福建工程学院
Fujian University of Technology

毕业设计（论文）

题　　目：分布式文件存储与分享设计与实现

学　　生：邱万利

2017 年 6 月

目录

摘要.....4

ABSTRACT.....5

第一章 绪论.....6

1.1 课题研究的目的及意义.....6

1.2 课题研究的现状.....6

1.3 课题研究的主要内容.....6

第二章 系统需求分析.....8

2.1 可行性研究.....8

2.1.1 社会可行性.....8

2.1.2 用户可行性.....8

2.1.3 技术可行性.....8

2.1.3 经济可行性.....9

2.2 相关技术介绍.....9

2.2.1 开发工具.....9

2.2.2 C/S 模式.....10

2.2.3 ProtoBuf 数据序列化技术.....10

2.2.4 Boost.Asio 网络框架.....11

2.2.5 开发环境.....12

2.3 方案论证.....12

2.3.1 ProtoBuf 序列化文件的选择.....12

2.3.2 开发技术的选择.....13

第三章 功能需求分析.....15

第四章 系统设计.....19

4.1 目录服务器.....19

4.2 文件存储服务器.....19

4.3 文件接口.....20

4.4 系统的数据结构的 ER 图.....20

4.5 数据字典.....27

4.6 实体对应的数据结构 28
第五章 系统实现 32
5.1 系统整体的实现 32
5.2 数据流向 32
5.2.1 文件的上传： 33
5.2.2 文件下载 34
5.2.3 文件分享 34
5.2.4 文件删除 35
5.2.5 分享链接的删除 35
5.2.6 URL 的下载 35
5.2.6 日志服务 36
第六章 总结和展望 37
6.1 总结 37
6.2 不足及未来工作展望 38
致谢 39

摘要

分布式存储[1]系统是 p2p[2] 技术的传输技术拓展。而本分布式文件存储与分享系统，与现在的 CDN[3]系统类似但也是有所不同的。本系统采用 C++编程语言[4]，Boost.Asio[5]以及 Google 的 ProtoBuf[6]开源框架，进行设计。这两个开源库都是经过企业实战检验的，非常适合网络传输和文件格式的定义，其成熟的架构、稳定的性能、简洁的逻辑，使得系统能迅速开发。而本系统主要适用于大文件的存储，以减少当大用户量对同一个文件的读写时对机器的负荷压力。采用分流的方式以缓和文件存储服务器的压力。而且未来是一个共享式[7]的文件存储模式，多个厂商互相提供接口，以一种简单的负载均衡[8]的方式共享资源和解决热点文件高并发下载时对文件服务器和带宽的冲击。本文件系统适合大文件用户，拥有文件系统集群，并采用“碎片化”存储。

关键词：分布式文件存储与分享；C++；Asio；ProtoBuf；

ABSTRACT

Distributed storage system is a p2p technology to expand the transmission technology. The distributed file storage and sharing system, similar to the current CDN system but also different. The system is designed in C ++ programming language , Boost.Asio and Google's ProtoBuf open source framework. These two open source libraries are all verified by the enterprise, very suitable for network transmission and file format definition, its mature architecture, stable performance, simple logic, making the system can be developed quickly. And the system is mainly applied to the storage of large files to reduce the amount of large users when the same file to read and write when the load load on the machine. Using a shunt approach to ease the pressure on the file storage server. And the future is a shared file storage model, a number of vendors to provide each other interface to a simple load balancing way to share resources and solve hot files high concurrent download on the file server and bandwidth impact. This file system is suitable for large file users, has a file system cluster, and uses "fragmented" storage.

Keywords: distributed file storage and sharing; C ++; Asio; ProtoBuf;

第一章 绪论

1.1 课题研究的目的及意义

在这个共享是经济快速发展的的时代，共享式资源的利用也越来越频繁了，如果利用好共享是经济的话，便掌握了行业的主动权。当多个文件同时下载同一个文件时，怎么解决？怎么提高并发度？一个几个 G 的文件当我们下载了 99%时，他断线了怎么办？而分布式文件存储与分享系统可以解决这几个问题，那就是采用多线程[9]的文件传输方式、文件的位图映射保存文件的下载的状态以及按块下载从不同的端下载同一个文件的数据块。这样大大的提高了文件传输的效率和降低文件下载失败过程中的重传代价。

1.2 课题研究的现状

在国外，分布式文件存储已经是比较普及了，但是国内此类应用比较少。而且随着信息爆炸式的增长，网络文件的增加量程大幅上升趋势，这也无形的增加了且也对硬盘的需求。而传统的文件存储方式并不适合这种 TB 级的文件存储和大用户量同时读写同一文件，而分布式的块式文件存储可以解决此类问题。并且可以利用剩余带宽提供第三方的下载服务，以加少企业的运营成本。

