理图

从图 2-3 中可以看到，当增加了节点 OSD3 之后，OSD1 与 OSD2 中存储的数据，会通过 CRUSH 算法将重新计算数据的存储位置（PG 值），计算完成后原来的 OSD 节点中部分数据会重新加载到新的 OSD3 节点中。重均衡策略是动态再分配存储位置的方式达到存储集群负载均衡。通过重均衡策略，新加入或者删除存储节点之后整个集群依然能够平衡数据的存储，达到集群中文件读写均衡的效果。

Ceph 提供的异地灾备方案在其 Jewel 版本中可用。由于 Ceph 是强一致性的，一个写操作的完成代表集群中所有副本上的写操作都需要完成，如果集群中的节点地域跨度很大的话网络延迟必定很大，存储集群的读写性能很大一部分都将受限于网络状况。实时性必定带来性能上的巨大损耗。在网盘系统中，文件大量读写的场景无处不在，实时备份相对于离线备份来说，对存储集群的响应时间有较大的影响。所以在本文系统中将会采用离线异地备份的方式，对文件数据做备份处理。

2.2 加密技术

加密技术[30]通常分为两种，对称加密技术和非对称加密技术。对称加密技术的加密和解密使用的是同一个密钥，它的特点是加密速度较快，比较适合大数据量的加密。常用的对称加密算法包括 DES 算法、3DES 算法、AES 算法[31]。非对称加密技术的加密和解密使用不同的密钥。它通常有两个密钥，也就是公钥和私钥。两个密钥配合使用才能对数据正确的加解密。非对称加密的基本过程是信息通信的双方首先生成一

对公私密钥，然后将各自的公钥发送给对方。在数据通信时，用对方的公钥对数据进行加密然后传输。在解密的时候，由于加密的数据是对方用自己的公钥加密的，所以只需要使用自己持有的私钥对文件进行加密。非对称加密技术的特点是比较安全，但是加密速度慢，适合小数据量的加密。常用的非对称加密算法有 RSA[32]、DSA、ECC 算法。

MD5[33]算法是一种信息摘要算法，应用于保证数据传输过程中的完整性。在文件生成 MD5 散列值之后，传输过程中即便是只修改数据的一个字节，接收计算文件数据的新散列值也是不一样的，所以它在签名、文件防止篡改等方面有较好的应用。它有易于计算、抗修改性好和压缩程度高等特点。在本文系统中采用 MD5 计算文件信息摘要的方式校验文件上传之后的完整性。

2.3 限流

限流技术(time limiting)是被用来控制网络传输过程中收发两端的传输速率的技术。这样的理念方法被借鉴到了现代互联网系统开发的过程中。通过对在本文中由于服务器的磁盘 IO 和 CPU 资源有限，所以会对关键的模块或者接口做限流处理，以防止整个服务器由于资源被消耗殆尽引发的服务崩溃和宕机。目前两种限流算法分别是令牌桶算法和漏桶算法。令牌桶算法是系统以一定的速度往一个大小固定的桶中放入令牌，当存在处理业务的需求时，需要先向桶获取一个或者多个的令牌授权，如果无法获取足够多的令牌时，则拒绝业务继续执行。漏桶算法是系统以一定的速度往一个固定大小的桶中发送执行请求，桶以一定的速度允许请求执行，而当注入到桶中的请求溢出时，请求则会被丢弃。本文采用令牌桶算法处理请求。

2.4 微服务架构

基于传统的应用架构开发的应用随着业务的更新和发展，应用的开发和修复难度以指数级的趋势增长。新增的简单功能需求或者缺陷修复都有可能引起整个应用的整体崩溃。微服务的出现简化了这些问题。微服务的架构风格[34]倡导复杂的应用应当由一个或者多个微小的服务组成，服务之间是松耦合的关系，而且这些服务各自专注于不同的业务领域，每个服务都是集中处理具体某类业务，而不像单体应用一样将所有的业务杂糅在一起。服务之间的协同工作使用轻量级的通信机制通信，用户能看到或者使用的完整功能，是由一系列的微服务组合提供的功能。同时这也给应用的快速迭代开发提供了便利，这些服务都可以独立的部署，各司其职。应用的管理和功能的交付都变得更加简单。

分布式网盘系统的几大模块都是采用微服务的架构进行设计，方便整个系统的开发、部署和运维。

2.5 gRPC

本文中通过 gRPC 来实现服务模块之间的通信。通过 gRPC 使用框架客户端可以像调用本地方法一样调用远程服务提供的方法。gRPC 是一个高性能的服务间通信框架，它基于 HTTP/2 协议标准设计，底层使用 ProtoBuf(Protocol Buffers)序列化协议开发，对大多数的编程语言环境都提供了支持。

gRPC 通信框架的特征如下：

（1）RPC 使用 ProtoBuf 序列化协议来定义服务，ProtoBuf 有较高的序列化性能。

（2）支持多种语言环境之间互相通信，包 Java、C++、go 语言等。

（3）基于 HTTP/2 标准设计，更好的网络数据传输性能。

gRPC 的通信示意图如图 2-4 所示。

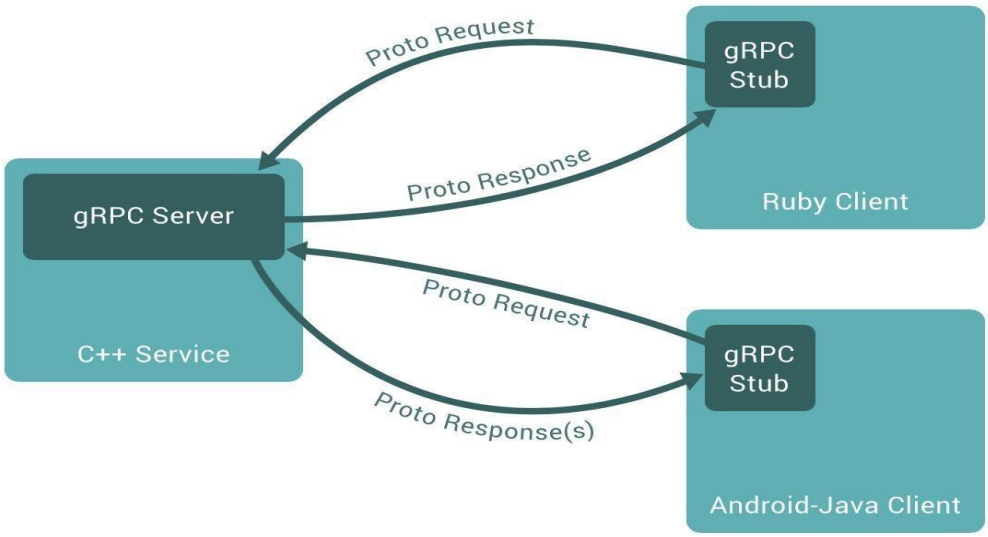

图 2-4 gRPC 通信示意图

2.6 Gin+dbr 系统框架

Gin 是一个基于 golang 编写的轻量级高性能的 web 应用服务框架，与其他的同类型的框架比如 beego 相比，在以下方面有几个明显的优点：更快的路由性能；完备的单元测试套件；更低的内存使用率；方便灵活的中间件机制（Middleware 和 context）。在开发体验上也更接近于原生 golang 的 net/http 库，理解和上手都相当的平滑。借助于 Gin 对 restful 风格的良好支持，分布式网盘的后端接口都将基于 restful 规范定义。

dbr 是一款基于 glang 编写的轻量级高性能的持久层框架，它支持结构体和数据库表之间灵活高效的映射，简单的事务语法，上下文缓存支持。dbr 这种接近于原生 sql 的语法，更易于开发和理解，结构体的映射关系避免了在业务层再次拼接数据，提升了开发效率。本文中的数据连接驱动是采用的是 Mysql 驱动，并使用 dbr 框架提供的数据库连接池，使用连接池可以提高整个系统的数据库性能。

项目每个功能模块都采用 Gin 和 dbr 组合的方式开发。Gin 框架传递并解析 http 请求，dbr 负责数据库操作。该组合具有高性能和扩展好的优良特性，同时清晰的代码架

构也为以后新增需求和功能缺陷修复提供较好的基础。

2.7 本章小结

本章首先对 Ceph 进行了介绍，分析了其相对于其他文件系统的优点，然后分别介绍了对称加密和非对称加密以及它们的使用场景，接着对限流算法进行了介绍，最后介绍了完成本文系统的重要开发技术，包括微服务架构、gRPC 技术以及 Web 开发框架。

第三章 系统需求分析

本章主要是对基于 Ceph 的分布式网盘系统进行需求分析。分别从系统的设计目标、主要实现功能以及软件模块整体的设计架构与实现进行总体分析。为系统的设计与实现奠定基础以及指明方向，并基于系统的需求分析将系统分模块设计。

3.1 系统整体目标

分布式网盘系统是从企业级用户的角度分析，设计一个安全可靠的、适用于企业组织架构且传输性能优秀的云存储工作空间，通过使用该系统来避免使用低效不安全的便携式存储设备和其他网盘服务，增强企业信息的安全性，提高企业用户的沟通协作效率。本文最终目标通过搭建一套可靠的Ceph存储系统，并且基于Ceph存储设计与实现文件存储功能，文件加密功能，异地机房的灾备以及提供用户对于文件存储的鉴权处理功能的系统网盘系统。

3.2 系统功能需求分析

在调查分析目前市场上存在的网盘系统的功能的基础上，同时对企业存储需求进行深入调研以后，对分布式网盘系统的功能进行了以下总结。

3.2.1 文件存储功能

分布式网盘是为用户提供文件数据的存储与管理的系统，在设计时考虑到用户有类似于操作本地存储文件一样操作网盘存储的文件的需求。文件模块功能包括以下几点：

（1）文件上传与下载功能。分布式网盘系统为提供的基本功能就是文件的存储。用户有在网盘中读写文件的需求，所以在系统中将会提供文件的上传和下载功能，也包括批量文件上传与文件下载操作。同时，为用户提供断点上传，秒传等功能以提高分布式网盘系统性能。

（2）文件查看功能。分布式网盘系统为用户提供了办公文档、图片文件、视音频文件等在线查看预览功能，用户可以不通过自己手动下载文件而能浏览文档文件的内容。

（3）文件分享功能。实现用户的文件在网络上的共享，可以直接分享到个人或者通过链接的方式分享，加快文件传播的效率。

以上的这些功能是设计分布式网盘系统时，为用户提供的基本的文件存储下载共享的需求，也是分布式网盘系统的基本要求，用户基本可以代替便携式实体存储设备的使用，满足用户的基本需求。

3.2.2 大文件上传功能

目前市面上的网盘在 Web 端对文件上传的大小都有一定限制，比如百度网盘限制的大小是 4G，而 owncloud 开源网盘的传输速率很低。而企业用户依然存在使用 Web 端上传大文件的需求[35]。市场上存在的云存储系统中（如百度云），对于大文件上传，分为两种情况进行解决：第一种是上传压缩包或者视频等文件时，在文件上传之前首先会计算文件的 MD5 信息摘要数值，并与存储系统中已经存在的信息进行比对，如果存在信息摘要一样的文件，则对已经存在的该文件的信息进行复制拷贝，文件在磁盘上拷贝到用户的域空间对应的目录中，文件元数据信息也拷贝一份存储在用户名下。通过此方法，可以实现已经存在的文件的快速上传。第二种是用户上传全库唯一的新文件，如新的学习视频等，在进行上传时，则只能进行读取文件流，将较大的文件分割，然后上传。对于用户主动中断上传过程的情况，在下次上传时依然能从断开处继续上传该文件。此时，可以明显的感觉到文件的上传速度变快[36]。根据此两种大文件上传的思想，需要考虑到两点：首先，需要对文件的唯一性进行验证，对于用户上传的文件，如何判断其是否已经存在于存储服务系统，其次，对于用户独占的大文件，如何提高其上传速率，是一个关键的问题。本次进行分布式网盘系统的设计时，需要通过以上两点考虑大文件的上传策略，解决大文件上传的速率问题。

3.2.3 文件安全功能

分布式网盘系统中，如何保证用户存储的文件数据安全是值得考虑的。目前市场上有的网盘是不对用户文件进行加密传输的，有的网盘中用户的数据下载之后已被损毁不能正常使用。本文系统设计时分别对文件的完整性和文件存储的加密性做以下考虑：

（1）需要考虑文件传输过程中由于网络等异常原因而出错导致的文件不完整时的情况，对于用户来说此时传输的数据皆为无用数据。所以考虑使用 MD5 算法做文件的完整性校验。MD5 算法虽然简单，但可以有效的应对文件数据的异常变动，即使是一个字节的数据量异常变动，通过 MD5 计算所得的散列值都是不一样的，它可以有效的验证文件在网络传输过程中的完整性。

（2）考虑文件存储的加密。如若 Ceph 存储集群中的数据被恶意流出，为了保证文件流出之后不被打开，需要对文件进行加密。在比较对称加密算法和非对称加密算法的性能和常见使用场景以后，本文系统采用非对称加密算法对文件数据加密。

3.2.4 群组/监控功能

群组和监控功能主要有以下几个方面。

（1）鉴定权限功能。用户对于文件有着各项不同的权限，主要有可查看，可下载，可分享，可保存等权限。

（2）团队管理功能。用户可以创建自己的私有团队。团队中有创建者，管理员，普通成员等角色。用户通过该功能可以创建自己的文件存储和分享的团队。

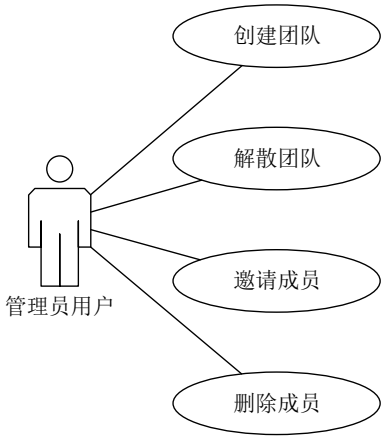

图 3-1 团队管理用例图

从图 3-1 可以看到用户可以自己创建团队组件自己的工作组，然后对该组做相应的团队管理功能的操作。

（3）监控文件操作功能。用户在网盘系统上的关键操作应该是记录，备案可查的。所以会对用户关于文件的敏感操作，比如删除、分享等记录入库。

（4）记录用户请求链路和耗时。链路和耗时监控有助于系统开发人员和运维人员了解系统性能，根据链路和耗时信息可以优化系统性能。

3.2.5 通知功能

用户有即时收到与自己相关事件的消息的需求。所以系统考虑引入邮件通知和钉钉通知功能以满足用户对于消息的及时性需求。用户在使用分享文件的功能、更新分享权限的功能、邀请团队成员以及解散团队功能的时候会触发通知功能。

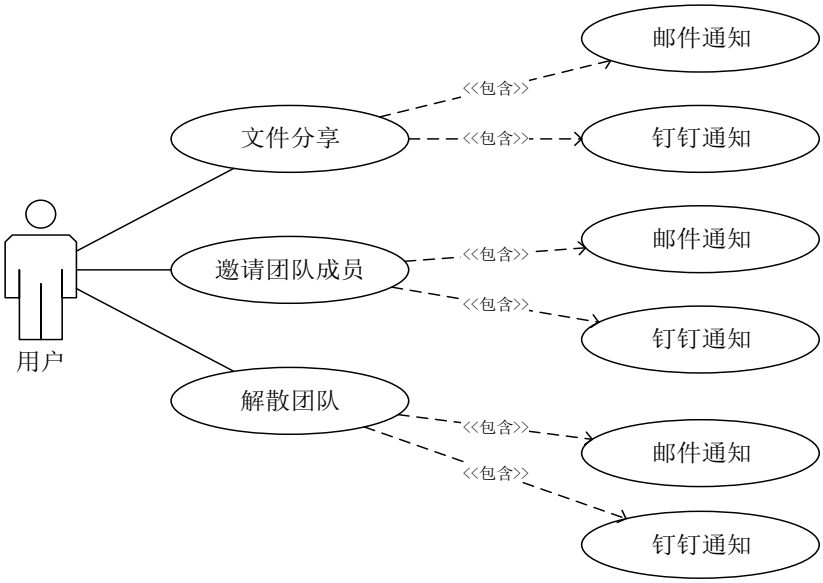

图 3-2 通知功能用例图

通知功能的用例图如图 3-2 所示。系统的通知功能作为一个组件可以嵌入到其他需要使用到通知功能的地方，扩展性也较好。

3.2.6 备份功能

在面对机房区域的自然灾害或者机房的整体硬件故障[38][39]时，需要考虑到文件全库丢失的问题。在遇到不可抗力的故障时，需要有个异地备份的功能。而 Ceph 目前的灾备方案目前还不是很完整，为了确保文件数据不丢失，故需要异地备份文件的容灾模块。考虑到全量备份在备份效率和磁盘容量上的要求，本文系统将采用定时增量备份的策略，对整个 Ceph 文件数据库做文件备份。通过定时任务向异地机房定时备份文件，可以预防文件丢失对企业的经济损失，尽量降低对企业正常运行的影响。

文件备份任务执行时候，必定会产生大量的 IO 消耗。根据企业用户的特征，在工作时间用户对网盘系统的使用依赖较高。如果备份任务在企业用户工作时间还在执行，会对用户造成影响。所以需要使用限流的方式，降低备份任务中文件读写的频次，尽量减小对正在使用系统的用户的影响。

3.3 系统性能需求分析

本系统作为一款云存储系统，在性能上有着较高的要求，必须要满足企业用户在内网环境中对于存储体验的要求。因此在设计与实现分布式网盘服务系统时，需要对以下几个方面的性能进行分析与研究。

（1） 文件的读写性能。本文设计的分布式网盘系统满足的是企业用户的需求也就是说是在企业内部的局域网进行使用，网络环境本身来说就是稳定的，因此，需要系统为用户提供一个良好的上传下载体验，也就是提高文件的上传下载速率与读写准确率。现在市场上的云存储系统中，由于网络环境不同以及收费等原因，很少提供免费的大容量快速率的网盘服务。对本文系统的读写性能[40]由用户客户端的上传下载速率指标作为依据。

（2）系统集群性能。本文所需要支持的企业用户，意味着对于网盘服务来说需要支持的 QPS 也要达到要求，单节点的 QPS 基本达到 250 即可满足。对于大量用户同时访问的情况，需要保证系统的正常的工作与运行，有较强的稳定性。所以服务将要以分布式的形式部署，业务服务和存储服务都将提供负载均衡的功能，将庞大的请求流量分布到多个节点，保证系统不会宕机。目前的稳定性指标一般都是 99.99%(一年时间内服务不可用的时间大约是 52 分钟)，本次系统设计中主要通过并发测试、压测来验证服务的稳定性。

（3）系统的响应时间。由于用户对于软件系统性能的直观感受和系统的响应时间是非常一致，所以必须保证响应时间这个指标在可接受的范围内，在用户不涉及文件 IO 操作的系统使用上，将响应时间尽量控制在 800ms 以内，这个指标是否能够达成需要在压测的过程中去验证。

3.4 本章小结

本章主要对分布式网盘系统设定了系统整体的目标，根据设定的目标和结合实际情况分析了系统功能需求和性能需求。在功能方面提出在文件存储方面的需求、团队管理和监控的需求、文件安全需求以及大文件上传的需求；在性能方面分析了性能需求以及确定性能指标。

第四章 系统概要设计

本章根据上一章的系统整体目标和需求分析来规划设计网盘系统的整体架构，对网盘系统进行模块拆分，并对模块中的具体功能进行概要设计。

4.1 系统概述

在本文的分布式网盘系统中，系统架构设计采用分层的思想，将系统分为客户端、Nginx 服务、分布式网盘服务和存储服务四个层次。系统整体拓扑层次如图 4-1 所示。

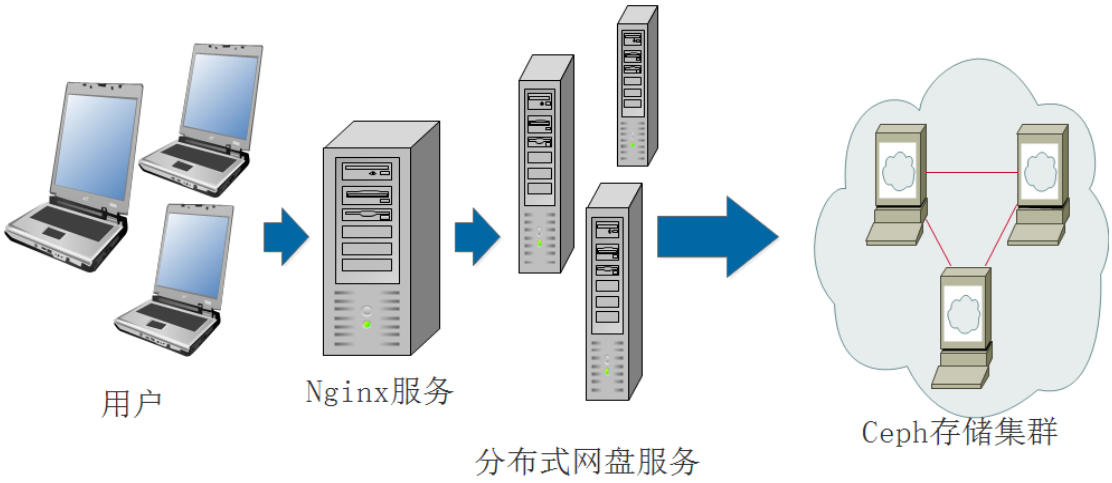

图 4-1 分布式网盘系统的拓扑图

在本文的系统中用户通过 Web 浏览器端的可视化界面访问分布式网盘系统，由于本文系统的网盘服务是以分布式的方式部署的，用户的请求会通过负载均衡服务器 Nginx 均匀分布到多个服务节点中的某一个，在该节点上处理相关的业务请求，如果业务还涉及到文件的 IO 操作，那么就会和 Ceph 集群通信，完成网盘关于文件资源的具体业务操作。

从图 4-1 可以看出，本文的系统把存储集群从网盘服务器上剥离出来。有状态的存储集群和网盘操作服务分开以达到系统架构上的微粒化。各个环节的组件化对于现代系统来说是非常必要的，而这正好是微服务架构的思想。对于本系统来说，存储服务的组件化可以提高文件数据的容错性和可用性，对于数据的备份和迁移也可起到作用。假如存储服务和业务操作服务部署在一台服务器中，任何业务服务器的问题都有可能对文件的读写产生影响，文件数据的丢失或者损毁也是有可能的。这样的架构分层在一定程度上也能起到保护存储服务的可用性以及文件安全性的作用，防止服务崩溃导致所有的文件数据损坏。

4.2 系统架构设计

本文的系统架构层次细分包括了用户访问层，网盘服务，数据存储层，以及运行环

境层，如图 4-2 详细展示各个层次的具体内容。

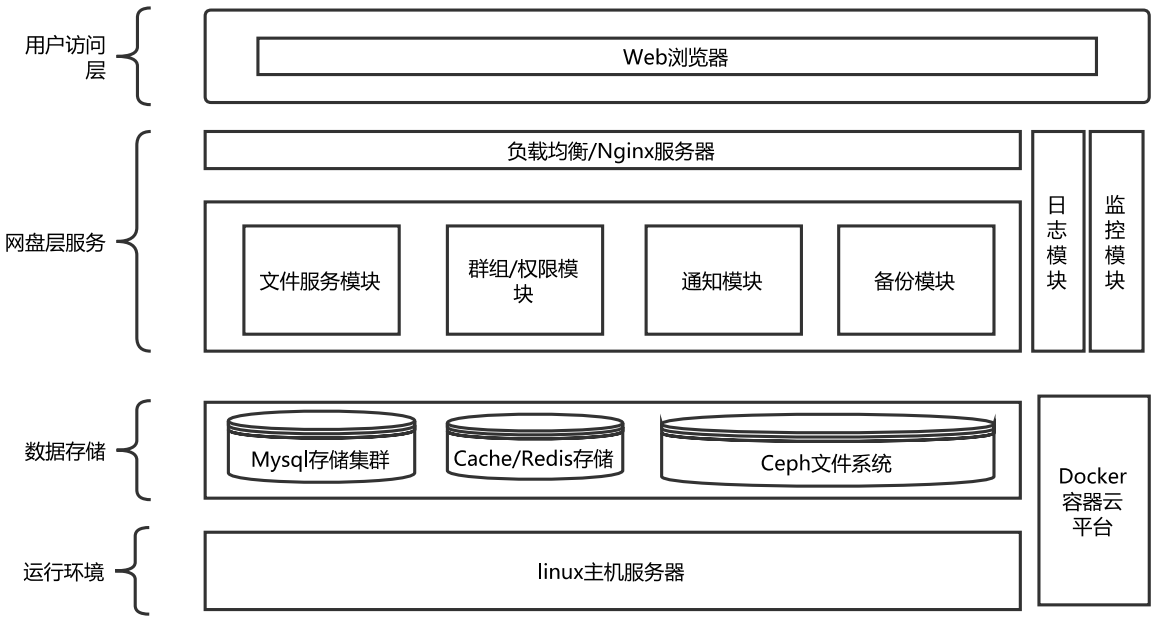

图 4-2 分布式网盘系统架构图

本文的分布式网盘系统的业务架构在上分为四个层次：分别是用户访问层、网盘服务提供层、数据存储层以及运行环境层。

（1）用户访问层。

指的是 Web 浏览器端，用户的操作都是在浏览器界面执行的。用户访问本系统的相关页面都属于该层次，用户访问时通过认证登录系统后，可以看到自己的用户信息、网盘的使用情况（容量信息）以及最近使用文件等信息。

（2）网盘服务层。

该层次是网盘服务的业务核心。

负载均衡：本系统的负载均衡使用 Nginx 服务器实现的。该模块是对多节点部署的网盘服务做负载均衡，同时还可使某个节点不可用的节点情况下，自动摘除故障节点。保障服务的可靠性和可用性。

文件服务模块：文件服务模块，主要是包含对文件的操作的后台模块。对文件资源的操作包括文件的上传、下载、移动、复制和删除等。同时也包含对大文件的处理以前安全加密的处理。

团队权限模块：本系统的对象是企业用户，所以企业用户对于群组以及权限的功能的需求非常强烈。在该模块中，主要提供基于角色的团队管理功能和对具体文件资源操作的权限鉴定。其他模块的鉴权操作都需要事先通过 RPC 接口调用鉴权模块提供的 API 获得权限。

通知模块：该模块中提供文件分享、邀请团队成员和解散团队等功能的钉钉和邮件通知。

备份模块：在该模块中提供整个 Ceph 集群的异地机房的备份功能。该功能是为应

对系统级的数据故障。

日志和监控模块：在这两个模块模块中为全系统提供日志规范，提供记录日志的功能。监控请求的耗时和链路信息，方便开发和运维追踪问题。同时由于企业用户的特殊性，防止企业关键文件数据的泄漏，会对关键的文件资源操作监控记录。

（3）数据存储层。

在该层中有 Mysql 关系数据库，Redis 缓存集群以及 Ceph 集群。Mysql 中存储了文件的元数据信息、文件目录结构、用户的配额、分享记录，团队相关的信息以及操作记录等。Redis 缓存层中存储了用户的信息等一些即时性要求不高的数据，用以提升系统的响应时间和性能。Ceph 存储集群中，存储了所有的文件信息，包括完整的文件以及缓存的分片信息。这部分的数据是整个分布式网盘系统的核心数据。

（4）运行环境。

本系统的集群都是部署在多个 Docker 容器中，在 Docker 中使用基于 Linux 操作系统的基础镜像。网盘服务部署在容器中利于系统的开发、部署和运维，有利于微服务架构的落地，在持续集成和持续交付等方面有较大的优势。

4.3 各模块的概要设计

系统功能模块如图 4-3 所示，本文的分布式网盘系统分为存储模块、文件服务模块、群组/权限模块、通知模块、备份模块等。下面将对各个模块的主要功能进行概要设计的阐述。

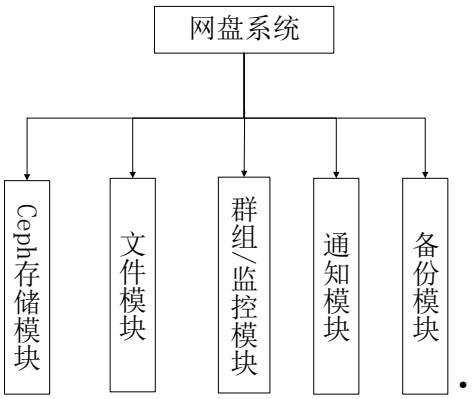

图 4-3 系统功能模块划分图

4.3.1 Ceph 存储模块

本文系统使用 Ceph 存储系统，需要的硬件环境有 6 个存储节点，分为两组，一组是网盘存储集群，另外一组是备份模块的存储集群。该模块主要搭建存储集群。以下是基本流程步骤：

（1）准备机器环境搭建存储集群，集群分为网盘服务存储集群和备份存储集群；

（2）安装 Ceph 存储集群；

（3）安装 CephFS 文件系统，并且使用 mount 工具挂载 CephFS 文件系统。

4.3.2 文件模块

文件模块的主要功能有大文件的上传下载、文件的加解密、以及文件预览以及秒传的功能。

（1）文件的上传下载功能流程如图 4-4 所示。

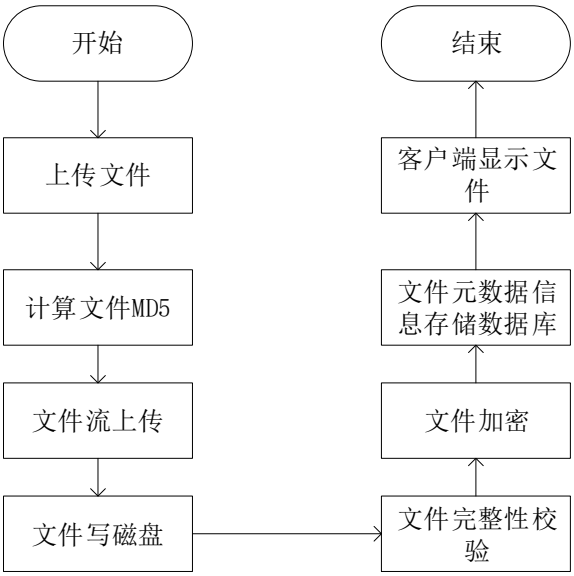

图 4-4 文件上传流程

文件模块的上传中需要注意以下两点：

1）计算 MD5 值作为参数保存，因为 MD5 是通过字符串变换算法对产生的唯一的文件信息的信息摘要，用以防止文件被恶意篡改。如果在网络中被截取篡改或者部分丢失的话，那么通过利用 MD5 的唯一性，在文件上传至服务器的时候对文件做一次 MD5 校验可以帮助我们判断文件是否被篡改过，以保证上传过程中文件的完整性和正确性。

2）整个网盘的目录结构存储在 MySql 关系数据库中，而实际的磁盘存储上，每个用户和团队在磁盘上都有一个域空间的概念，所有的文件都存在相关用户的域空间目录下面，文件在磁盘上的存储目录关系与可视化界面不一致。这样做的目的是简化文件存储结构，过于复杂的文件目录在读写文件的时候会消耗一定的资源，也可以提高磁盘访问的效率，同时降低出错率和开发难度。这样的设计还可以方便实现秒传功能，对于同一用户存储的多个相同文件在磁盘上实际只保存一份，也能提高磁盘利用率。鉴于 ext3 文件系统文件夹下的文件数是有上限的，每个文件的实际会存储在以该文件 MD5 前两位为名的文件夹中，提高了在不同文件系统下的容错能力。对于小文件而言，直接以该文件 MD5 为名保存。而对于大文件，以该文件 MD5 为名称创建文件夹，文件夹中存储该文件的所有分片。在本系统中以 1G 作为判断文件是否是小文件的数据指标。

（2）文件加密解密功能。

文件的加密解密首先都需要判定是普通小文件和大文件，不同的文件加解密流程是不一样的。

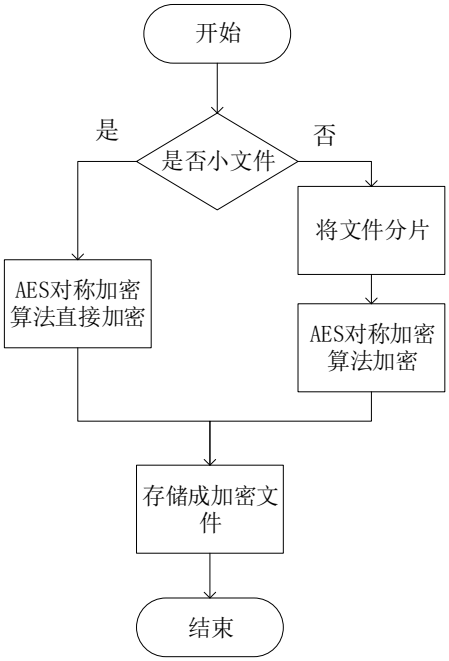

图 4-5 文件加密流程图

在图 4-5 中，文件加密是在服务端存储文件的时候加密，使用用户的文件的 MD5 信息作为加密的密钥。

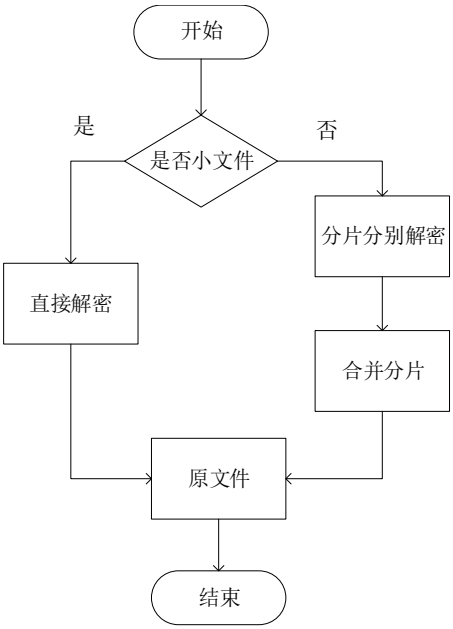

图 4-6 文件解密流程图

在图 4-6 中，文件的解密流程是预览文件或者下载文件时调用，解密的过程是在下载文件之前调用。由于加解密要在内存中读入文件，所以对于大文件，会消耗很多的系统资源。这也是大文件按照分片存储策略的原因之一。

（3）大文件上传下载功能。

1）在图 4-4 中，阐述了一般文件的上传流程。对于大文件来说，显然由于文件过

大的特殊性，可能会导致客户端的崩溃和服务器资源消耗太多的问题。所以采用分片上传的策略。

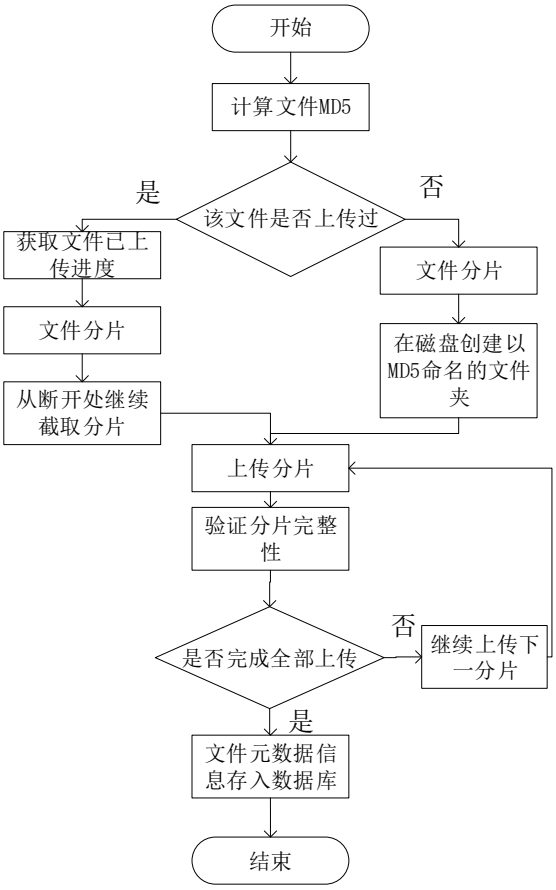

图 4-7 大文件上传流程图

图 4-7 中详细介绍了大文件上传的流程，包括分片以及文件的完整性校验。大文件在该系统中不采用单独文件存储而采用分片存储的方式，一方面是由于大文件本身已经被切割成小文件分片，直接存储更方便，另一方面也由于如果将各个小文件分片合并成一个大文件会消耗大量的系统资源，这是得不偿失的。

大文件的完整性是通过上传的分片的完整性和分片上传的个数保证的。每个分片在上传的过程中也会携带该分片的 MD5 校验码，在该分片上传完成之后使用 MD5 来校验该分片的完整性。在所有的文件分片上传完成之后，会将文件的元数据信息持久化保存到 Mysql 关系数据库中。

2）大文件的下载功能。

大文件的下载中，需要在服务端按分片顺序解密并按流式文件方式的返回给客户端。当所有分片完成之后对文件进行基于 MD5 的唯一性校验，如果传输前后两个文件的 MD5 散列值是一样的则说明文件完整。

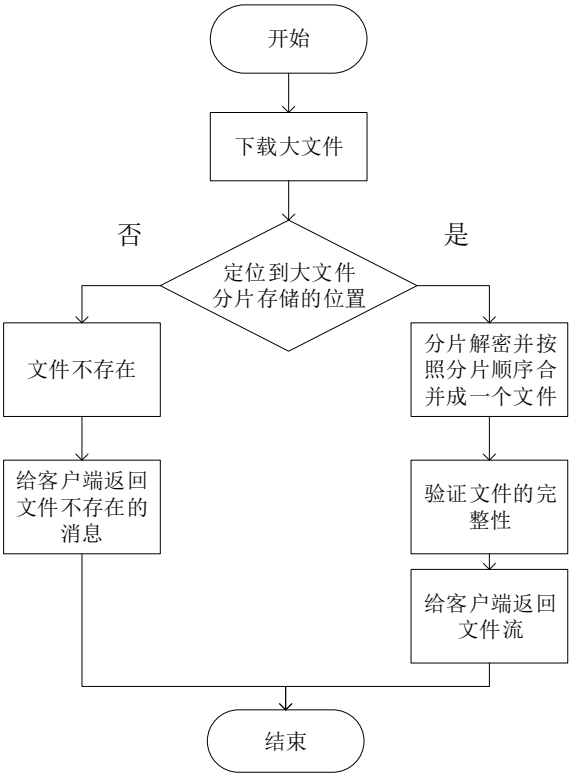

图 4-8 大文件下载流程

图 4-8 介绍了大文件的详细下载流程。当用户下载的文件是大文件的时候，由于其在磁盘上的存储方式与普通文件存储的方式不一样，所以下载的流程也不相同。下载文件的时候首先需要根据用户 NameSpace 和文件 MD5 信息定位文件的存储位置，如果不存在则直接响应客户端文件不存在的消息。如果定位到文件位置后，在待下载目录对分片文件解密并合并成一个文件，最后给客户端返回文件流，结束下载。

与普通文件的区别在于分片在服务端按顺序以流的方式返回给客户端。

（4）文件预览查看功能。

由于 Web 客户端本身不支持 Word 和 PPT 格式的文件的直接预览功能，但是浏览器可以直接查看 PDF 文件。基于这样的现状，本功能在设计的时候使用了将其他文档格式的文件转换成 PDF 格式的方案。转换的工具需要使用到 Unconv 插件。Word、PPT 的预览是使用 Unconv 插件，将源文件转化成 PDF 文件，然后 PDF 文件提供给前端预览下载。Unconv 插件需要在部署的时候使用 shell 脚本安装在部署的环境中。对于视频和图片来说，Web 前端技术本身就可以提供在线播放的功能。