1.3 课题研究的主要内容

第一章，主要简单介绍了开题研究的目的、意义，还有国内该课题的研究现状。

第二章，介绍了需求分析、探究可行性分析、方案论证、所有技术、实际开发所用到的平台环境，相关技术，开发工具。

第三章　对功能需求进行分析和设计对应关系

第四章，说明了数据存储结构和网络文件传输结构的设计，ER 图，以及详细设计。

第五章，对系统的功能进行说明，实现了什么功能。

第六章，总结了这次设计遇到的问题，还有受到的启发，以及系统的一些不足之处。

第二章 系统需求分析

2.1 可行性研究

可行性分析就是要评估产品是否值得开发，并且要系统和全面的去分析系统的需求，同时尽可能考虑到各种能够影响到项目的可能性因素，要调查该项目的可行性，分析收集到的资料对项目作出客观的评测。而且必须适应中国的国情，信息指标体系能够满足标准化要求。

2.1.1 社会可行性

随着计算机的普及、互联网的快速发展以及网络信息的爆炸式增长，使得网络分享已经成为人们生活不可或缺的一部份，并且国内资源共享（云盘链接分享）已经逐渐发展为一种文件下载和分享的主要方式。而分布式文件存储与分享为这种方式提供了底层技术支撑，同时也可以通过提高短距离的带宽占用来，降低国内长距离的网络之间的带宽占用（即主干网络）。

2.1.2 用户可行性

该系统面向的是个人、小型分享网站和大型企业企业，个人可以利用本系统进行不同客户端之间的文档的同步，小型分享网站和企业用来进行文件的存储和分享，并且可以利用剩余带宽对本企业拥有的资源进行共享，从而降低整体的系统开销。因此用户可行性是完全没问题的。

2.1.3 技术可行性

该系统开发语言选择的是 C++[10]，C++是一种高级语言，并且拥有较高运行效率，是其他语言所不能比的。而且还支持跨平台，一个平台下编辑多个平台下运行。特别是在 Linux 下部署简单简单，实用性强。并且 Boost 库和 ProtoBuf 经过时间的检验，拥有高效稳定的运行时。为系统的 24*7 提供了稳定的解决方案。

本系统使用的开发软件是 Visual Studio 2017 Community ，数据存储采用 Google ProtoBuf 进行序列化文件存储，在不同的平台上相通的需求可进行相同的配置，减少文件的配置问题。

对于系统架构，我比较倾向于的是 C/S 架构，因为 C/S 灵活性、通用性、易操作性并且重用性好，可进行热更新。C/S 架构升级和维护的方式比较简单，省去了工作量。这种架构已经被大家广泛的运用，已经有很多优秀的项目，积累了大量的宝贵经验这方面。所以技术完全可行。

2.1.4 经济可行性

在考虑到经济这个方面需要考虑的减少开发成本，缩短开发周期，并且还要保证系统的质量与效果，提高系统的工作效率，所以该系统的开发周期在 3 个月左右，包括构想、可行性分析、设计、编码、调试等

硬件方面只要一台本地目录服务器服务器，多台 PC 机用于存储文件。经济方面也是完全可行的。

2.2 相关技术介绍

2.2.1 开发工具

1、Microsoft Visual Studio 软件

Microsoft Visual Studio 2017，简称 VS2017，它是 windows 系统下最完美的一套几乎完整的 IDE，它包括了整个软件生命周期过程中需要的大部分工具。并且提供了高级开发工具、调试功能、可视化编程、支持多种语言、数据库功能和创新功能，也是是目前最流行的 Windows 平台应用程序的可视化集成开发环境。开发本系统所使用的 Microsoft Visual Studio 2017 Community 就是基于.NET Framework 4.6.2。

2、ProtoBuffer 软件

ProtoBuffer 是 Google 开源的模型反转工具，所生成的代码采用编码优化，文件的解析效率比较高，文件所需的空间较小，并且是原生的 C++支持，可跨平

台跨语言进行数据交互。

3、MFC 框架

Windows 系统下的比较常用的可视化 GUI 框架[11]，有着较高的使用率，并且支持空间拖拽进行进行界面设计和布局。

4、Sublime Text 3

Sublime Text 3 是一款非常棒的文件编辑工具，支持多样化的插件，并支持不同开发语言的高亮显示，ProtoBuf 的抽象对象文件技术是利用该软件插件进行编写的。