文档转换的流程如图 4-9 所示。

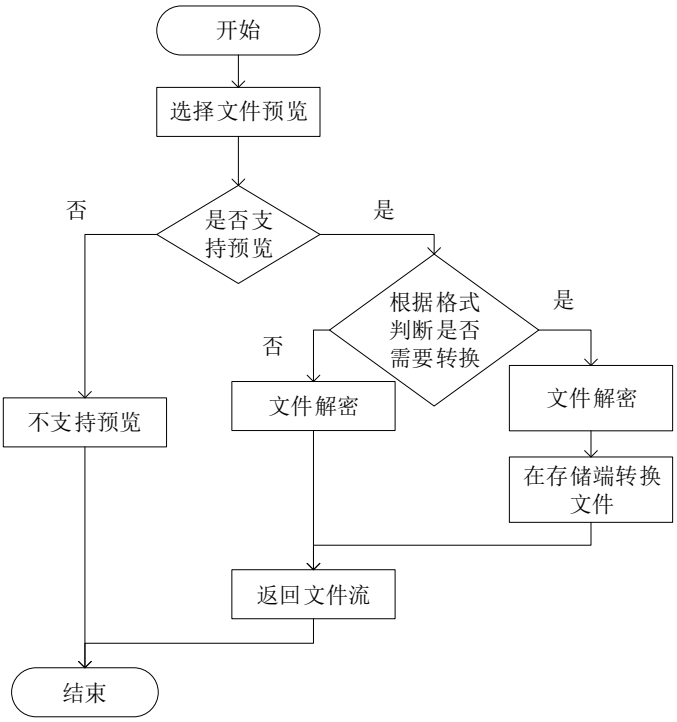

图 4-9 文档预览流程图

文件预览功能中需要根据文件的格式和大小判断是否支持文件的预览，如果文件过大则不支持预览。对于常见的图片、音频、视频等格式则可直接提供给客户端文件流，对于 Word 和 PPT 的文档格式则需要进行文件转换操作再返回文件流。

（5）文件秒传功能。

文件秒传功能是指该文件存在于该用户空间，如果文件再上传一遍，必然导致同时存在两份文件，不仅浪费磁盘空间，也降低用户体验。基于这样的考虑，在上传的时候，如果这个文件已经上传过了就会直接告诉用户上传成功，在 Mysql 数据库中直接存入文件的元数据信息。文件则是通过数据库中记录的 MD5 信息和磁盘文件做关联。

文件模块的有 resource（资源）表和 quota（用户配额）表。

资源表如表 4-1 所示。

表 4-1 资源表

字段	数据类型	含义
ID	BIGINT	表示文件的 ID
UUID	VARCHAR	源文件的 MD5 校验值
PARENT	BIGINT	上层文件夹（0 表示在顶层目录）
NAMESPACE	VARCHAR	用户空间
SPACETYPE	TINYINT	用户空间类型（0 用户，1 团队）
DISPLAY_NAME	VARCHAR	文件名
TYPE	TINYINT	文件类型（0 文件，1 文件夹）
SIZE	BIGINT	文件大小
CREATE_TIME	DATETIME	文件上传时间
MOTIFY_TIME	DATETIME	文件更新时间

CREATE_BY	VARCHAR	创建者
DELETED	TINYINT	是否删除（0 文件，1 文件夹）
DELETE_TIME	DATETIME	文件删除时间

用户配额表如表 4-2 所示。

表 4-2 用户配额表

字段	数据类型	含义
ID	INT	表示自增记录的 ID
NAMESPACE	VARCHAR	用户空间
VALUE	BIGINT	配额
DISPLAY_NAME	VARCHAR	文件名
CREATE_TIME	DATETIME	记录创建时间
MOTIFY_TIME	DATETIME	记录更新时间

4.3.3 群组/监控模块

在群组权限和监控模块中主要是为文件服务模块提供鉴权和团队管理的功能。涉及到的具体功能有分享功能，对文件的权限鉴别功能，团队相关的管理功能，对文件资源关键操作的监控记录功能。

（1）文件分享功能。

在本系统中文件分享可分为直接分享和间接分享。

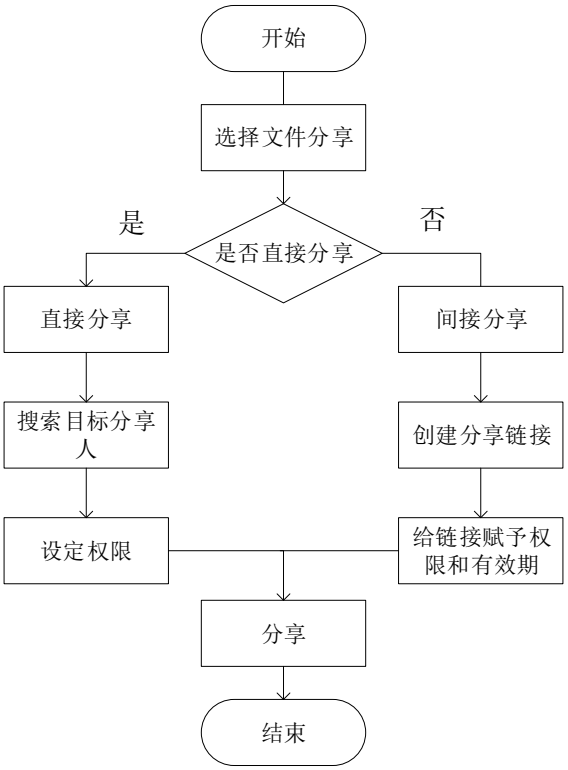

图 4-10 分享功能流程图

本文系统面向的用户是企业用户，企业有 LDAP 账号体系，用户信息都会存在该系统中。文件分享流程如图 4-10 所示，本文系统直接调用企业 LADP 系统提供的用户搜索接口的搜索结果作为直接分享的目标，每次分享会附带权限，这些权限有查看、下载、

分享等。如果是直接分享的话还会通过钉钉和邮件通知被分享者。

（2）团队的角色权限管理功能。

在团队中有基于角色的权限。用户角色可以分为创建者、管理员和普通用户。创建者有所有的人员管理权限和团队解散等团队相关的高级权限，管理员有人员管理的权限，普通用户根据管理员分配的权限使用。具体角色与权限关系对应如表 4-3 所示。

表 4-3 用户角色权限对应表

用户	权限
团队创建者	团队的解散、设置和取消管理员、以及包括管理员的所有权限
团队管理员	邀请和移除成员、给普通成员设置权限
普通团队成员	查看、下载、删除、分享权限

（3）监控文件操作的功能。

本文系统的监控是会对用户的资源操作进行监控。利用 Gin 框架的 middleware 机制，对请求做处理，如果是注册过的需要记录的操作，将会把相关操作的信息封装成一个任务对象，然后写入到数据库中，同时会把写入关系数据库的操作异步化，将相关写入任务放入协程池中，由协程池多任务并发执行，而正常的业务操作不会受到影响。

文件监控具体流程如图 4-11 所示。

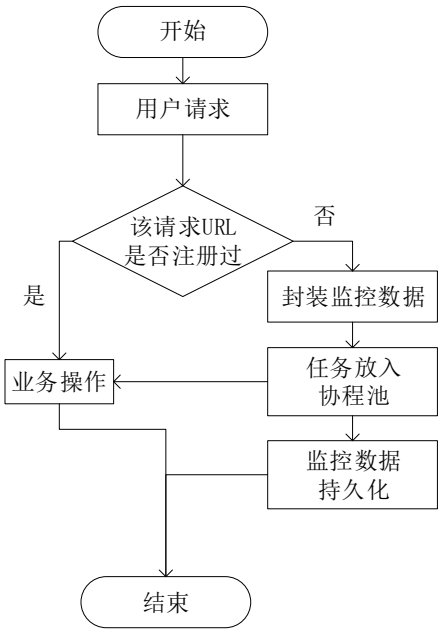

图 4-11 对文件操作数据的监控流程图

（4）请求链路和请求消耗时间的日志记录。

1）请求链路的记录。

利用 MiddleWare 的机制，当发起请求的时候，新建 Trace 对象，分别创建 TraceId，SpanId，LogId，TraceId 是由网络 IP+时间戳+进程 ID 生成的一个字符串。SpanId 是一个随机数，LogId 也是一个随机数。一次完整的 Http 请求就是一条 Trace 链路，随后会

将这些 TraceId 和业务日志写入日志文件。

2）请求消耗时间的记录。

利用 MiddleWare 和 context 的机制，在开始请求的时候记录个开始时间，结束的时候再次记录一个时间，最后相减得到了请求的消耗时间，并记录日志。

本模块的功能主要设计了分享表，团队表，团队成员表，权限表四张表。

分享表的数据库字段如表 4-3 所示。

表 4-3 分享表

字段	数据类型	含义
ID	BIGINT	记录 ID
SHARE_TYPE	TINYINT	分享类型（0 直接分享，1 间接）
LINK_ID	VARCHAR	间接分享链接
PASSWORD	VARCHAR	分享链接的密码
EXPIRATION	DATETIME	链接失效时间
RESOURCE_ID	BIGINT	文件资源 ID
CREATE_TIME	DATETIME	分享时间
MOTIFY_TIME	DATETIME	更新时间
SHARE_FROM	VARCHAR	分享者
SHARE_WITH	VARCHAR	被分享者

团队表的数据库字段如表 4-4 所示。

表 4-4 团队表

字段	数据类型	含义
ID	BIGINT	团队记录 ID
NAME	VARCHAR	团队名称
NAMESPACE	VARCHAR	团队域空间名
CREATE_BY	VARCHAR	创建者
CREATE_TIME	DATETIME	团队创建时间
MOTIFY_TIME	DATETIME	更新时间

团队成员表的数据库字段如表 4-5 所示。

表 4-5 团队成员表

字段	数据类型	含义
ID	BIGINT	成员记录 ID
TEAM_ID	INT	团队记录 ID
ROLE	TINYINT	成员角色
USER_NAME	VARCHAR	用户名
PERM	TINYINT	权限
CREATE_TIME	DATETIME	团队创建时间
MOTIFY_TIME	DATETIME	更新时间

用户权限表的数据库字段如表 4-6 所示。

表 4-6 权限表

字段	数据类型	含义
ID	BIGINT	记录 ID
TEAM_ID	INT	团队 ID
RESOURCE_ID	BIGINT	资源 ID
NAMESPACE	VARCHAR	用户空间(权限被赋予者)
PERM	TINYINT	权限
GIVER	VARCHAR	权限赋予者
GIVER_TYPE	INT	权限类型

4.3.4 通知模块

通知模块的功能主要是在用户定向分享，邀请团队成员，更新权限，团队解散等功能的时候发出钉钉和邮件通知。

钉钉通知和邮件通知功能主要设计是调用第三方 RESTAPI 接口，邮件内容是事先定义好的一些 Html 模板。通知模块的第三方接口信息如下表 4-7 所示。

表 4-7 通知接口的 API 协议

接口名	接口 URL	请求方式	返回值
钉钉通知接口以及参数	http://api-service.notify.com/feige_service/feige/sendDingNotify 参数：ACCESS_Token（访问 token） SENDER（发送者） TOUSERS（发送对象） CONTENT（通知文本内容）	POST	Messag 消息体 Code：0 失败 Code：1 成功
邮件通知接口以及参数	http://api-service.notify.com/feige_service/feige/sendMail 参数：ACCESS_Token（访问 token） SENDER（发送者） TOUSERS（发送对象） CONTENT（邮件正文内容） Title（邮编标题）	POST	Message 消息体 Code：0 失败 Code：1 成功

4.3.5 备份模块

由于 Ceph 的分布式文件存储目前没有完整的灾备系统，鉴于文件安全和预防灾难故障等方面的考虑，本系统采用定时向异地机房的备份策略保证文件的在地区机房全故障的情况下保证数据的安全。备份可分为增量备份和全量备份。全量备份的情况下，由于可能出现备份的同时数据依然在写入，网盘系统服务需要停机维护，影响用户的使用，影响用户体验。出于对成本方面和备份任务对于现有系统的影响和用户体验方面的考虑，本系统将两者结合使用，采用首次停服全量备份，之后增量备份的策略。该备份模块的示意如图 4-12 所示。

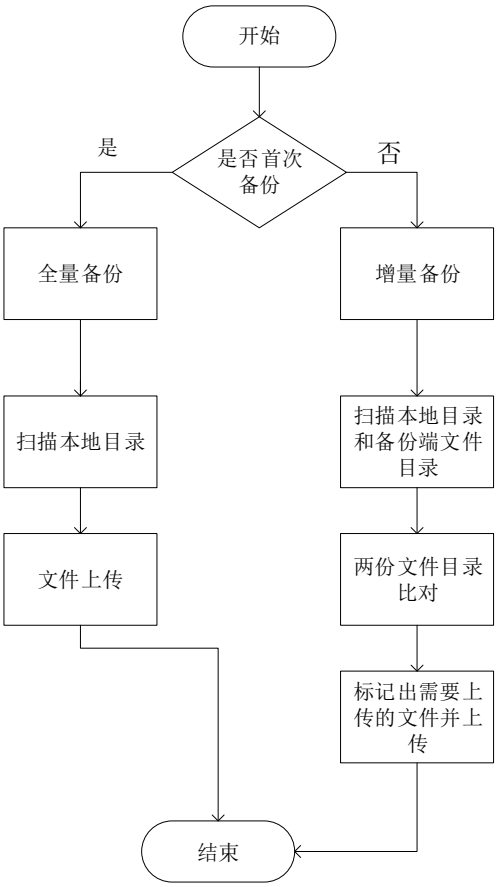

图 4-12 备份模块流程图

针对本文系统的目标用户是企业员工的情况，备份任务选择用户对网盘需求不大的非工作时间开启，使用 cron 工具开启定时任务功能。为了防止备份任务执行时间过长，导致当到达工作时间用户开始大量使用网盘时，服务系统 IO 负载过高的问题，对备份接口进行限流。限流的算法使用令牌桶算法。令牌桶算法的流程示意如图 4-13 所示。

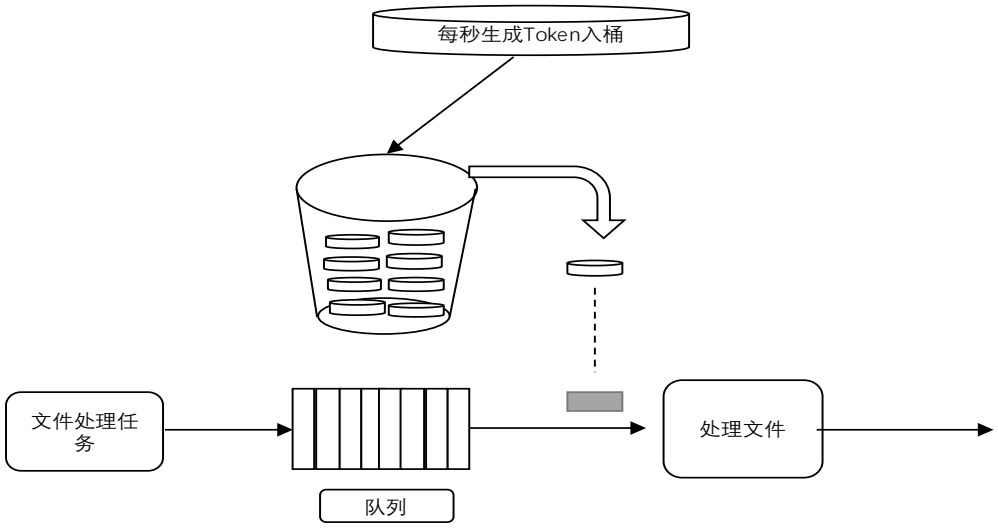

图 4-13 限流示意图

当需要向备份集群备份文件时候，先要向令牌桶中取令牌 token，成功拿到令牌才能继续执行任务。如果不能拿到则该文件的处理任务则阻塞住。令牌的产生速度可以是

一秒一个或者多个，可自由控制。文件备份结束标志存储在的索引文件中。

4.4 本章小结

本章主要根据需求分析，根据对模块的划分，介绍了本文系统的基本架构，以及各个模块的具体功能，并对重要的功能做出概要设计，给出了初步的设计方案，引导未来具体详细方案的实施。

第五章 系统详细设计与实现

本章主要是在第三章的需求分析和第四章的概要设计的基础上，对基于 Ceph 的分布式网盘系统进行详细设计，并且实现网盘的各个功能模块。

5.1 Ceph 存储模块

该模块提供网盘的存储和备份模块的存储，所以本模块专注于 Ceph 文件系统的搭建。首先需要搭建 Ceph 存储集群，主要分为以下步骤：

（1）安装 Ceph 集群部署工具

1）执行命令安装包管理工具：

sudo yum install -y yum-utils && sudo yum-config-manager --add-repo https://dl.fedoraproject.org/pub/epel/7/x86_64/ && sudo yum install --nogpgcheck -y epel-release && sudo rpm --import /etc/pki/rpm-gpg/RPM-GPG-KEY-EPEL-7 && sudo rm /etc/yum.repos.d/dl.fedoraproject.org*

2）安装 Ceph 部署工具：

sudo apt-get update && sudo apt-get install ceph-deploy

（2）Ceph 节点安装

1）安装 NTP 工具（确保分布式集群的时钟不会漂移）：

sudo yum install ntp ntpdate ntp-doc

2）安装 SSH 服务：sudo yum install openssh-server

3）创建 Ceph 用户并赋予权限：

ssh user@ceph-server

sudo useradd -d /home/cooper -m cooper

sudo passwd AQDjMmlbwz6gMBAA9+DkZJ1pLGEL4jFb1ZPzYA==

echo "cooper ALL = (root) NOPASSWD:ALL" | sudo tee /etc/sudoers.d/cooper

sudo chmod 0440 /etc/sudoers.d/cooper

（3）创建集群

1）创建 monitor 节点：ceph-deploy new cooper-monitor-node

2）安装 Ceph：ceph-deploy install admin-node node1 node2 node3

3）登陆节点 2 和 3 分别给 OSD 守护进程创建目录：/var/local/osd0 和/var/local/osd

4) 节点 admin 准备和激活：

ceph-deploy osd activate node2:/var/local/osd0 node3:/var/local/osd1

5）创建 MDS 元数据服务器

（4）创建 CephFS 文件系统

1）创建存储池命令：

ceph osd pool create cephfs_data;

ceph osd pool create cephfs_metadata（一个 Ceph 文件系统需要至少两个 RADOS 存储池，一个用于数据、一个用于元数据）

2）创建文件系统

ceph fs new cephfs cephfs_metadata cephfs_data

基于以上的详细步骤一套基于 Ceph 的分布式文件系统就实现了，网盘服务只需要使用 mount 的方式挂载到 Ceph 文件服务器上即可，随后就可以像操作本文文件一样使用 Ceph 存储系统。

5.2 文件模块

文件模块的主要功能有文件上传下载、大文件的上传下载、文件加解密、预览、秒传的功能。

5.2.1 普通文件上传功能

文件上传是分布式网盘系统提供的基础功能，本文的文件上传分为普通文件和大文件上传，分割标志是是否超过 1G。用户进行普通文件上传的操作流程：用户点击上传按钮选择需要上传的文件，并确认。随后页面会显示正在上传且有传输进度条。等到完成上传后会在网盘文件中显示上传结果。