2.2.2 C/S 模式

C/S 模式，即客户端/服务器模式。客户端通过 UI 界面的数据，传输到中间层进行数据的序列化后，将文件流通过 Boost[12]::Asio 网络框架传输到服务端，对文件进行反序列化后得到正确数据。同时，若 UI 某些数据有返回值，在服务端通过进行数据的序列化后，将文件流通过 Boost::Asio 网络框架传输到 PC 端，解析问价后，将返回值回写到 UI 界面上。C/S 模式最大的优点就是数据安全性高，可支持多对多的文件传输，受到网络影响较小，并且稳定。

2.2.3 ProtoBuf 数据序列化技术

ProtoBuf 是 Google 提供的一个开源高效的序列化框架（采用 C++编写的），虽然类似于 JSON[13]、XML[14]和 Lua 这样的数据表示语言，但其最大的特点是基于二进制，并且其底层采用编码方式进行空间优化，因此比传统的 XML 等数据表示语言表示高效短小。虽然是二进制数据格式进行存储，但并没有因此变得复杂，我们可以按照一定的语法定义结构化的消息格式，然后利用 ProtoBuf.exe 命令行工具，自动生成相关的类，可以支持 C++ 、Go、Java 等多种语言环境。通过将这些生成类添加到项目中，可以很容易的调用相关方法来完成消息和数据结构的序列化与反序列化到文件流。

ProtoBuf 在 GRPC（Google Remote Procedure Call）中是一个比较核心的基础库，作为分布式系统会涉及到大量的不同格式消息的传递，如何简洁高效，

节省带宽的表示、操作这些数据消息在 GRPC 这样的大规模应用于分布式网络系统是至关重要的。而 ProtoBuf 库正好是在效率、数据大小、易用性之间找到了一个很好的平衡点。

2.2.4 Boost.Asio 网络框架

Boost Asio (asynchronous input and output)的是网络和低级别的跨平台的 C++库，为开发者提供使用现代 C ++的做法一致的异步模型，只需关注异步输入输出。Boost Asio 库提供了跨平台的异步数据处理能力（当然它也支持同步数据处理），可进行不同平台间的数据通讯。对于一般的数据文件传输过程需要通过函数的返回值的状态码，来判断数据传输是否成功。但是 Boost Asio 将数据传输分为两个互不关联的步骤：

1. 采用异步的方式开始数据传输。
2. 将传输结果通知调用端

并且 Asio 网络通信库是基于操作系统提供的异步机制，采用前摄器模式（Proactor）实现可移植的异步 IO 操作，不需要使用多线程和锁，有效避免了因多线程和并发编程带来的副作用（如竞争，死锁等）。

Asio 网络库封装了操作系统的 select、kqueue、poll/epoll、overlapped I/O 等机制，实现异步 IO 模型。在同步模式下，程序发起一个或多个 IO 操作，向同一个或不同的 io_service 提交请求，对应的 io_service 把操作转交给操作系统，同步地等待。当 IO 操作完成时，操作系统通知相应的 io_service 实例，然后相应的 io_service 再把结果发回给程序，完成整个同步流程。在异步模式下，程序除了要发起多个 IO 操作，还要定义一个在文件处理完成时用于处理的回调函数。io_service 同样把 IO 控制权转交给操作系统去执行，之后立即返回。调用 io_service 实例的 run()方法可以等待异步操作结果，当异步操作完成时 io_service 从操作系统获取结果，再调用 handler（）执行后续处理逻辑。因此 Aiso 是线程安全的框架。

2.2.5 开发环境

1、软件开发环境

操作系统：Windows 10 专业版

系统类型：64 位操作系统

开发工具：Microsoft Visual Studio 2017 Community

第三方库管理软件：Vcpkg（微软的开源依赖管理工具）

数据库：无

2、硬件开发环境

CPU：Intel(R) Core(TM)i7-7700HQ CPU@ 2.80GHz

内存：8GB DDR4 2400MHz

硬盘：256GB SSD + 1024GB 机械硬盘

2.3 方案论证

2.3.1 ProtoBuf 序列化文件的选择

对于一条消息结构化数据，使用 ProtoBuf 进行序列化后的空间占用大小是 json 文件的 8 分之一，xml 文件的 15 分之一，是二进制序列化文件的 8 分之一，总的来说 ProtoBuf 的优势，是我选择它作为分布式文件存储与分享系统消息传输所考虑的。

1.灵活稳定（便于接口更新）、高效（效率经过 Google 的编码优化，传输效率比 Json 等高很多）；

2.便于操作；我们可以通过按照一定的语法来定义结构化的消息模型，生成不同语言的文件。在不同的编程语言上例如 C++、Go、Python 等调用不同的实现方法，来进行数据的序列化和反序列化。

3.语言支持：原生支持 C++、java、python

适用 ProtoBuf 的场合：

1.需要需要进行不同平台间做消息交换的，对消息大小比较敏感的。那么使

用 ProtoBuf 序列化库就再适合不过了，它语言、平台无关，消息空间相对 JSON 和 XML 文件格式等节省很多空间和时间。

2.较小的数据。如果你是大数据（例如文件存储等），用它就不太适合。

3.对于项目语言是 C++、Go、python 的，它们可以使用 Google 的原生序列化类库，序列化和反序列化的效率非常高。其它的语言需要第三方或者自己写，因为每个人的水平各不同，编写的质量就很难保证就使得序列化和反序列化的效率得不到保证了。

4.因此 ProtoBuf 序列化库还是非常好用的，也被很多开源项目用于数据通信的消息格式的定义工具，在 Google 也是核心的基础库。

2.3.2 开发技术的选择

MFC 还是有着很多优势的：

1.体积小，静态编译后体积也不大。

2.在 windows XP、windows7、windows 10 等上兼容性都较好。

3.并且对系统底层 windows API 的调用较方便。

4.支持拖拽式的界面设计和消息绑定机制。

采用 SHA512[15]作为文件唯一标识

现在 MD5[16] 的消息摘要已经很普遍了，但是对文件采用 MD5 进行标识并不理想，其碰撞的概率实在很大，使得文件的安全性（数据冲突）大大的降低了。所以一开始采用的是 SHA1[17]的消息摘要算法生成文件的唯一标识。但是最近 Google 却宣布已经成功破解了 SHA1 的消息摘要算法。当恶意用户生成两个数据不相同但器 SHA1 的值却相同的文件进行上传时，会使得后上传的文件，出现数据的文件的不正确，使得 SHA1 变得不太安全了，因此我决定采用 SHA512 的消息摘要算法生成文件的唯一标识，毕竟在未来十几年中 SHA512 是不可能被破解的。但是由于 MD5 的效率比 SHA1 和 SHA512 的速度快很多，所以采用 MD5 散列算法来进行文件的验证还是可以接受的。

之所以采用 ProtoBuf 定义消息传输文件的格式，是因为在客户端与服务端通信时较多的是大量短消息（例如：认证信息、目录信息、文件信息、分享信息、

PC 端信息等），对于这种消息 Protocol 拥有着各种优势，并且还可以降低带宽的使用使得使用成本降低。

采用 Boost Aiso 是因为它（支持 TCP/IP[18]以及 UDP）是一个跨平台的、毕竟可以用在网络通信上，也可以作用在串口通信等一系列基于文件流的数据交互的 C++网络库。虽然计算机进行网络传输的设计方式有很多种。但是 Boost.Asio 的方式（异步通知）远远优于其它的设计方式。它在 2005 年就被包含进 Boost“C++ ‘准’标准库”，然后被大量 Boost 的用户测试并在很多项目中使用，比如 Remob、ibtorrent、PokerTH 等应用。奇虎 360 的网络框架也是借鉴了 Asio 的设计理念。

并且 Boost.Asio 在网络通信、COM 串行端口和文件上成功地抽象了输入输出的概念。使得我们可以基于这些进行同步或者异步的输入输出编程。

最重要的是 Boost.Asio 网络框架是与平台无关的支持 Boost.System、Boost.Regex、Boost.DateTime、OpenSSL 库，因为他底层提供了相应的平台实现，不需要我们考虑太多的底层实现，而且只需要简单的代码既可以实现的加密文件传输和会话。