文件上传功能的时序如图 5-1 所示。

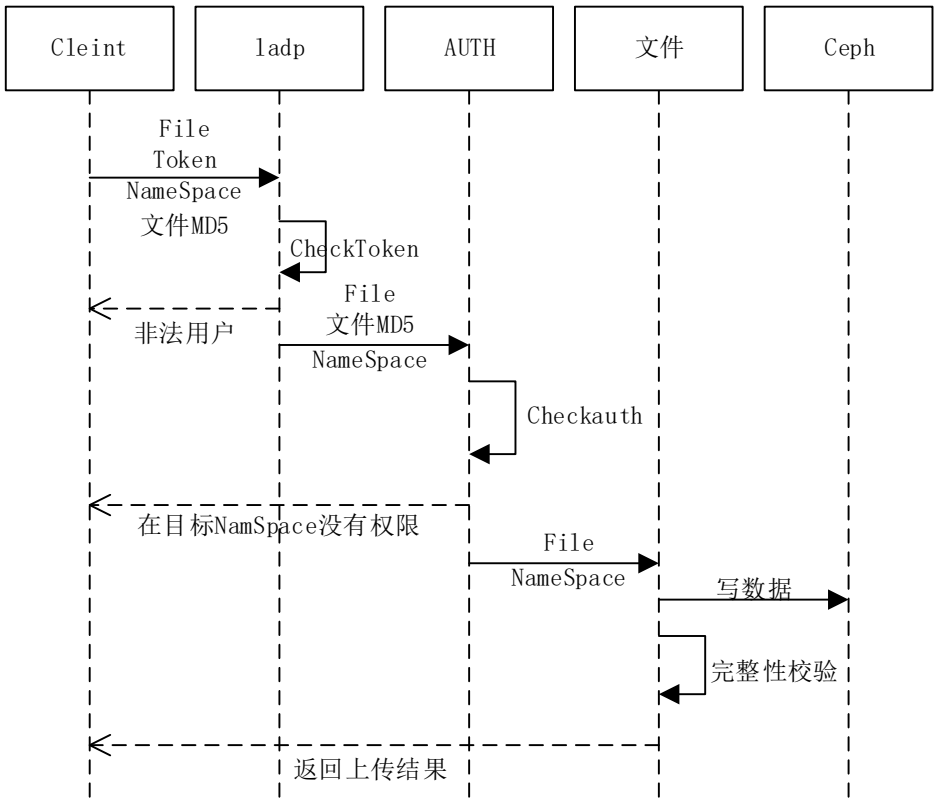

图 5-1 文件上传时序图

在上传的过程中需要使用 Token 作为参数调用 LADP 接口来验证用户身份的合法性，然通过 gRPC 远程调用 Auth 模块验证用户对目标空间（个人空间和团队空间）是否存在上传权限，验证通过之后才上传文件。核心伪代码如 5-1 所示：

伪代码 5-1：fileUpload 输入：file 输出：message
1. nameSpace←userName,md5←file.MD5 2. if file.isExist() then 3. return exist 4. else destDir←buildFileDir(nameSpace,md5) 5. destFile←saveResource(destDir,file) 6. destFileMd5← destFile.MD5 7. if destFileMd5≠md5 then 8. return error 9. end if 10. encrypt(dstFile) 11. saveFileMetaDataToDB() 12. return success 13. end if

上传文件的时候，首先会调用 LADP 提供的接口验证用户的登陆信息，如果验证结果是 token 过期或者非法 token 的情况，接口则会返回非法用户。然后会调用权限模块的接口查看用户是否在目标空间有上传权限。权限验证通过之后的步骤写文件。写文件需要在磁盘上根据目录生成规则定位存储的目录，其中以 MD5 前两位多生成一个上层目录的原因是某些文件系统存储单个文件夹存储的文件数量是有限的，多加一层目录可以增加在 ext3 文件系统情况下文件存储的数量。并以文件的 MD5 校验文件的完整性。完整性校验通过之后对文件加密，最后当磁盘的读写结束之后，再将文件的文件元数据信息保存到数据库中。

5.2.2 大文件上传功能

前面小节已经介绍了普通文件上传的方式。面对超过 1G 的大文件，本文使用分割的方法，将大文件分割成每 100M 为一分片，按分片作上传处理。完整的一个大文件，如果直接上传的可能会引发异常的情况。由于直接完整上传需要将文件全部读入内存，这样对浏览器的性能和机器内存有要求，如果性能不够或者内存不足，浏览器很可能出现崩溃或者卡死的情况。按分片上传的策略一方面可以降低前端浏览器崩溃的可能，另一方面也可以提高系统性能。后端系统主要提供两个接口：MultiPartUpload 接口和 CompleteMultiPart 接口。MultiUpload 接口是在分片上传的时候使用，该接口接收前端

的参数有 data，表示文件分片；chunkMd5Value 表示分片的 MD5 信息；fileMd5Value 表示文件的 MD5 信息；index 表示该分片处于第几块的位置。CompleteMultiPart 是客户端完成所有分片上传之后做的整个文件的完整性校验。大文件上传的时序图如图 5-2 所示。

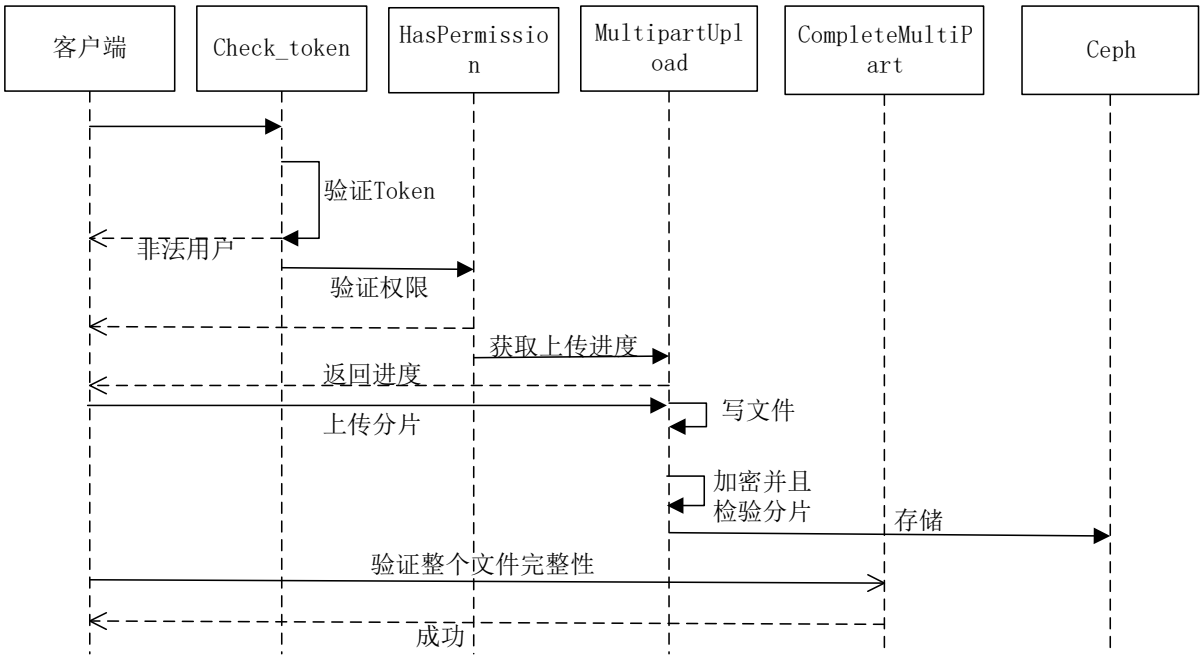

图 5-2 大文件上传时序图

在图 5-2 中，首先需要使用 LDAP 的 checkToken 接口对用户的身份进行认证，然后使用 gRPC 远程调用 Auth 模块的权限接口验证权限。然后通过判断数据库中是否有该文件元数据记录和磁盘中的文件记录判断是否存在部分分片，如果有的话则返回已存在分片的个数返回给客户端，客户端则从断开处上传分片。如果不存在已有分片则上传完整文件。首先需要生成文件的存储目录，在该目录下写分片文件，校验分片的完整性并加密。当客户端所有的分片上传成功之后，客户端调用 CompleteMultiPart 接口验证整个大文件是否上传成功，是否成功的依据在于分片数量是否和按文件大小切割的数量一致。核心伪代码如 5-2 所示：

伪代码 5-2：MutipartUpload 输入：shadingfiles[],file 输出：message
1. nameSpace←userName,md5←file.MD5,nums← shadingfiles.length,index←0 2. if file.isExist() then 3. return exist 4. else destDir←buildChunkFileDir(nameSpace,md5) 5. for index to nums do 6. destFile←saveResource(destDir,shadingfiles[index])

```
7.          destFileMd5← destFile.MD5
8.          if destFileMd5≠shadingfiles[index].md5 then
9.            return error
10.         end if
11.         encrypt(dstFile)
12.         end for
13.         checkChunks(namespace,md5)
14.         saveFileMetaDataToDB()
15.         return success
16.   end if
```

大文件上传的过程中首先会判断该文件是否上传过，如果上传过则返回。如果没上传过，则通过 buildChunkFileDir 创建文件存储目录，然后循环写分片并对分片进行分片的完整性校验和加密，接着验证文件上传的分片个数是否符合，最后再将文件的元数据信息存储到数据库中。存储的大文件以分片的方式存储避免再次合并成为一个大文件，这样降低了 CPU 资源的消耗，一定程度上也有利于提高系统的性能。

5.2.3 文件安全功能

文件加密解密是作为一个组件分别在写文件和读文件的过程中使用，本文采用的加密算是 AES 加密算法。在普通文件的写过程中，加密是对整个文件加密。而对于大文件来说，是对各个分片加密。加密的密钥是文件加密之前的 MD5。由于 MD5 算法的不可逆性，泄漏之后的文件依然不能获取原文件的 MD5 信息。文件数据和数据库中存储的 MD5 数据是分开存储的，这样也可保障文件的安全性。本文系统中采用 cfb 模式对文件加密，也就是密文反馈模式。

伪代码 5-3：Encrypt

输入：dst,fileSrc

输出：fileDst

```
1.  key←file.MD5,r←bufio.NewReader(fileSrc)
2.  block←aes.NewCipher(key)
3.  stream←cipher.NewEncrypter(block)
4.  reader←cipher.StreamReader(stream,r)
5.  io.copy(dst,reader)
```

如伪代码 5-3 所示是文件加密过程中需要提供的方法 API，使用文件的 MD5 作为加密密钥，首先读入源文件流并且生成加密 block，利用 cipher 提供的 NewCfbEncrypter 方法生成加密 stream，然后对文件加密，最后使用库函数 io.copy 方法生成文件。

伪代码 5-4：Decrypt 输入：dst,fileSrc, fileMD5 输出：fileDst
1. key←fileMD5,r←bufio.NewReader(fileSrc) 2. block←aes.NewCipher(key) 3. stream←cipher.NewCfbDecrypter(block) 4. writer←cipher.StreamWriter(stream,dst) 5. io.copy(writer,r)

伪代码 5-4 所示是文件解密过程中需要提供的方法 API，使用文件的 MD5 作为解密密钥，首先读入源文件并生成解密 block，利用 cipher 提供的 NewCfbDecrypter 方法生成解密 stream，在 cfb 模式下加密则依然使用 cfb 模式解密，最后使用库函数 io.copy 方法生成最后的解密文件。

大文件加密过程的状态图如 5-3 所示。分别从前端分片到文件具体写到磁盘中的具体文件状态转换。

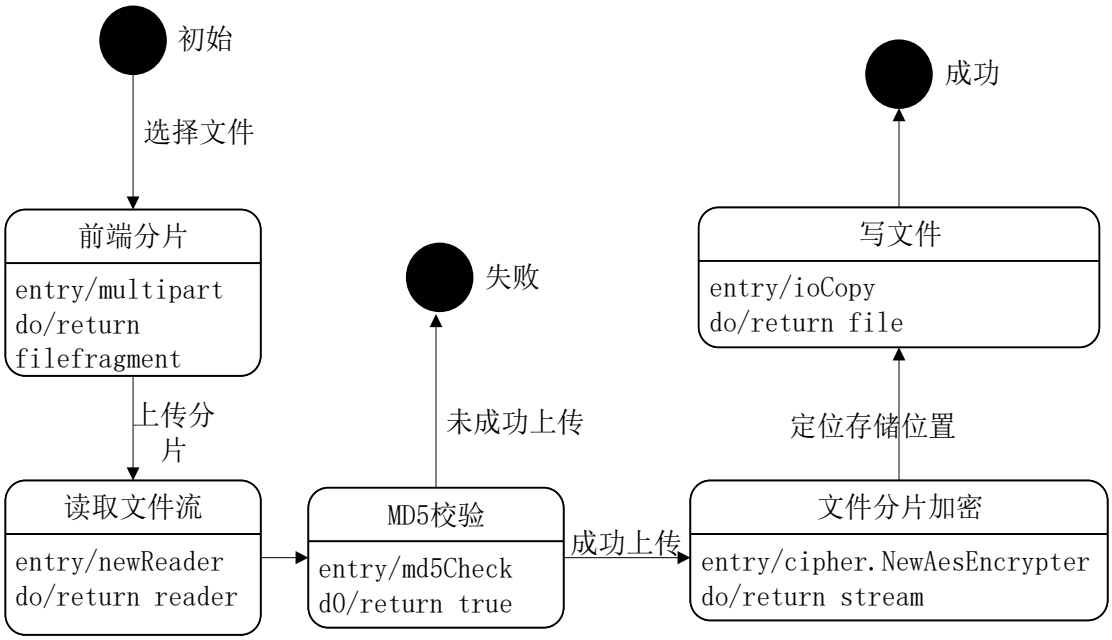

图 5-3 文件加密状态图

在图 5-3 中可以看到，文件的加密在校验完分片或者小文件的完整性之后进行，验证文件文完整性是通过 md5Check 函数获得结果，通过比对上传之前的 MD5 和上传之后的 MD5 的散列值，如果两个值一致则说明上传的文件完整，继续执行加密操作，如果两个值不一致则表示文件上传错误，与原文件不一致，需要重新上传。当发生错误之后将文件上传失败的消息返回给前端浏览器。

5.2.4 文件下载功能

由于文件大小的不同在磁盘上存储的方式也不相同。对于普通小文件而言，根据存储策略，文件在磁盘上的存储形式还是单个的完整文件，存储文件的文件是以其 MD5 为名，所以需要通过/{NameSpace}/{MD5 前两位}/{MD5}/来定位文件。如果{MD5}层级的属性是个文件，则表明所存储的是个小文件，则对该文件解密之后即可直接按照流的方式返回给前端下载。而当确定{MD5}层级是个文件夹的时候，即可确定所存储的是个大文件。文件在磁盘上是分片存储的。所以需要对大文件的每个分片进行解密并按照分片的顺序合并成一个大文件，然后以文件流的形式返回前端浏览器。文件下载的功能也支持批量下载，在后端接口接收到下载多个资源的请求时，首先根据资源的 MD5 定位读出各个文件的位置，如果其中包含大文件，还需将大文件的各个分片按照分片的顺序 Merge 合并成为一个完整的文件，最后把所有的所需文件下载打包压缩成一个压缩包文件。而这个压缩包文件存在/temp 目录的下面，当完成文件的下载之后，会清理该临时文件，以免过多的临时文件占用有限的存储空间。

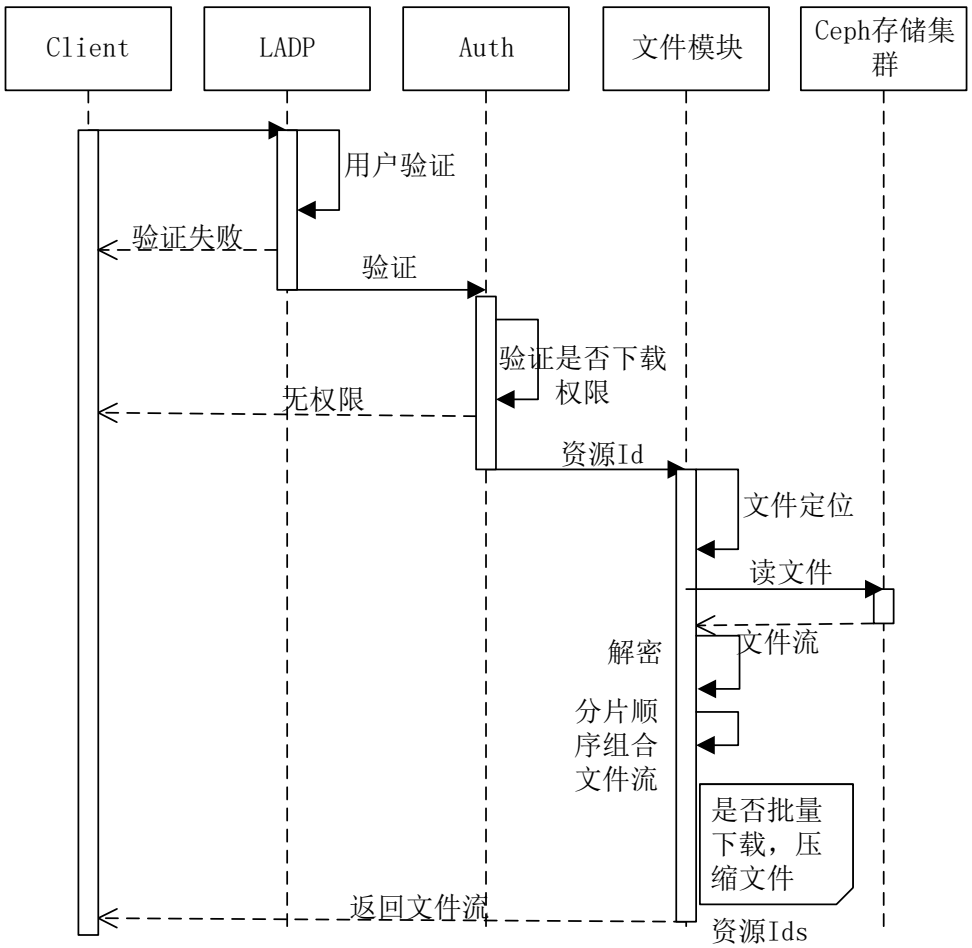

图 5-4 文件下载时序图

本文的下载是在 Download 接口中实现的，首先是校验用户身份的合法性，然后对用户关于资源的下载权限进行校验，最后在文件模块定位文件的位置，使用 Decrypt 方

法对文件解密，通过 getDownloadpath 方法返回文件流地址。用户下载的时候客户端获取以文件流的形式返回的文件。如果是批量下载文件，还需要在存储端通过 ZipFile 方法将所有需要下载的压缩打包，再以流的形式传输回客户端。

5.2.5 文档预览功能

预览功能是针对文档类型和图片类型提供的功能，由于前端不支持 Word、PPT 格式查看，但是支持 PDF 格式，所以本系统的是将文档文件转化为 PDF 文件，然后提供给前端下载。

Unoconv 插件的安装描述写在 DockerFile 文件中，在构建镜像的时候就会执行以下的命令安装并赋予可执行权限：

```
curl -Ls https://raw.githubusercontent.com/dagwieers/unoconv/master/unoconv -o /usr/local/bin/unoconv && chmod +x /usr/local/bin/unoconv。
```