第三章 功能需求分析

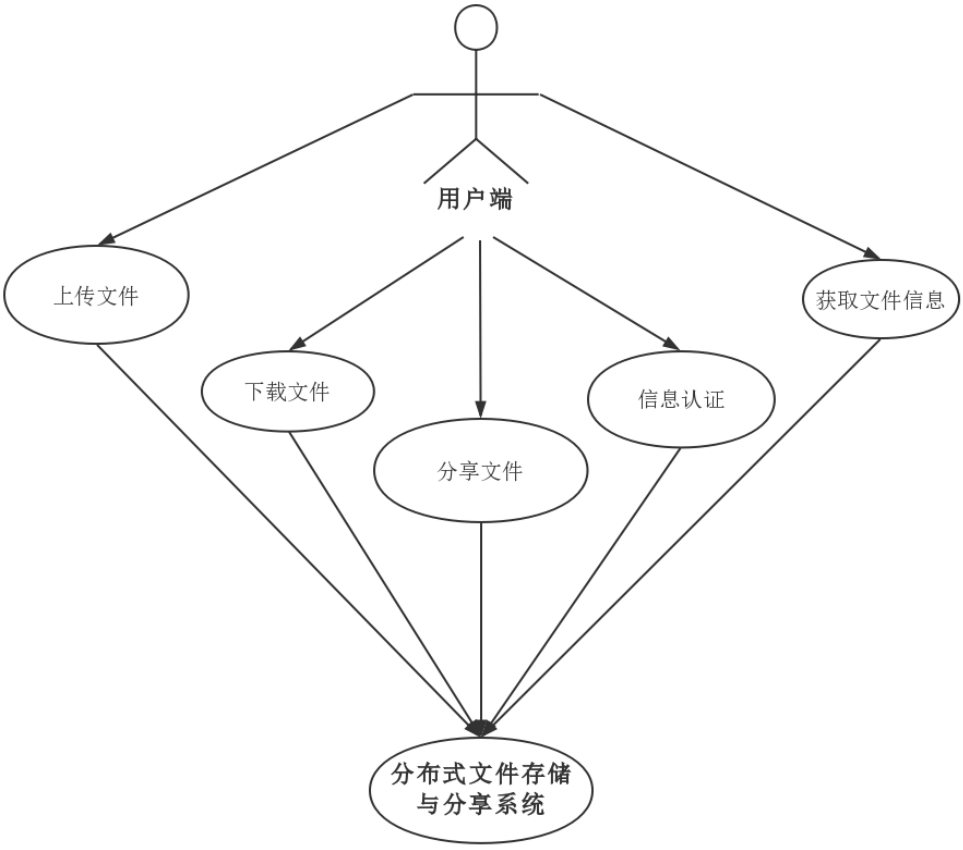


图 3-1 系统整体需求分析图

对于一个分布式文件存储与分享系统，该系统至少应提供文件的上传、分享、下载功能，并且要有认证机制和获取文件信息的基础

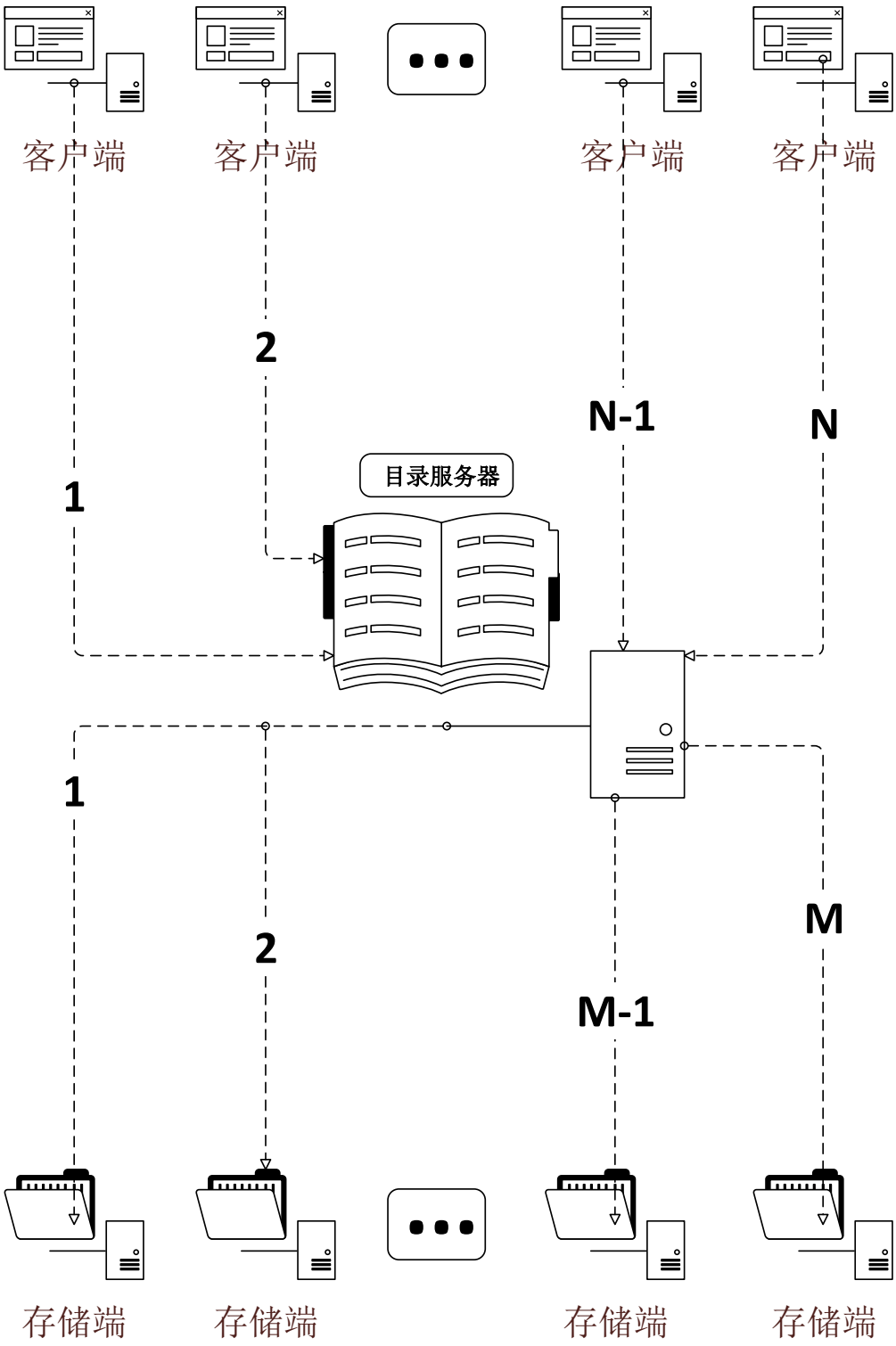


图 3-2 初步设计结构图

对于初步设想，采用客户端通过目录服务器进行文件的管理，客户端将文件分块信息发送给目录服务器，之后从目录服务器获取存储端的认证信息，之后与存储端进行文件传输，传输完成时通知目录服务端。文件下载时，先向目录服务

端请求，后从存储服务器获取文件。

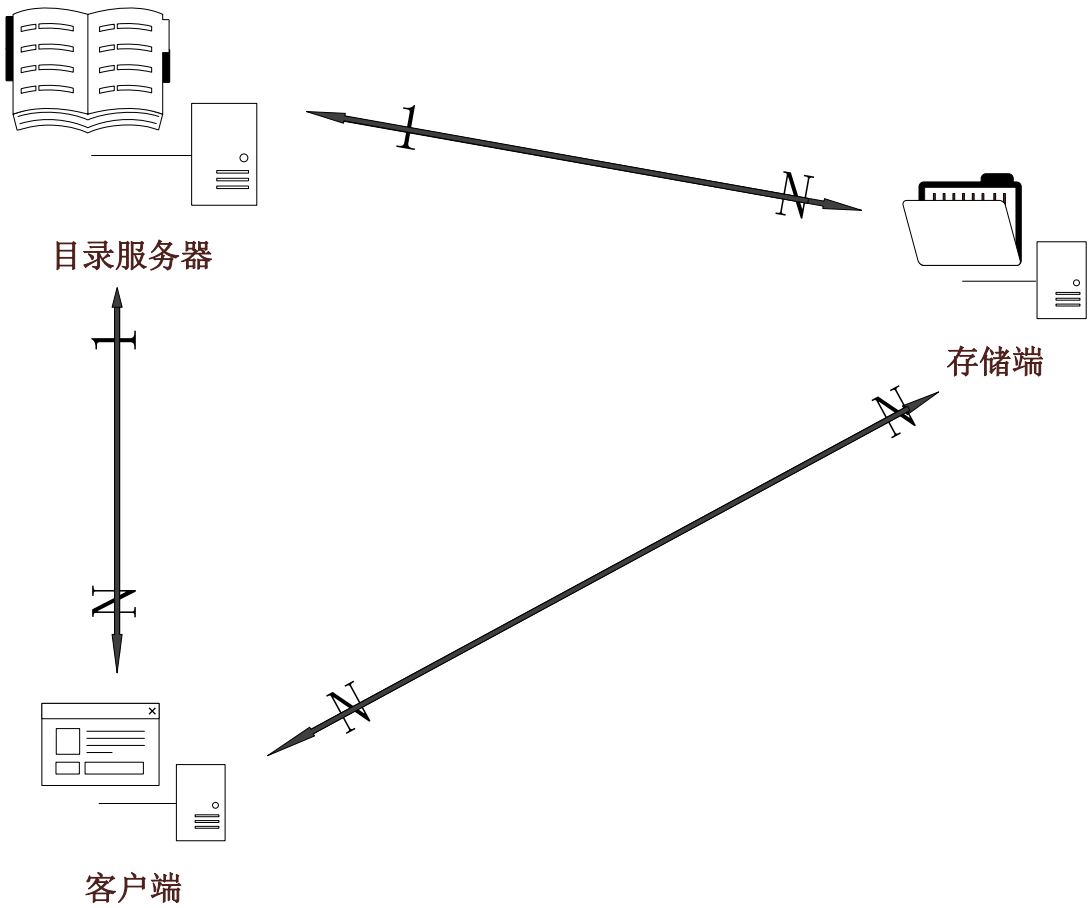


图 3-3 模块的对应关系

目录服务端、客户端以及存储服务器之间的对应关系，因为是分布的文件存储，所以必须要有一个块信息分布的存储服务器，用于查询块信息的位置，便于上传与下载。而客户端与存储服务器的数量可以是不确定的，便于系统存储的拓展。