文件预览的时序图如 5-5 所示。

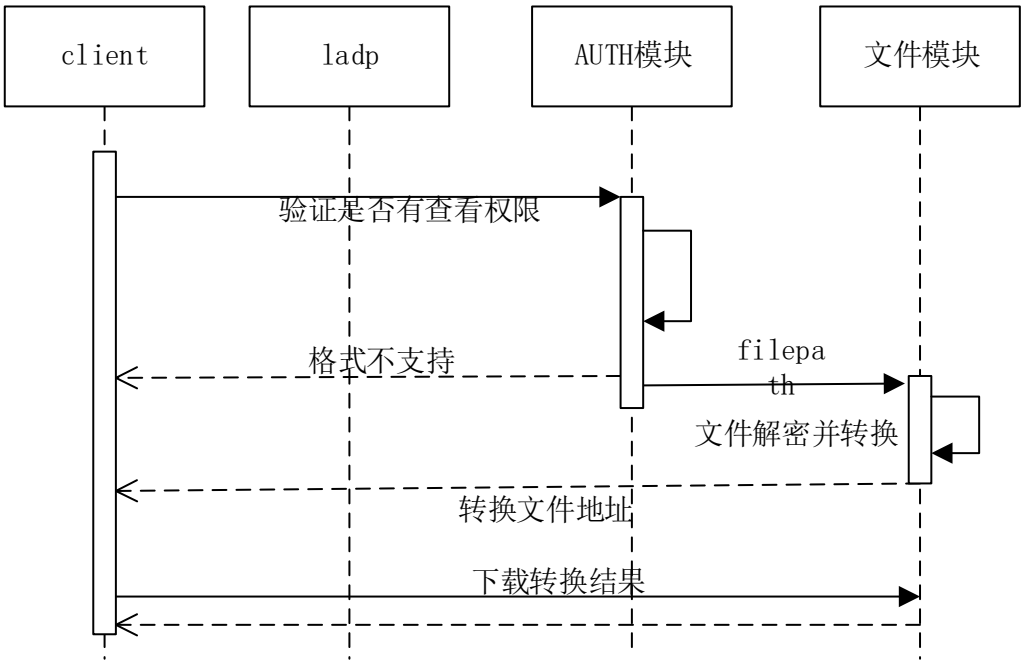

图 5-5 文件预览时序图

在用户发起对文件的预览请求的时候，首先会验证对文档是否有权限。然后通过 CheckMimeType 查看文件的格式是否支持，当文件不支持的时候直接返回格式不支持的消息给前端。如果格式支持，需要使用 Decrypt 方法对文件解密，并且用 ConvertToPdf 方法将该文件转化 Pdf 格式，转换格式的命令是：unconv -f pdf -o dst src，在转换完成之后给客户端返回文件流的地址。客户端则再通过下载地址获取 Pdf 文件显示给用户。

5.2.6 文件秒传功能

一个用户在网盘空间上传同一份文件，如果实际再上传一遍就是对网络带宽资源和服务器资源是一种浪费。秒传功能就是在文件真正上传之前检查文件是否已经存在，如果在该用户空间有同样的一份文件，直接保存文件信息到数据库中，不需要实际传输文件。如果在其他用户空间则拷贝文件之后，再存信息。文件秒传功能时序图如图 5-6 所示。

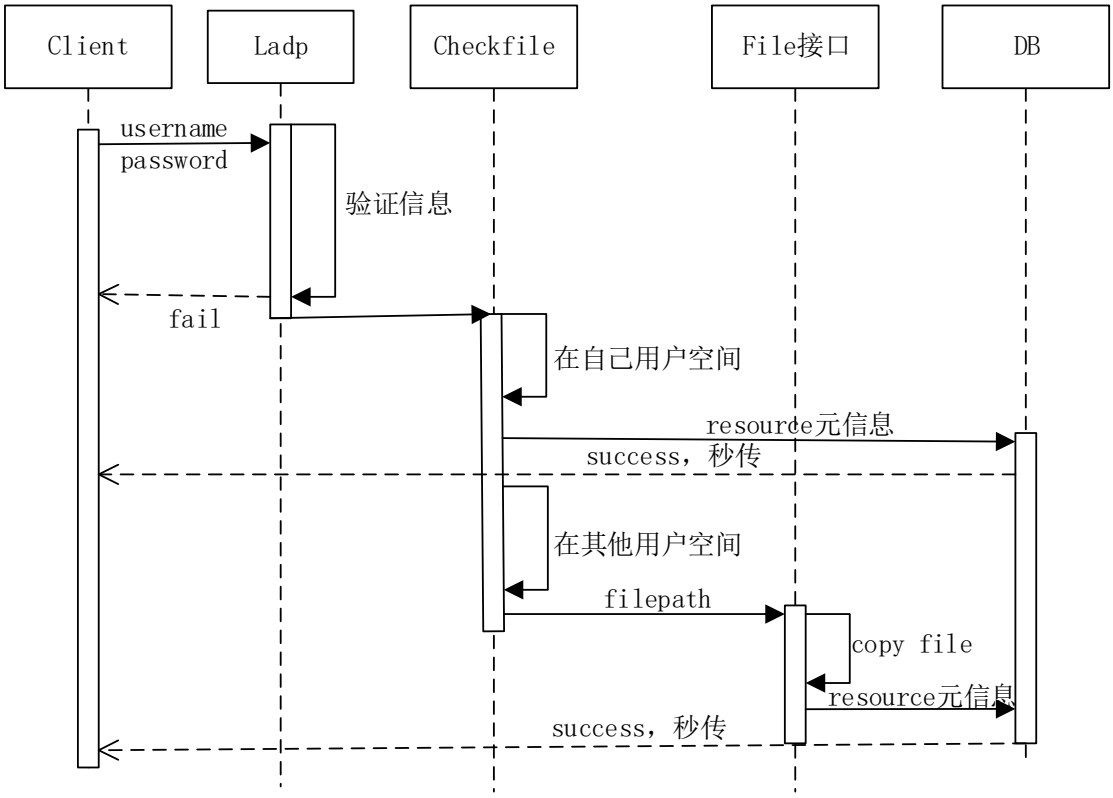

图 5-6 文件秒传时序图

代码逻辑的核心在于通过 CheckResource 方法以 MD5 为搜索条件在数据库中搜索记录，如果没有则不存在相同的文件，如果能够匹配命中则说明该文件在网盘中已存在，然后再根据文件是否在用户自己的存储空间来决定是否需要拷贝文件。如果是在用户自己的空间，则直接将文件元数据信息保存至数据库，返回上传成功。如果是在其它空间中，则通过 CopyFile 方法将文件以拷贝的形式保存至用户的空间，并保存文件元数据至数据库中。磁盘的拷贝必定比网络传输的速度要快很多，所以依然能够达到秒传的效果。

5.3 群组/监控模块

群组/监控模块主要是提供鉴权功能、链路以及操作监控等功能的设计与实现。

5.3.1 鉴权功能

该模块主要是根据用户在数据库的权限提供鉴权的功能。包括用户在团队中的各类权限以及对分享文件的各类权限。基于数据库的 Team 表、Member 表以及 Authoration 表，分别实现个人在团队中的权限以及个人在团队中具体资源的权限。同时个人在团队中对具体资源的权限可以覆盖个人在团队中的角色权限。用户角色权限的关系如图 5-7 所示。

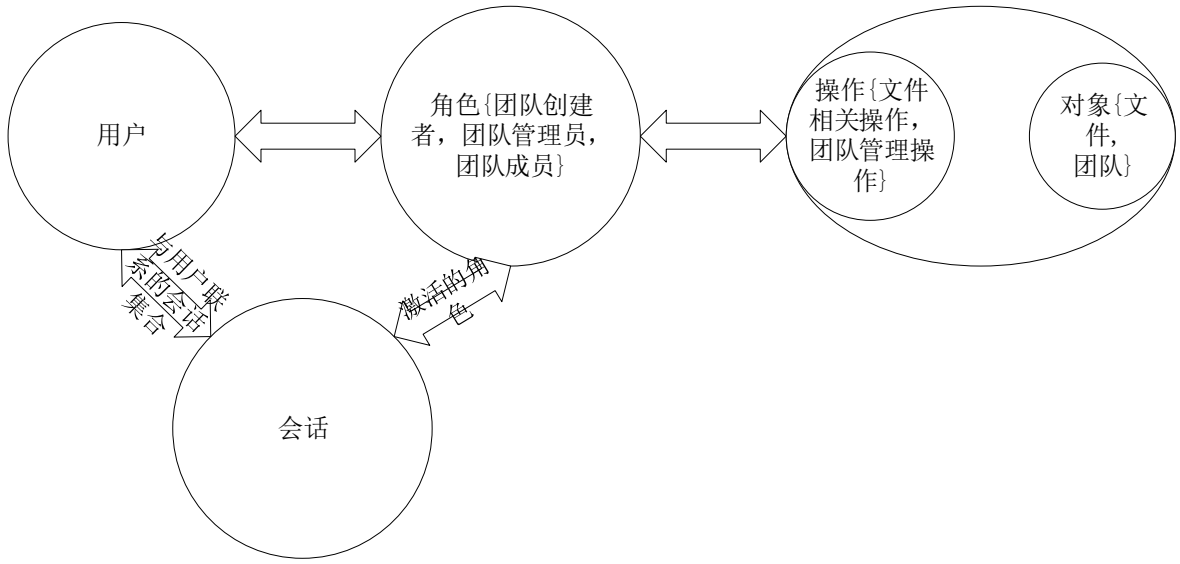

图 5-7 用户角色关系图

用户对于个人所有的文件拥有所有的权限，在该模块中主要描述的是个人在团队中的角色权限和对单个文件的访问控制权限。该模块的接口是基于 gRPC 协议的接口。首先需要使用 go get 工具与安装 protoc 编译器和 gRPC。然后定义 auth.proto 文件。在该文件中定义好客户端请求的数据格式和服务端响应的数据格式。消息如下：

```
1.  Message  Authrequest{
2.  String username = 1
3.  String TeamNameSpace = 2
4.  Int32 resourceId = 3
5.  String authType = 4
6.  }
7.  Message Authresponse{
8.  String username = 1
9.  String TeamNameSpace = 2
10. Int32 resourceId = 3
11. String authType = 4
12.  Int perm = 4
13. }
```

消息的整个转换过程的过程如图 5-8 所示。

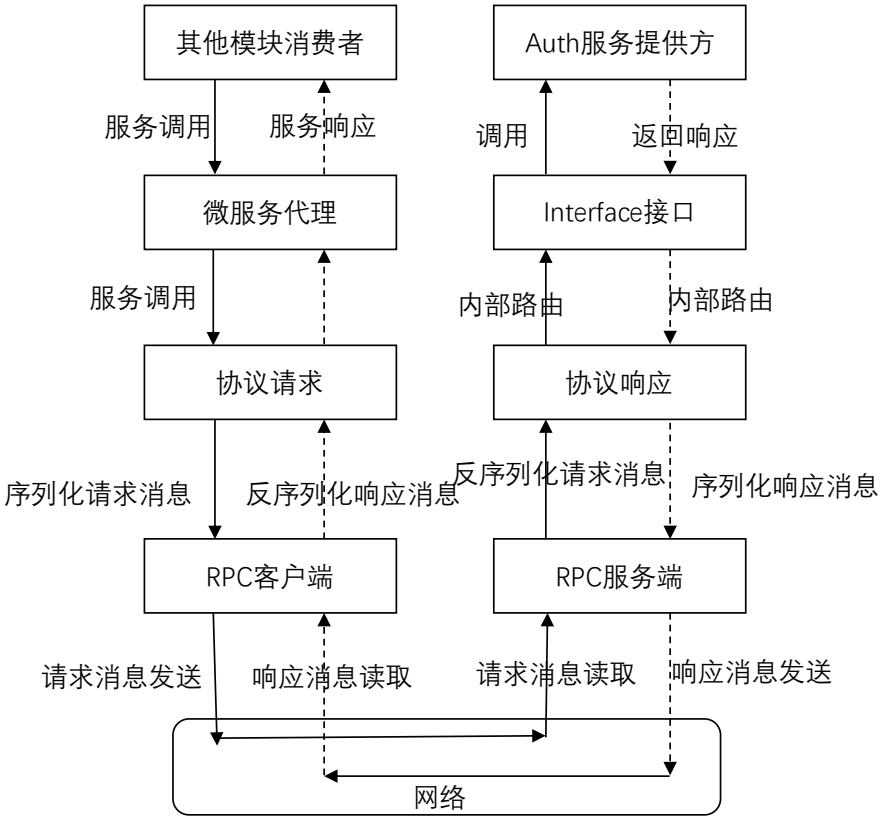

图 5-8 消息转换过程图

如图 5-8 所示，当其他模块需要通过调用权限模块的时候，首先发起请求并进行协议转换，然后将请求的数据使用 protobuf 协议做序列化压缩数据处理，最后通过 rpc 客户端发起网络请求，请求到达 rpc 服务端后，读取并反序列化请求的消息，最后通过既定的路由指派特定的函数处理消息。返回处理结果的时候依然会对响应消息做序列化处理并通过网络返回，到达服务调用方的时候还需对消息做反序列化处理，最后服务调用方获取需要的处理结果。

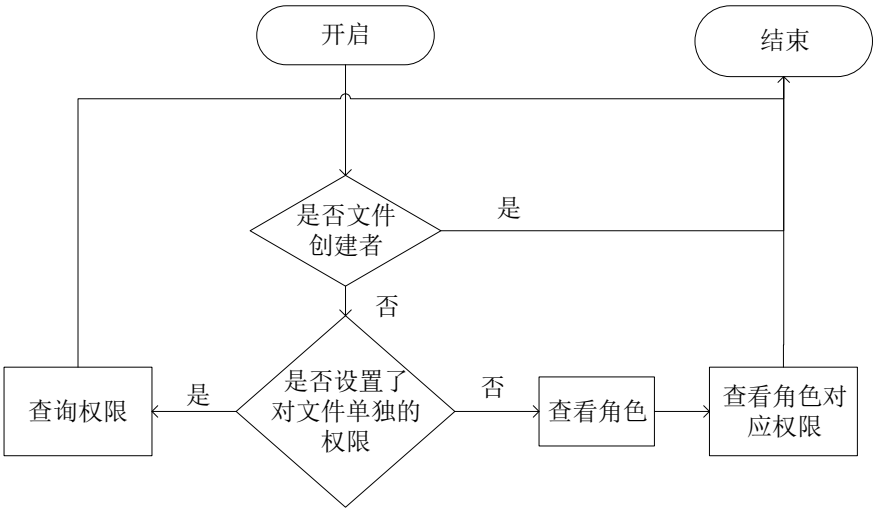

图 5-9 用户对文件权限查询流程图

用户对于团队中单个文件的权限查询如图 5-9 所示。首先查看用户是否是此文件的

创建者，如果是，则返回该用户对该文件拥有所有的权限（删除、修改、分享等），如果不是则判定团队创建者或者管理员是否给该用户对于该文件设定了权限，如果设定了权限，则返回设定的权限。如果没有则判定在用户在团队中的角色。角色分为团队创建者、管理员、普通用户，针对不同的角色相应返回不同的权限。如果是团队创建者或者管理员，则返回对文件的所有操作权限；如果是普通用户，则返回除删除之外的权限。

5.3.2 监控功能

该模块主要是对请求链路和请求消耗时间的日志记录，便于发现延时较长的接口以及查看导致延时较大的相关的业务操作信息。同时也会记录用户的在网盘上的相关敏感操作，比如删除、下载、分享等操作。该模块的实现依赖于 Gin 框架的 MiddleWare 机制，所有请求须通过 TimingMiddleWare 和 SessionTraceMiddleWare 过滤。

请求耗时监控时序图如图 5-10 所示。

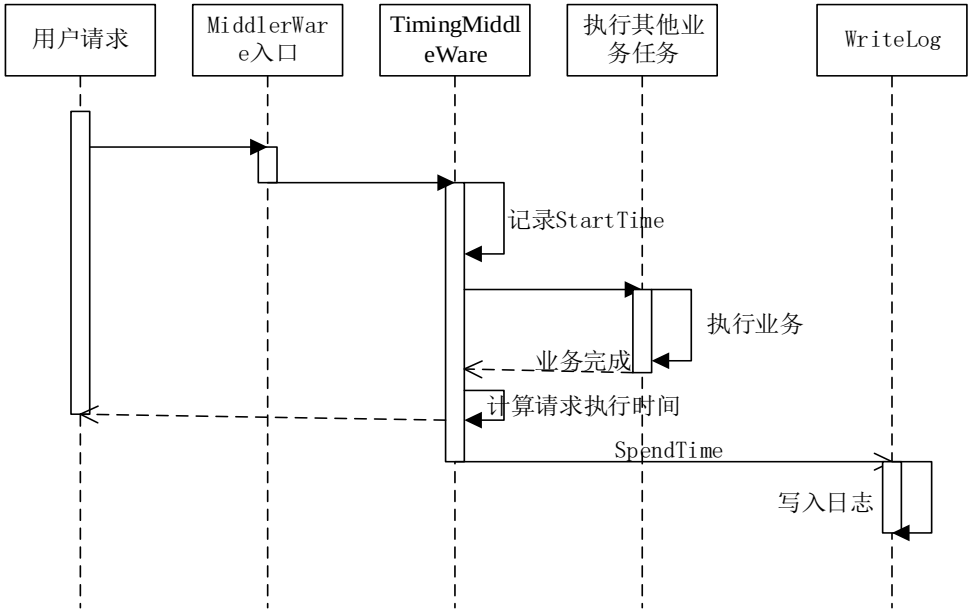

图 5-10 请求耗时监控时序图

请求耗时监控是对所有请求进行拦截，计算每次用户的具体请求的耗时时间。当用户请求被 TimingMiddleware 拦截之后，首先通过 time.now 方法记录下当前的时间，然后利用 Context.next()方法，允许用户执行具体操作的业务，当用户执行完成之后，即可调用 Time.Since(startTime).Nanoseconds()记录下消耗时间，最后将请求信息和耗时时间写入日志文件。

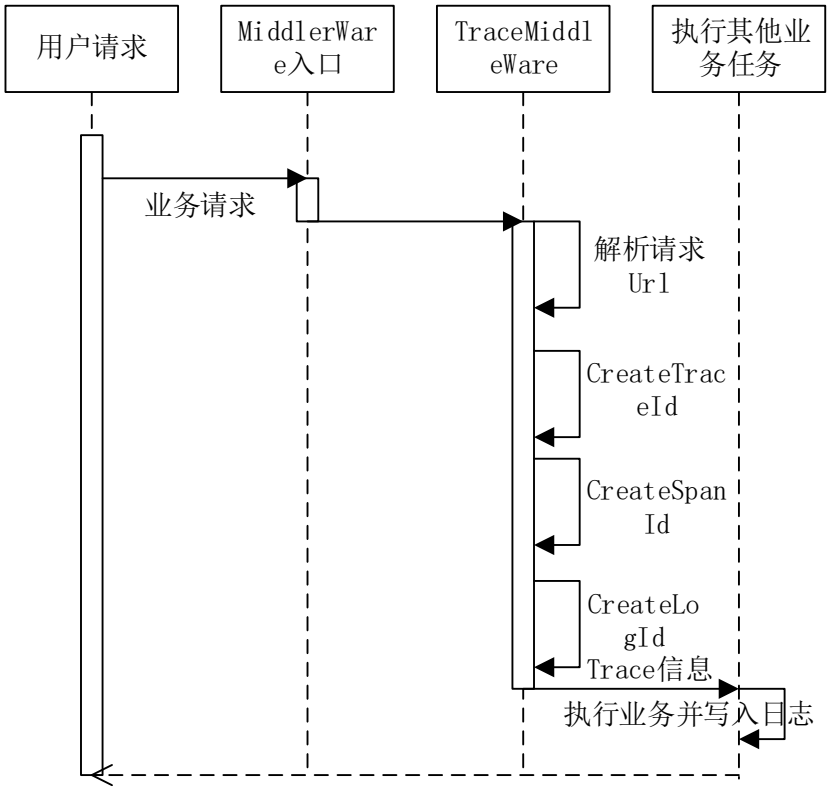

图 5-11 请求链路监控时序图

请求链路监控如图 5-11 所示。SessionTraceMiddle 方法会拦截用户的客户端请求，所有在该 ApiGroup 中的请求均会被拦截，生成 Trace 并注入到 Context 中。其中 Trace 的生成策略是由 traceId、spanId 和 LogId 共同组成的。TraceId 由请求的机器 IP、时间和进程号组成，在同一时间同一个机器的一个进程只会执行一个任务，TraceId 也就是唯一的。SpanId 和 LogId 都是由八位的随机数字随机生成。然后通过 Gin 框架将 trace 信息设置到 context 中，context 作为参数也会注入每个方法的参数列表中，随后请求链路中的每个方法上就都可以获得 trace 信息。每个模块或者方法执行业务的同时可以将必要的业务执行情况和 trace 信息一起写入日志文件，这样用户执行请求链路上的所有执行过程都可以记录。在具体模块中，会重新生成 SpanId 表示在执行不同模块的过程，在每个方法中又可以重新生成 LogId 以表示该方法中执行的过程。这样做的好处是可以在日志中清晰的查看在具体模块中的执行过程和具体方法中的执行过程，可以方便开发者和运维者跟踪定位问题。

文件操作监控功能时序图如图 5-12 所示。

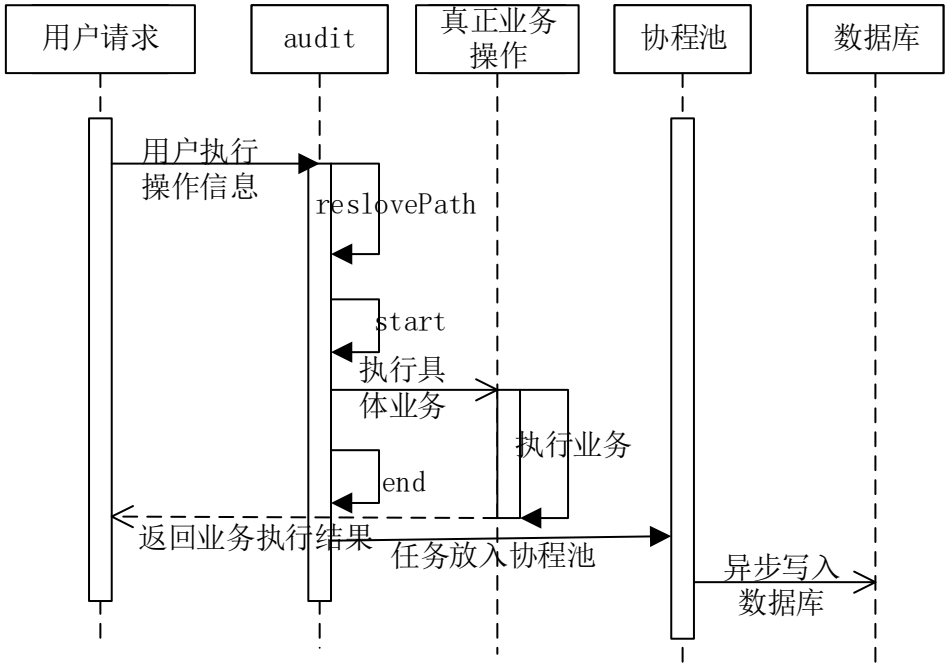

图 5-12 文件操作监控时序图

文件监控主要是记录用户对文件的删除、分享、下载等操作，以达到用户在网盘上的操作可追溯的目的。首先需要利用 Gin 框架的 MiddleWare 对 Http 请求拦截，然后解析 URL 和请求方法（POST、GET、PUT 等）。根据 URL 和请求方法的组合判断是否需要对该请求执行监控。如果不需要执行监控则直接调用 Context.next()方法，执行具体的业务。对于需要监控的方法，首先使用 start()方法，记录下用户、资源对象、具体操作、以及用户的 IP 等信息，然后再调用 Context.next()使得请求可以继续执行其业务操作。当获得了用户本次操作的具体信息之后则调用 end()方法，异步化存入数据库中，而异步化则需要引入并实现协程池。

实现协程池的核心思想是维护一个先进先出的队列，队列中存放了一定数量的 Worker 也就是工作协程。执行任务的流程是首先检查协程池中是否有空闲的 Worker，如果有则取出来执行任务，如果没有空闲的 Worker 则阻塞等待直至有空闲 Worker 可用。每个 Worker 执行完任务之后都会被放还至协程池中。在协程池获取 Worker 的伪代码过程：

伪代码 5-5：GetWorker
输入：pool 输出：worker
1. waiting←false,n←pool.idleWorkerNums 2. lock 3. if n ≤ 0 then 4. waiting←true 5. else w ←pool.workers[n]

```
6.  end if
7.  unlock()
8.  if waiting  then
9.      while(freesignal)
10.         lock
11.         w ←pool.workers[n]
12.         unlock()
13.     end while
14.  end if
15.  return w
```

伪代码 5-5 是从协程池中获取空闲 Worker。其中在获取 worker 部分由于可能同时存在多个地方同时获取 Worker，所以需要对真正执行获取 Worker 的部分做加锁的操作，避免产生并发错误的问题，freesignal 是一个接受协程池中有空闲 Worker 信号的协程，它是并发安全的。通过引入协程池的方式不仅可以避免程序大规模使用协程时内存暴涨，同时也能减少系统的调度压力。在文件操控过程中，执行完 end 函数后将写数据库的任务放入协程池，即可由协程池帮忙调度任务的执行。异步化执行写监控数据任务，可以提高系统的响应时间，并且未大量地入侵业务代码，具有良好的可扩展性。

5.3.3 文件分享功能

文件分享的链接分享功能主要用于用户之间分享文件，用户通过页面上的分享图标，创建 URL 的方式来将文件分享给其他用户，其他用户得到分享 URL 之后（如果有密码，输入密码），打开即可查看文件列表，选择需要的文件下载。直接分享是用户直接搜索选择被分享者，分享文件。被分享着可以在文件列表中查看文件。

文件分享的过程中，首先如果是在团队空间中则需要验证用户对于该文件资源的权限（是否有分享权限），在用户自己的用户空间则不需要验证。然后是创建分享链接，给本次分享赋予权限（查看，下载，保存到网盘等）以及设置分享的失效时间，失效时间可分别设置为 1 天，3 天，7 天和永久。在数据库中失效时间记录的是到期时间，如果是 0 则代表该链接分享永久有效。分享文件的时序图如图 5-13 所示。

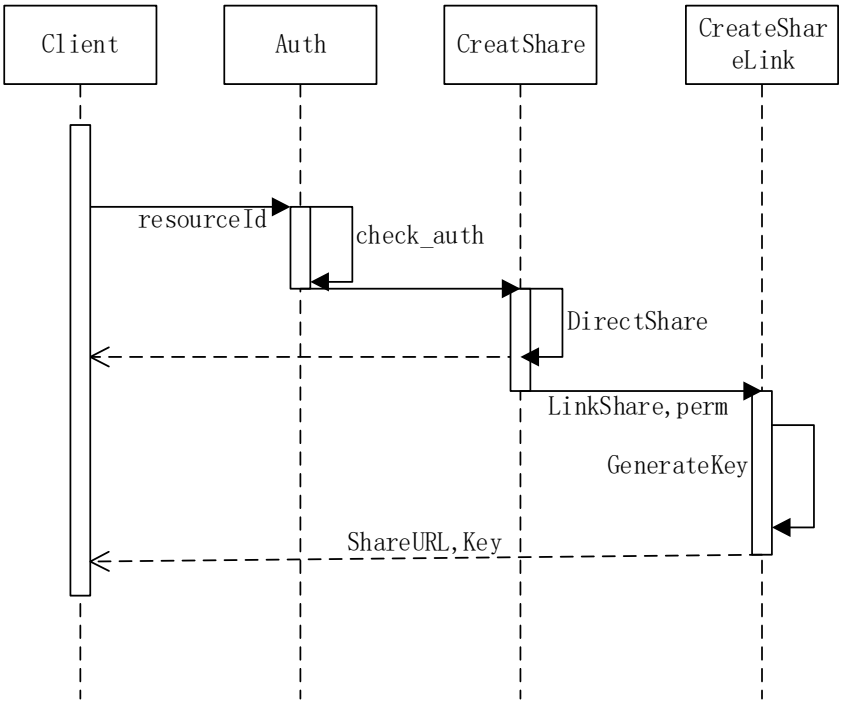

图 5-13 文件分享时序图

创建链接主要使用了 Auth 模块的 Check_auth 方法，判定用户是否对资源有分享的权限，然后后台根据是直接分享还是链接分享做出响应，如果是直接分享，则直接将资源放入分享记录表中，如果是间接分享的话，通过设置权限，有无密码以及有效期等可配置选项生成一个唯一链接。该链接的字符串使用的是 UUID 生成的 15 位字符，其中链接的访问密码是使用 GenerateKey 生成的一个六位随机字符。最后将这些信息存入数据库 Mysql 中。用户访问的时候根据分享链接定位唯一的一条记录，然后根据用户传入的参数验证权限、密码以及有效期，验证通过之后，用户即可访问下载。

5.4 通知模块

本文系统的通知功能分为钉钉通知和邮件通知。用户在使用分享文件的功能、更新分享权限的功能、邀请团队成员以及解散团队功能的时候会触发通知功能。通知模块也作为一个组件服务提供给其他模块的功能调用。实现该功能还使用了设计模式中的单例模式，在整个系统中 DingClient 和 MailClient 均只存在唯一实例，初始化的时候利用 Once.Do 方法保证初始化代码只执行一次。

通知模块的类图如图 5-14 所示。

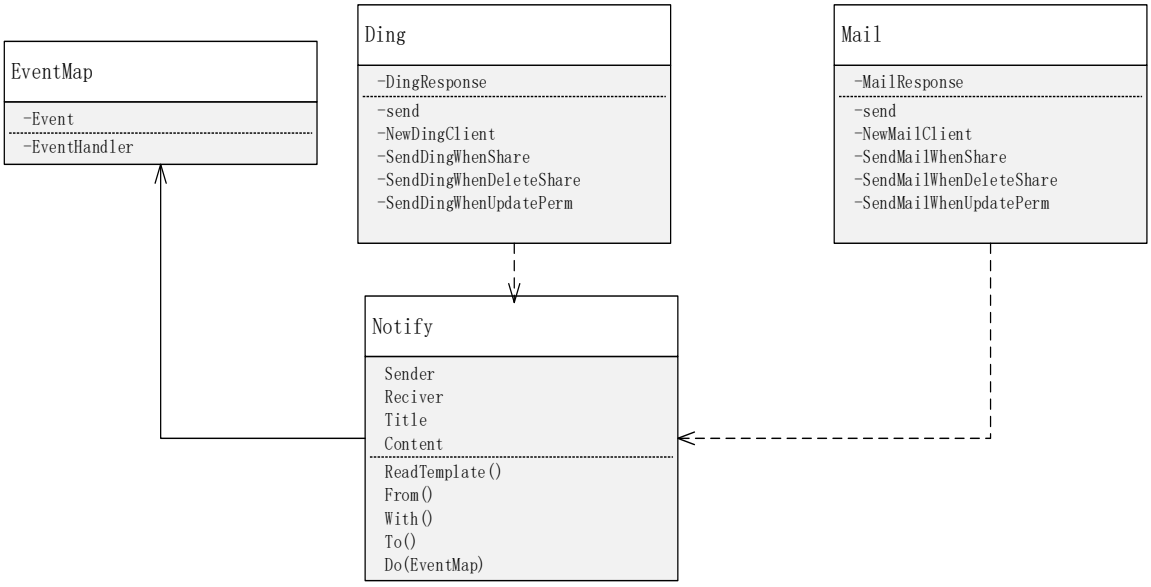

图 5-14 通知模块类图

在该模块中 Ding 和 Mail 类实现 Client 和定义业务中需要实现通知的功能，所有的通知事件和通知的处理函数都需要注册到 EventMap 中，再由 Notify 中方法去解析通知事件中关键的发送者、接收者以及主题等信息，解析完成之后，根据 EventMap 中的事件名找到对应需要读取的模板文件，根据解析的信息完成模板文件中字符串的替换。最后调用 Dingding 和 Mail 提供 Restful 接口，完成通知。

5.5 备份模块

本模块的实现是基于 Ceph 的备份机制目前还不是特别完善而采用的一种异地备份的方案。备份是指将 Ceph 集群里面存储的文件数据通过网络传输到另外一个异地机房的 Ceph 集群中。这样就可以应对机房级别的硬件故障或者系统故障，一旦出现集群的故障，可以切换存储集群，以此达到降低集群故障导致的影响的目的。在本文系统中通过文件索引树来描述完整的网盘文件存储结构。而文件索引树是由多个文件节点组成的，文件节点 FileNode 的数据结构如表 5-1 所示。

表 5-1 文件节点数据结构

属性名称	数据类型	Json 字段	字段含义
Name	String	name	文件名
Tag	Int	tag	表示该节点的状态（0 无状态，1 保留 2 传输，3 删除 ）
MD5	String	Md5	文件 MD5 串
FileSize	Int	fileSize	文件大小
Type	Int	type	文件类型
Children	[]*FileNode	children	子节点

文件索引树的数据结构如表 5-2 所示。

表 5-2 文件索引树数据结构

属性名称	数据类型	Json 字段	字段含义
Root	*FileNode	无	根节点
Total	int64	total	总的文件数量（包括文件夹数量）
Files	int64	files	文件的数量
Size	int64	size	文件的大小
Folds	int64	folds	文件夹的数量
Net	int64	net	网络传输数量

备份功能的时序图如图 5-15 所示。

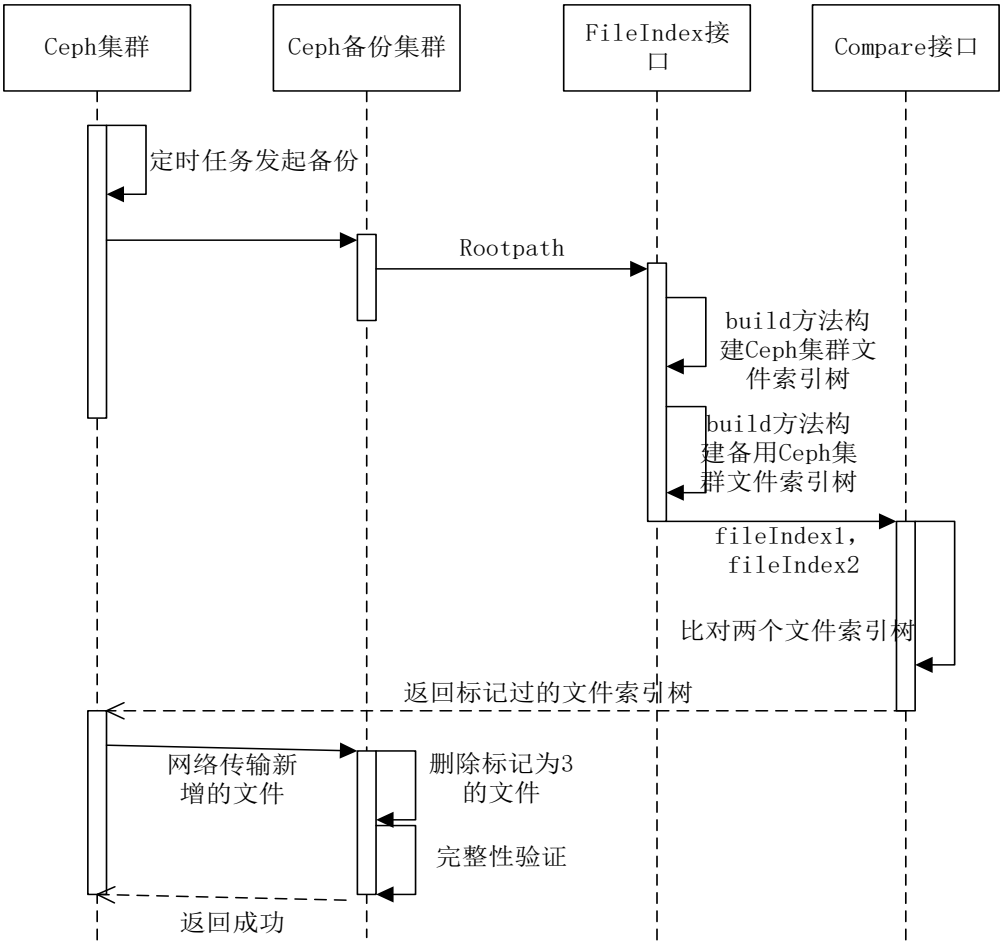

图 5-15 备份功能时序图

首先通过自定义的定时任务发起备份的操作。一般定时任务的设定时间为凌晨或者周末等非工作时间，用户对系统的使用需求较小，服务器的 IO 较为空闲的时间段。这样的选择对用户体验较好并且对系统的压力一定程度上有所缓解。然后任务启动的时候

读取本系统的文件存储根目录，从根目录开始扫描 Ceph 集群获得本地索引，与此同时也会扫描备用 Ceph 集群的本地文件，这时候就用 build 方法同时构建出两份文件索引树，一份文件索引树是描述 Ceph 集群上的文件存储状况，另外一份文件索引树是描述备用 Ceph 集群上的文件存储状况。然后调用 Compare 接口的比对方法可以获得两份文件索引树的差别，在主 Ceph 集群的文件索引上记录下文件的存储状态，也就是 FileNode 数据结构上的 Tag 字段。当标记完成之后，从根节点开始解析文件索引树，依据每个节点的解析结果在备份的 Ceph 集群上构建文件目录，根据 Tag 字段的值确定文件操作的策略，在 Tag 字段上如果是 1 则是保留原文件，如果是 2 那么就是需要传输的新文件，如果是 3 则是删除原文件。备份过程的文件同步也是多节点并行提高效率，引入中间件 Redis，扫描文件目录索引的时候将所有文件以 NameSpace+MD5 的格式生成的唯一数据信息存入 Redis 中，当其中某个节点需要操作节点文件时，先将在 Redis 里面存入的该值删除再去操作文件，Redis 相关的 Get 与 Set 操作封装成原子操作，以分布式锁的方式解决多个节点同时操作用一份文件产生的冲突问题，这样多节点并行处理文件索引树也可以提高文件备份的效率和性能。当操作执行完成之后，调用 IntegrityCheckFile 接口校验备份的完整性，文件的完整性依然使用文件信息摘要 MD5 是否相同判断，具体是比对文件索引树中记录的 MD5 的值和传输完成的文件的 MD5。

当备份功能的执行到达设定的时间之后，需要对备份接口做限流处理来避免影响用户使用。触发限流功能使用 cron 工具实现，配置为"0 0 10 ? * *"（每天上午 10 点触发）。限流的处理过程是：首先配置一个列表容量为 100 的 list 作为桶，每秒生成 2 个令牌放入桶中，如果桶满了则不可再放入，当需要处理文件时，如果是大文件则取走 2 个令牌执行，如果是小文件则取走 1 个令牌，当桶中的令牌为 0 时文件处理任务阻塞，等待能获得足够的令牌时继续执行。

限流的伪代码如下。

伪代码 5-5：令牌桶限流
输入：task,bucket 输出：grant
1. taskTime←task.time,capacity←100 2. rate←2,grant←false,tokens← bucket.currentTokens 3. now←time.now 4. token← min（capacity, tokens + (now - taskTime) * rate） 5. if (token<1) then 6. grant←false 7. else 8. bucket.currentTokens ← token-1 9. grant←true 10. end if 11. return grant

从伪代码 5-5 中可以看到当令牌桶里面的令牌 token 不够的时候，任务的执行授权不通过，会继续留在队列中，等到能够拿到令牌的时候即可执行备份文件的上传操作。

5.6 本章小结

本章主要详细阐述了如何构建一个高可用的 Ceph 存储集群，以及介绍了文件模块、群组/监控模块、通知模块、备份模块的主要功能的详细设计与实现。在文件模块主要实现的有普通文件的上传下载功能、大文件的上传功能、文件的安全功能、以及文件秒传功能，其中在该模块中文件的存储策略提高了磁盘的利用率，提高了文件上传的效率，加密以及完整性校验给文件安全方面提供了保障。在群组/监控模块主要实现了团队管理的功能、权限的管理功能以及文件的分享功能，对于文件共享以及细粒度的文件权限控制，用户请求的链路监控和耗时监控让开发和系统运维人员更了解系统的运行状况。在通知模块实现了邮件通知和钉钉通知的功能，用户可以即时获得网盘的各类通知。在备份模块实现了定时向 Ceph 集群的异地备份的功能，提高了网盘系统应对机房系统级的硬件故障的能力。