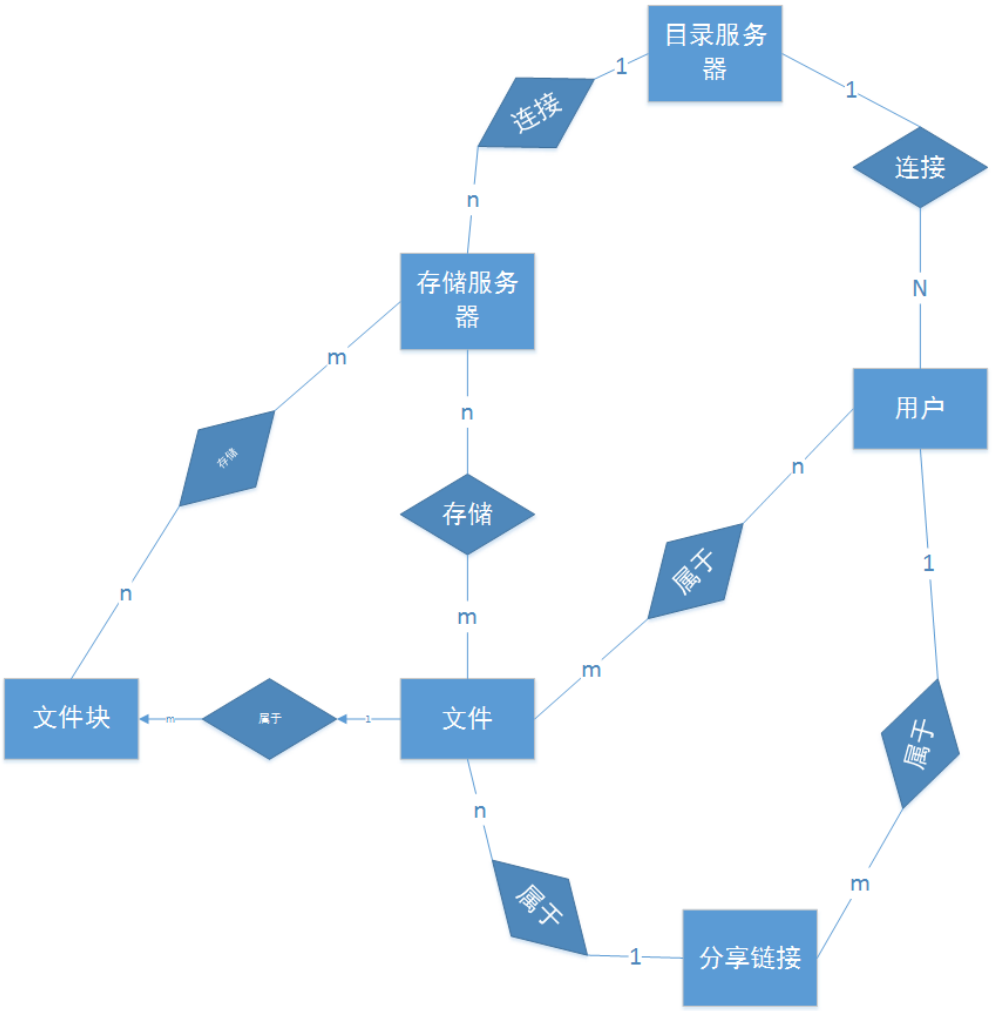


图 3-4　实体之间的 ER 图

目录服务器中保存的分享链接和文件块信息是唯一的，而对应的链接客户端和存储服务器则是一对多的关系。

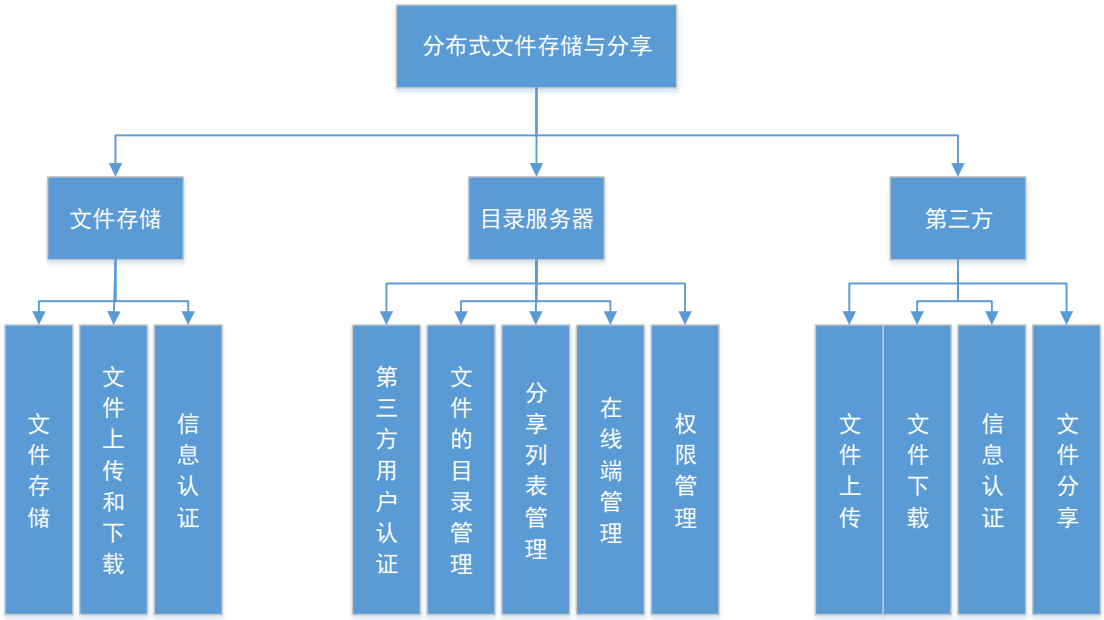
一个文件可以存储到多个文件存储服务器，并且一个文件存储服务器可以存储多个文件块，所以他们之间是多对多的关系。