第六章 系统测试

通过对分布式网盘系统的重要模块的详细设计与实现，本文系统可以部署运行。按照本文系统的实际测试需求，系统测试分为功能性测试和性能测试。通过系统测试，验证系统各项功能的可用性。

6.1 测试概述

6.1.1 测试平台机器配置

本次系统测试的硬件环境主要由实习公司提供机器，包括服务部署机器以及存储集群机器。

测试所需机器环境配置表如表 6-1 所示。

表 6-1 机器环境配置表

机器类别 / 名称	服务器详细参数	存储服务器	客户访问机
操作系统	CentOS 7.2 (x64 架构)	CentOS 7.2 (x64 架构)	Windows10
CPU 型号	Intel Xeon E7-4890 v2	Intel Xeon E5-2658	I7
主频	2.8GHZ	2.1GHZ	2.8GHZ
核数	8 核	8 核	4
内存	64GB	64GB	8
硬盘容量	1TB	10TB	1TB
网络带宽	企业内网 1000M 宽带	企业内网 1000M 宽带	企业内网 1000MB 宽带
Docker 版本	17.03		
浏览器型号版本			Chrome 浏览器；版本 60.0.3112.78
Ceph 版本	0.87		

6.1.2 测试环境部署关键点

本文系统部署在装有 Docker 的服务器中，部署节点实例的数量可自由控制，本文部署的节点实例是 3 个。系统部署在 Docker 中的基础镜像是 CentOS7.2，为了防止容器中运行的网盘服务因为异常崩溃，配置一个 supervisord（linux 中的一个程序监控工具）的配置脚本 start.conf，并设置好自动重启以及重启次数，这样可以降低网盘服务不可用的机率。同时在 Docker 中安装负载均衡软件 Nginx，配置好负载均衡。配置文件的部分代码如下。

```
    upstream clouddisk.com {
            server 10.97.61.29:32767;
            server 10.97.61.29:32768;
            server 10.96.67.2:32769;
    }
    server{
        listen 80;
        server_name www.cooper.com;
        location {
            proxy_pass              http://www.cooper.com;
            proxy_set_header    Host                $host;
            proxy_set_header    X-Real-IP           $remote_addr;
            proxy_set_header    X-Forwarded-For     $proxy_add_x_forwarded_for;
        }
}
```

本文系统需要六台机器存储机器，每三台一组，一组是主 Ceph 存储集群，另外一组是异地机房的 Ceph 存储集群。安装部署好 Ceph 集群之后，在容器中执行 mount 命令挂载到远程的 Ceph 存储集群。命令如下：

```
mount -t ceph 10.96.67.29,10.96.65.29,10.96.65.28,10.96.64.29,10.96.64.28:/cooper /cooper -o
name=admin,secret=AQCSpvpaYdfZABAAzuIsYiZhRTQTbuSgJ7KqHA==
mount -t ceph 10.96.67.29,10.96.65.29,10.96.65.28,10.96.64.29,10.96.64.28:/cooper-back
/cooper-back -o name=admin,secret=AQCSpvpaYdfZABAAzuIsYiZhRTQTbuSgJ7KqHA==
```

6.1.3 测试计划

本文系统的测试包括了系统功能测试以及性能测试。

（1）功能性测试主要是对本文系统的设计与实现的各个功能模块进行测试，采用黑盒测试的方法验证功能是否符合需求分析、是否完全可用以及接口是否稳定。

（2）性能测试主要是对接口的响应时间进行测试、用户上传下载的速度测试和磁

盘的性能测试。

6.2 系统功能测试

本文系统的功能性测试是对文件模块、团队权限模块、监控模块、通知模块和备份模块的主要功能的测试。

6.2.1 文件模块测试

该模块的主要功能有普通文件的上传、大文件的上传、文件安全功能、文件下载功能、文档预览功能、文件秒传功能。

（1）普通文件上传功能测试。

测试结果如表 6-2 所示。

表 6-2 普通上传功能测试用例表

测试点	测试内容	测试方法	测试结果	是否符合测试预期
单个普通文件上传功能验收	测试单个文件上传功能	上传 1 个 80M 左右文件	当前用户可以成功上传文件，并在文件列表页面展示	符合
多个普通文件上传	测试多个文件同时上传功能	5 个 80M 左右文件	当前用户可以成功上传所有文件，并在文件列表页面展示所有文件	符合

文件上传效果如图 6-1 所示。

图 6-1 文件上传效果图

表 6-2 是对普通文件上传的测试用例，分别对单个 80M 大小的文件上传和批量文件上传进行了验证，并在用户的页面上查看是否上传成功，并对文件删除。经过测试验证，普通文件上传的效果和预期相符。

（2）大文件上传功能测试。

测试结果如表 6-3 所示。

表 6-3 大文件上传功能测试用例表

测试点	测试内容	测试方法	测试结果	是否符合测试预期
大文件上传功能	测试单个文件上传功能	1 个 5G 左右文件	当前用户可以成功上传文件，并在文件列表页面展示并且上传速度较快，如果取消文件上传，下次上传从取消处开始上传	符合

表 6-3 是大文件上传功能测试的可用性测试用例，将一个 5G 文件上传至网盘中，可以看到用户可以上传成功，并且在浏览器的控制台可以看到多次调用了上传接口。当用户主动断开，下次上传同一份文件的时候，可以看到无需上传断开处之前的分片，系统继续从断开处上传文件，符合预期。

（3）文件安全/下载功能测试。

测试结果如表 6-4 所示。

表 6-4 文件安全和下载功能测试用例表

测试点	测试内容	测试方法	测试结果	是否符合测试预期
文件安全/文件下载	测试文件上传至服务后是否加密	使用 FTP 工具登陆服务器一个普通文件，使用本文系统下载同一份文件	使用 FTP 工具 FileZilla 下载的文件无法打开，使用本文系统下载的文件可以正常打开	符合

表 6-4 是文件安全测试用例。首先将文件上传至网盘后，随后下载。同时使用 ftp 工具至服务中下载同一份文件，可以看到使用网盘下载的文件可以正常打开，而使用 ftp 下载的文件因为并未对文件解密不可打开，说明存储在磁盘中的文件是加过密的，用户通过网盘上传下载文件中有加密解密的过程，符合预期。

（4）文件预览功能测试。

测试结果如表 6-5 所示。

表 6-5 文件预览功能测试用例表

测试点	测试内容	测试方法	测试结果	是否符合测试预期
视频/音频/图片文件预览验证	测试系统是否可以查看常见格式的视频、音频和图片文件。文件格式包括 mp3、mp4、wav、mp4、rmvb、jpg、png 等。	打开网盘存储的对应格式的文件。	可以预览常见格式的视、音频和图片文件。	符合
文档预览验证	测试是否可以查看常用文档格式文件。文档文件的格式包括 PDF、Word、PPT。	打开网盘存储的对应格式的文档文件。	可以预览常见格式的文档文件。	符合

表 6-5 是文件预览功能的测试用例表。经过测试，系统对常见格式的视频、音频和图片等格式的预览支持良好。对于文档格式，系统将 Word 和 PPT 格式的文件转换成 PDF 之后，也可以良好支持预览功能。

文件预览效果如图 6-2 所示。

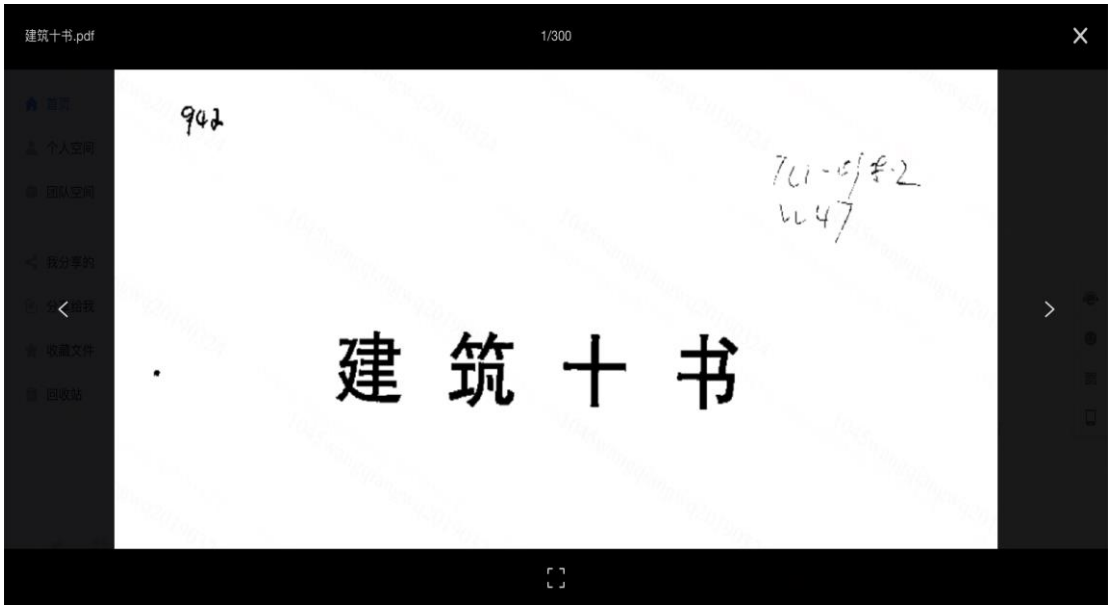

图 6-2 文件预览效果图

（5）文件秒传功能测试。

测试结果如表 6-6 所示。

表 6-6 文件秒传功能测试

测试点	测试内容	测试方法	测试结果	是否符合测试预期
文件秒传功能验证	测试上传用户空间已有的文件时是否秒传。	连续两次上传一个 2G 的文件，查看第二次上传时上传的速度。	第一次是正常的大文件上传，在用户在第二次上传完文件时，当校验完成 MD5 时，直接显示上传成功，并在列表显示文件。	符合

表 6-6 是文件秒传功能的测试用例。用户先后两次上传同一份文件的时候，可以看到第二次上传的文件在校验完 MD5 后，系统判定在该用户空间下已存在同一份文件，直接保存了文件数据信息，并没有实际的上传操作，达到了秒传的效果，有较好的用户体验，符合预期。

6.2.2 群组/监控模块测试

该模块测试的功能主要有文档分享中的权限测试以及在团队中团队管理的权限测试。

（1）团队权限功能测试。

测试结果如表 6-7 所示。

表 6-7 团队权限功能测试

测试点	测试内容	测试方法	测试结果	是否符合测试预期
管理员角色权限测试验证	测试是否可以邀请和删除成员，以及给普通成员设置权限	在团队中设置用户 A 为管理员，设置权限	用户可以正常邀请删除成员，并可以给单个用户设置权限，包括普通用户的所有权限	符合
普通成员角色权限验证	测试是否可以上传、查看、分享文件，是否不可删除文件。	普通成员用户，在网盘对文件执行查看、下载、分享和删除操作	用户可以查看、下载、分享，但不可删除文件	符合
单个用户对特定文件的权限	测试给单个用户设定不可下载文件后，是否依然可以下载	管理员设置普通用户对单个文件无下载权限	用户无法下载无下载权限的特定文件	符合

表 6-7 是权限测试用例表。用户在不同的角色下对应的用户权限也不同，管理员和给普通成员设定权限的功能，而普通成员只有查看下载文件的权限。同时当给团队成员关于文件设置不同的权限的时候，用户在执行文件操作的时候权限也不同。测试中当给用户设置了不可下载特定文件时，用户执行下载操作时将弹出无权限的消息弹框，符合预期。

（2）监控功能测试。

测试结果如表 6-8 所示。

表 6-8 监控功能测试

测试点	测试内容	测试方法	测试结果	是否符合测试预期
请求耗时监控	请求耗时是否监控记录	查看日志文件	日志文件中已记录	符合
请求链路监控	请求链路是否监控记录	查看日志文件	日志文件中已记录	符合
文件操作监控	用户关于文件操作是否监控	查看数据库 audit 表	用户操作已被记录	符合

表 6-8 是监控功能测试用例，通过分别查看日志文件和数据库 audit 表，可以看到相关记录都已存在，符合测试预期。

（3）文件分享功能测试。

测试结果如表 6-9 所示。

表 6-9 文件分享功能测试

测试点	测试内容	测试方法	测试结果	是否符合预期
直接分享	用户是否可在分享列表查看他人分享的文件	查看分享列表	显示他人分享的文件	符合
公开链接分享	用户是否能够打开他人的分享链接并下载文件	直接打开链接	可打开并下载文件	符合
加密链接分享	用户是否能够打开他人的加密分享链接并下载文件	打开链接并键入密码	可打开并下载文件	符合

表 6-9 是文件分享功能测试用例表。经测试，用户可以使用直接分享功能、公开链接分享和加密链接分享功能，符合预期。

链接分享效果如图 6-3 所示。

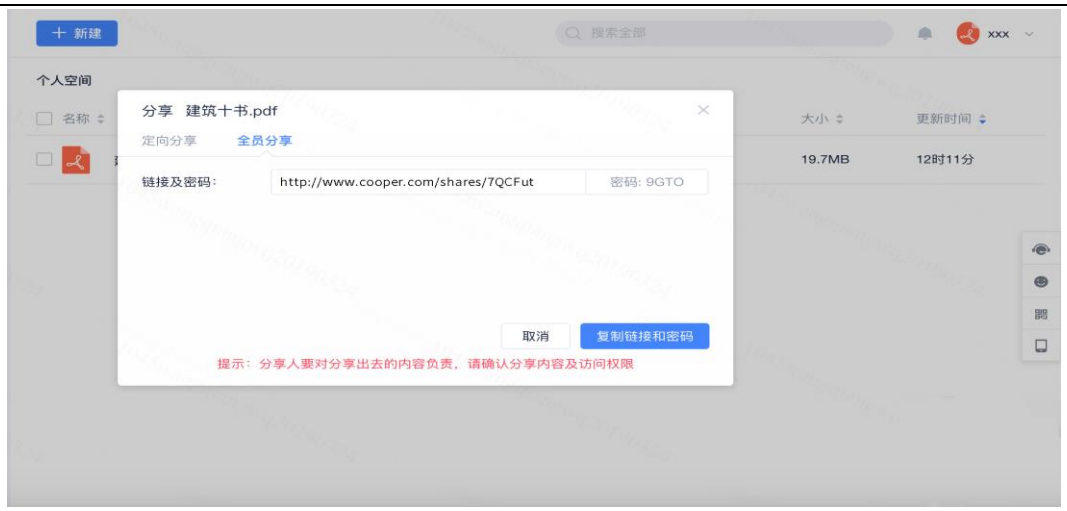

图 6-3 链接分享效果图

6.2.3 通知模块测试

测试结果如表 6-10 所示。

表 6-10 通知功能测试

测试点	测试内容	测试方法	测试结果	是否符合预期
邮件通知	用户在分享文件、邀请成员是否收到邮件通知	查看是否收到邮件通知	收到通知	符合
钉钉通知	用户在分享文件、邀请成员是否收到钉钉通知	查看是否收到钉钉通知	收到通知	符合

通知功能效果如图 6-4 所示。

图 6-4 通知效果图

表 6-10 是通知功能测试用例表。用户在使用直接分享文件、邀请成员的时候可以

收到钉钉和邮件通知，符合预期。

6.2.4 备份模块测试

测试结果如表 6-11 所示。

表 6-11 备份功能测试

测试点	测试内容	测试方法	测试结果	是否符合预期
定时备份	测试备份是否完成	比对网盘存储集群和备份集群的文件是否一致	存储集群和备份集群的文件一致	符合

表 6-11 是备份功能的测试用例。在测试中分别登陆存储集群和备份集群，比对存储集群和备份集群的文件目录以及文件数是否一致，可以看到结果一致，符合预期。

6.2.5 功能测试小结

根据以上功能测试得出以下结论。

（1）本文系统用户可以良好的完成文件上传、下载、预览和秒传等操作。可以成功上传大文件且达到预期效果，传输速度满足需求。

（2）本文系统系统对用户在团队中的管理功能和文件分享功能支持良好，对于单个文件权限的控制精确无误。在监控中可以记录完整请求链路和耗时时间。

（3）通知模块能够正常运行，用户在操作涉及通知的功能时，通知都能正确的接受和发送。

（4）备份模块中的任务能够定时发起，良好的完成备份功能。

综上所述，各个模块的功能都通过验收，本文系统可以良好运行，满足需求分析的要求。

6.3 性能测试

随着系统的使用用户越来越多，系统性能测试包括以下方面：常用的查询接口测试（不涉及文件读写操作）、文件上传下载性能测试以及 Ceph 集群的文件读写测试。

6.3.1 常用查询接口性能测试

对于系统常见的接口，用户对于响应时间指标的要求一般在 1200ms 以下。如果响应时间超过 1200ms，系统的使用者会明显感受到系统页面的卡顿。

本文系统所有接口中用户最常使用的是展示文件列表接口，所以对该接口测试。

测试思路：使用 Go 语言编程模拟一定数量的用户并发查询文件接口，并记录接口的响应时间。用户并发数和接口响应时间如图 6-5 所示。

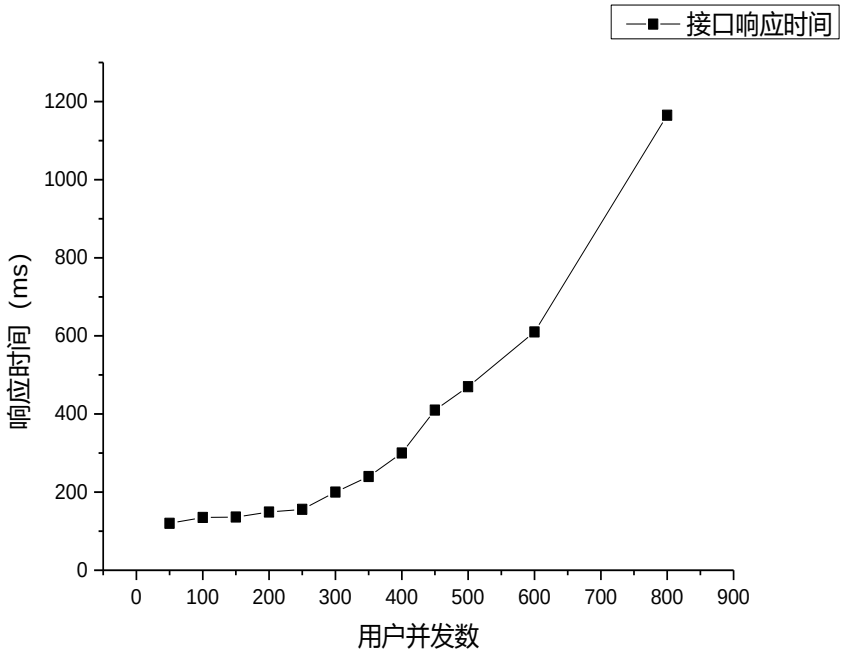

图 6-5 用户并发数和接口响应时间关系图

从图 6-5 可以看出在用户数小于 300 的时候，随着用户数量变高，接口响应时间并没有发生特别明显的变化，响应时间维持在 200ms 以内的优秀水平，但是当用户的数量超过 300 时，可以明显观察到随着用户数量的增加，对接口响应时间的影响越来越明显，当用户数量达到 800 的时候，基本达到用户忍受响应时间的极限，所以可以判定此时的用户并发数已达到系统不影响用户体验的极限。如果用户并发数继续增加，系统会明显卡顿。为了解决系统卡顿的问题，可以考虑增加网盘服务的节点，节点之间负载均衡以提升系统的性能。

6.3.2 文件上传下载性能测试

本文系统的文件上传下载测试是在企业内网环境下进行。通过分别测试普通文件和大文件的上传与下载，观察传输速率是否满足用户对于传输性能的要求，并在同等网络下与市场上其他网盘相比较，测试结果如表 6-12 所示。

表 6-12 文件上传下载性能测试

测试点	测试内容	测试方法	测试结果
单个普通文件的上传与下载	测试平均上传下载消耗时间	单个 100M 以内文件上传并下载	平均上传速率 3.07M/s 平均下载速率 3.98M/s
批量普通文件的上传与下载	测试平均上传下载消耗时间	多个 100M 以内文件上传并下载	平均上传速率 2.86M/s 平均下载速率 3.55M/s
单个大文件的上传与下载	测试平均上传下载消耗时间	单个 2G 文件上传并下载	平均上传速率 2.24M/s 平均下载速率 2.35M/s

批量大文件的上传与下载	测试平均上传下载消耗时间	多个 2G 文件上传并下载	平均上传速率 2.17M/s 平均下载速率 2.30M/s

从以上分别对大文件和普通小文件的单个与批量测试的结果来看，普通小文件由于不需要文件切分，文件的完整性校验和加密的速度相对来说也较快，所以上传与下载的速度是优于大文件的。

而大文件则因为切割文件的原因一定程度上比普通小文件稍慢，但传输的速度依然满足用户需求，完全达到了理想情况。

市场上其他网在同一网络环境下测试结果如表 6-13 所示。

表 6-13 市场上网盘在同一网络下性能测试表

名称	测试内容	测试方法	测试结果
百度云盘	测试平均上传下载消耗时间	单个 2G 文件上传并下载	平均上传速率 1M/s 平均下载速率 300k/s
腾讯微云	测试平均上传下载消耗时间	单个 2G 文件上传并下载	平均上传速率 1.24M/s 平均下载速率 1.05M/s

由于百度云盘在未开会员的情况下速度受到了限制，而腾讯微云的表现则稍好。相比本文系统，在同样的企业网络环境下本文系统有明显的速率上的优势。

6.3.3 Ceph 集群读写性能测试

Ceph 集群的性能分别体现在四个方面：随机读、顺序读、随机写和顺序写四个方面。本文系统使用 fio 测试工具编写测试脚本对以上四个方面分别测试。同时使用一块装有 ext4 文件系统的原生磁盘和装有 CephFS 文件系统的磁盘比较在读写方面的性能。

首先使用 ssh 登陆测试集群，分别在两个主机上创建测试目录/cephfs_test/data，然后运行以下命令：

```
fio -filename=/cephfs_test/data -direct=1 -iodepth 1 -thread -rw=read -ioengine=psync
-bs=4k -size=2G -numjobs=30 -runtime=1000 -group_reporting -name=mytest
```

其中可更改 rw 参数值来测试，rw 参数的可选值有 read（顺序读）、write（顺序写）、randread（随机读）、randwrite（随机写），还可通过 size 参数更改文件的大小。本次测试设置的块大小均为 4K。

吞吐量是衡量存储系统读写性能的常用指标，它的含义是指系统每秒读写文件数据的数据量，在不同读写模式下吞吐量的大小均有所不同。本文系统经过测试，汇总 Ceph 与普通磁盘吞吐量测试对比结果如图 6-2 所示。

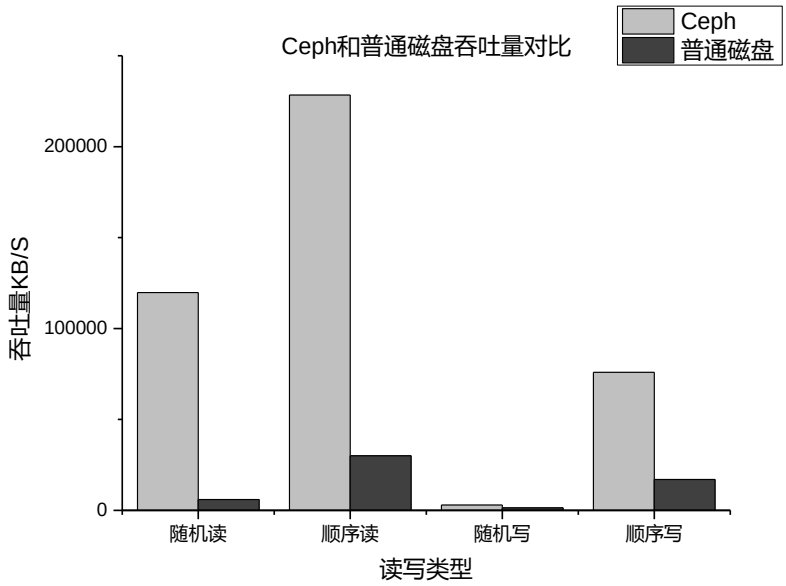

图 6-6 Ceph 与普通磁盘吞吐量对比

从图 6-6 中可以看出，Ceph 在随机读、顺序读、顺序写方面与普通磁盘相比有较大的领先，顺序读可以达到 240M/S 的速度，而两者在随机写指标上并没有太大差距，并且数值都相对较低，仅有 4M/S 左右的速度。从图 6-6 还可以看出顺序读写相对随机读写有较大的优势。

IOPS 也是衡量存储系统读写性能的常用指标，它的含义是指系统每秒的 IO 读写次数。和吞吐量指标类似，在不同的读写模式下，每秒的 IO 次数均不同。Ceph 与普通磁盘 IOPS 的对比如图 6-7 所示。

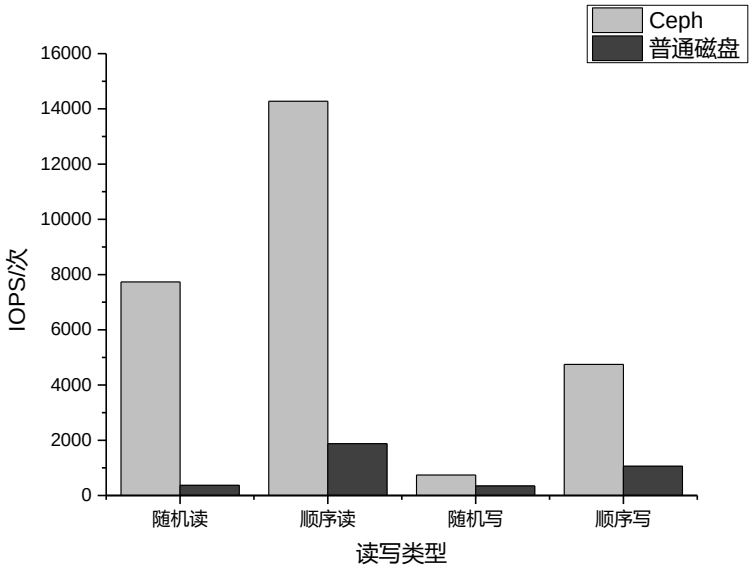

图 6-7 Ceph 与普通磁盘每秒读写次数对比

从图 6-7 可以看出，磁盘读写次数对比趋势与吞吐量的对比趋势基本相同，Ceph 依然能够提供良好的读写性能，Ceph 的顺序读 IOPS 是普通磁盘的 7 倍之多，虽然 Ceph

在随机写模式下 IOPS 很低，但也是普通磁盘的 2 倍多，其他读写模式下也有明显的性能优势。所以综上可以认为 Ceph 文件系统与普通磁盘相比，有更好的性能表现。

6.3.4 性能测试小结

通过以上的性能测试可以得出以下结论：

（1）网盘系统最常用的文件列表展示接口 QPS 可以满足目前企业用户的规模。当网盘系统卡顿时，可以相应的增加网盘服务节点，以降低响应时间。

（2）在经过与市面上常见的网盘系统在同一网络环境下对比测试，由于网盘系统属于内部网络服务，上传和下载的传输速度优势明显。可以满足用户对于网盘传输性能的要求。

（3）在经过 Ceph 集群的性能测试后可以发现。Ceph 文件系统与普通磁盘相比，在磁盘吞吐量和 IOPS 方面有较大的读写上的优势，能够稳健的提供存储服务。

综上所述，网盘系统分别在存储集群层、存储服务层以及用户使用体验上均满足用户的需求，性能测试结果与预期相符。

6.4 本章小结

本章主要介绍了测试平台，测试环境部署的关键点，制定了测试计划。功能测试分别对文件模块，群组/监控模块，通知模块，备份模块作了详尽的测试；性能测试中对常用的接口、文件接口以及集群读写性能做了测试。最后得出结论，本文系统基本满足了用户对于网盘系统的功能和性能方面的需求。

第七章 总结与展望

7.1 本文工作总结

本文以企业信息化建设为背景，帮助企业实现海量文件数据的云端管理为目标，基于 Ceph 分布式文件系统、AES 加密算法和 MD5 算法以及令牌桶算法限流等技术实现了一个高性能可靠的分布式网盘系统。同时在大文件的处理上进行了优化，提高了系统性能。在系统中实现异地灾备的功能，提高了系统的抗灾和应对故障的能力。本文的主要完成的工作如下。

（1）分析了企业用户对网盘的常用需求。文件的管理、权限的管理、安全加密、抗故障能力等需求。

（2）对分布式网盘系统功能的设计与实现。设计与实现了文件存储模块，针对不同的文件大小采用不同的上传策略，实现了大文件上传的功能，优化大文件在存储端的存储结构；实现了文件加密的功能，保障了文件恶意流出之后，文件原内容不会被轻易获得。还实现了预览和秒传的功能，增强了用户体验和减少物理存储空间的浪费。设计与实现了群组和监控模块，针对企业对权限的特殊要求，在用户角色权限模型的基础上引入了更细细粒度的文件权限，同时在监控模块中实现了对请求的耗时和链路的监控功能，提升了开发体验以及降低了系统问题定位的难度，对于未来系统功能的优化也打下了基础。在通知模块中实现了邮件和钉钉通知的功能，即时性的提高也可提升企业的运行效率。在应对系统级故障和灾难方面，实现了异地备份功能。

（3）对系统进行了详细的功能测试和性能测试。保证系统在功能和性能方面满足用户的需求。

本文系统开发完成之后，对系统的功能和性能分别进行了测试验证。目前该系统已在企业内部上线运行，运行情况良好，基本满足了用户的需求。

7.2 展望

本文实现的基于 Ceph 的分布式网盘系统可以替换其他企业提供的公有云盘，并且在公司内部的网络环境下传输效率比其他共有网盘更高、安全方面有着更优的表现、以及有可控的故障应对方案，一定程度上降低了实体存储设备的支出，降低了成本。但是，基于 Ceph 的分布式网盘系统仍然不够完善，需要进一步研究。

（1）系统功能方面。本文实现的分布式网盘系统，是基于 B/S 架构的，文件的切割等操作依赖于浏览器的计算能力，浏览器的性能有限，并且没有下载续传的功能，所以需要基于 C/S 模式开发客户端，研究并实现下载续传。既能提升网盘性能又可以实现续传功能。

（2）安全方面。本文系统在加密过程中的密钥过于简单且策略固定，未来需要考

虑文件加密密钥的动态性；同时对文件也可以使用多重加密算法对文件进行加密，增强加密等级，保证文件的安全性；升级网站为 HTTPS 协议，保障网站的安全。

（3）移动端访问。目前只能在浏览器端使用本系统，用户的使用便捷程度依然不足。在当今的移动互联网时代，可以考虑开发手机移动客户端，随时随地的方便用户处理文件数据，可以进一步的提高效率和用户体验。

致谢

时光飞逝，短短三年的研究生求学生涯即将结束，本论文的写作在此也即将结束。自在实习企业单位寻找论文研究课题开始起到现在论文的撰写即将结束也已经过了一年多。回首这段时光，在论文题目的拟定、论文的开题、论文的撰写以及论文修改的过程中，我得到了诸多的帮助，正是因为这些帮助才有了本论文的完成。所以在本论文完成之际，我要对那些帮助过、关心过、鼓励过我的人，致以我最真诚的感谢。

首先，感谢我的校内导师陶军老师。本论文的完成和陶军老师的悉心指导和严格要求是分不开的。陶军老师的治学严谨和上课时的幽默风趣给我留下了深刻的印象。陶老师在论文的开题、写作、修改等方面提供了诸多的帮助。在开题的过程中，陶老师帮助我指出了论文研究的关键点，并在细微之处也提出了很多宝贵的建议和意见，使我顺利通过了开题。在论文写作过程中，陶老师也经常指导并帮助我，给出了许多宝贵的修改意见。在此，谨向陶老师致以我最真诚的感谢。

其次，感谢我的校外导师金尚高工和实习企业同事侯心主，感谢他们在繁忙的工作之余依然对我的论文关心，在技术上给了许多帮助。特别需要感谢我的同事侯心主，是他在论文系统的实现方面提供给我诸多思路，帮助我查找论文系统相关的技术书籍。同事们平易近人、工作认真负责的态度，在职业生涯方面给了我诸多启发。

然后，感谢一起学习成长的同学们。感谢他们的陪伴和相互鼓励，希望在未来大家能一起奋进。

还需要感谢在开题、答辩过程中百忙之中抽空评阅的老师。

最后特别要感谢我的父母。父母在背后无理由默默的支持一直是我前进的动力。你们在物质和精神上无私的奉献，才有了今天的我。在未来，我将努力工作，不辜负父母殷切的期望。

参考文献

[1] Wang C, Chow S S M, Wang Q, et al. Privacy-preserving public auditing for secure cloud storage[J]. IEEE transactions on computers, 2013, 62(2): 362-375.

[2] Yang J, He S, Lin Y, et al. Multimedia cloud transmission and storage system based on internet of things[J]. Multimedia Tools and Applications, 2017, 76(17): 17735-17750.

[3] Zhang X, Gaddam S, Chronopoulos A T. Ceph distributed file system benchmarks on an openstack cloud[C]//2015 IEEE International Conference on Cloud Computing in Emerging Markets (CCEM). IEEE, 2015: 113-120.

[4] 冯朝胜, 秦志光, 袁丁. 云数据安全存储技术[J]. 计算机学报, 2015, 38(1): 150-163.

[5] 傅颖勋, 罗圣美, 舒继武. 一种云存储环境下的安全网盘系统[J]. 软件学报, 2014, 25(8): 1831-1843.

[6] 程靓坤. 基于 Ceph 的云存储系统设计与实现[D]. 中山大学, 2014.

[7] Bacis E, di Vimercati S D C, Foresti S, et al. Protecting Resources and Regulating Access in Cloud-Based Object Storage[M]//From Database to Cyber Security. Springer, Cham, 2018: 125-142.

[8] 陆峰. 基于 FastDFS 的企业私有云盘的设计与实现[D]. 吉林大学, 2016.

[9] 谢金星. 基于云存储的网盘系统设计与实现[D]. 2016.

[10] Acquaviva L, Bellavista P, Corradi A, et al. Cloud Distributed File Systems: A Benchmark of HDFS, Ceph, GlusterFS, and XtremeFS[C]//2018 IEEE Global Communications Conference (GLOBECOM). IEEE, 2018: 1-6.

[11] Panidis A, Natanzon A, Cohen S. Initializing backup snapshots on deduplicated storage: U.S. Patent Application 10/031,690[P]. 2018-7-24.

[12] Yang C T, Chen C J, Chen T Y. Implementation of ceph storage with big data for performance comparison[C]//International Conference on Information Science and Applications. Springer, Singapore, 2017: 625-633.

[13] Weng J Y, Yang C T, Chang C H. The Integration of Shared Storages with the CephFS and Rados Gateway for Big Data Accessing[C]//2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC). IEEE, 2018, 2: 93-98.

[14] Borges G, Crosby S, Boland L. CephFS: a new generation storage platform for Australian high energy physics[C]//Journal of Physics: Conference Series. IOP Publishing, 2017, 898(6): 062015.

[15] 凌升杭. 基于 Ceph 分布式存储系统的安全访问控制设计和实现[D]. 东南大学, 2016.

[16] 苏艳森. 分布式文件存储平台文件备份与恢复系统设计与实现[D]. 杭州: 浙江大学, 2008.

[17] 李翔. Ceph 分布式文件系统的研究及性能测试[D]. 西安电子科技大学, 2014.

[18] 李宜. OpenStack 存储优化与负载均衡的研究与实现[D]. 华南理工大学, 2017.

[19] Mohammad A S, Mangalam B V, Joshi P, et al. Channel accessible single function micro service data collection process for light analytics: U.S. Patent Application 14/870,801[P]. 2017-3-30.

[20] Chung S M, Shieh M D, Chiueh T C. A Security Proxy to Cloud Storage Backends Based on an Efficient Wildcard Searchable Encryption[C]//2018 IEEE 8th International Symposium on Cloud and Service Computing (SC2). IEEE, 2018: 127-130.

[21] Watkins N, Sevilla M, Jimenez I, et al. Ceph: An Open-Source Software-Defined Storage Stack[J].

[22] Vaidya M, Deshpande S. Comparative analysis of various distributed file systems & performance evaluation using map reduce implementation[C]//2016 International Conference on Recent Advances and Innovations in Engineering (ICRAIE). IEEE, 2016: 1-6.

[23] Oral S A, Weil S, Settlemyer B W, et al. Performance and Scalability Evaluation of the Ceph Parallel File System[J].

[24] Weil S A, Brandt S A, Miller E L, et al. Ceph: A scalable, high-performance distributed file system[C]//Proceedings of the 7th symposium on Operating systems design and implementation. USENIX Association, 2006: 307-320.

[25] 杨飞, 朱志祥, 梁小江. 基于 Ceph 对象存储的云网盘设计与实现[J]. 电子科技, 2015 (2015年10): 96-99.

[26] Weil S A, Brandt S A, Miller E L, et al. CRUSH: Controlled, scalable, decentralized placement of replicated data[C]//SC'06: Proceedings of the 2006 ACM/IEEE Conference on Supercomputing. IEEE, 2006: 31-31.

[27] 沈良好, 吴庆波, 杨沙洲. 基于 Ceph 的分布式存储节能技术研究[J]. 计算机工程, 2015, 41(8): 13-17.

[28] 周江, 王伟平, 孟丹, 等. 面向大数据分析的分布式文件系统关键技术[J]. 计算机研究与发展, 2014, 51(2): 382-394.

[29] 杨飞, 朱志祥, 梁小江. 基于 Ceph 对象存储集群的负载均衡设计与实现[J]. 计算机系统应用, 2016, 25(4): 268-271.

[30] Wahid M N A, Ali A, Esparham B, et al. A Comparison of Cryptographic Algorithms: DES, 3DES, AES, RSA and Blowfish for Guessing Attacks Prevention[J]. 2018.

[31] Mahajan P, Sachdeva A. A study of encryption algorithms AES, DES and RSA for security[J]. Global Journal of Computer Science and Technology, 2013.

[32] Rizvi S A M, Hussain S Z, Wadhwa N. Performance analysis of AES and TwoFish encryption schemes[C]//2011 International Conference on Communication Systems and Network Technologies. IEEE, 2011: 76-79.

[33] Yalame M H, Farzam M H, Bayat-Sarmadi S. Secure Two-Party Computation Using an Efficient Garbled Circuit by Reducing Data Transfer[C]//International Conference on Applications and Techniques in Information Security. Springer, Singapore, 2017: 23-34.

[34] 王磊. 微服务架构与实践[M]. 电子工业出版社, 2016.

[35] 许艳艳, 雷迎春, 龚奕利. 基于可变长分块的分布式文件系统设计与实现[J]. 计算机工程, 2016, 42(5): 80-84.

[36] 张毕涛. 分布式存储系统小文件性能优化方案的设计与实现[D]. 北京邮电大学, 2016.

[37] 高志栋. 基于 Ceph 的分布式存储在数据中心的设计与实现[D]. 兰州大学, 2018.

[38] 黄家栋. 基于增量方式的数据备份技术研究与实现[D]. 华南理工大学, 2017.

[39] Panidis A, Natanzon A, Cohen S. Initializing backup snapshots on deduplicated storage: U.S. Patent Application 10/031,690[P]. 2018-7-24.

[40] 张伯阳, 张晓, 李阿妮, 等. 云存储系统可扩展性评测研究[J]. 计算机应用研究, 2017, 34(7): 1957-1961.