一个文件包含多个文件块，一个文件块包含在一个文件中，他们之间是一对多的关系。

一个用户对应多个分享链接，用户与分享链接，之间是一对多的关系。

第四章 系统设计

本分布式文件存储与分享系统采用目录服务器和文件存储服务器，并提供文件分享和下载接口、验证模块、文件传输模块



系统整体设计图

4.1 目录服务器

1. 验证模块：身份验证、身份授权、并分配临时密钥
   对链接本目录服务器的链接进行身份验证。
2. 文件管理模块：文件查询、目录文件下载、目录文件上传，目录的更新
3. 日志模块：操作日志、连接用户信息存档
6. 存储服务端状态模块：连接状态、将用户编号转化为相应的 IP 地址
7. 文件分享模块：分享链接的解析、分享链接的生成

4.2 文件存储服务器

1. 文件管理模块：文件块的读取、文件块的写入、文件块的更新
2. 文件下载模块： 文件块的上传、文件块的下载

3．认证模块：链接的认证

4.3 文件接口

1．文件上传：目录上传，文件块的上传

2．文件下载：目录获取、文件块下载

3．连接认证：获取认证信息、认证

4．文件分享 ：创建分享链接

4.4 系统的数据结构的 ER 图

本系统总共设计了八个实体：原文件数据实体、文件列表实体、存储服务器信息实体、数据存储块实体、文件传输信息实体、断点续传实体、私人文件实体、用户实体、分享链接实体、日志信息实体等。

1. 用户实体是用来保存用户认证信息的包括：用户 ID、用户密码共两个字段

图 4-1 用户列表结构

2. 原文件数据实体是用来保存记录原始文件的各种信息的包括：文件 SHA512、文件实际大小、文件创建时间、文件修改时间、文件属性、文件名称、文件总块数、备注等字段。
   如图 4-2 所示

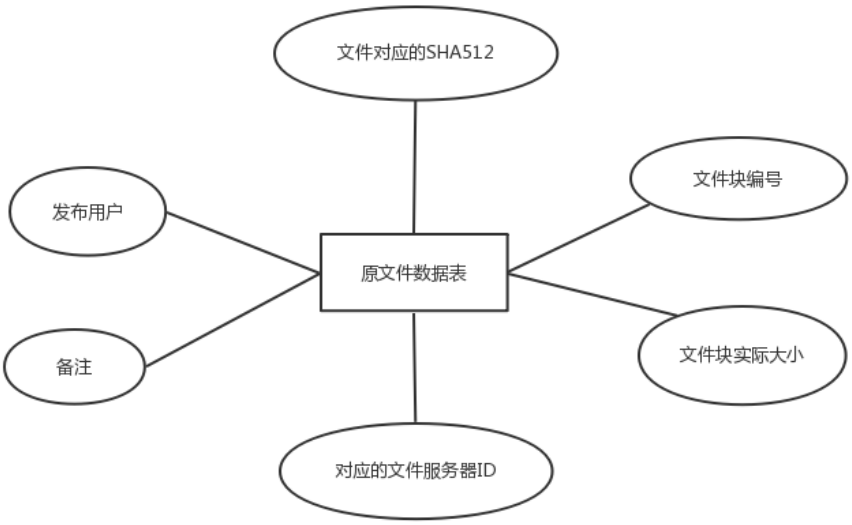


图 4-2 原文件数据实体列表结构

3. 文件列表实体是用来保存目录服务器的文件列表，包括：文件 SHA512、文件 MD5、文件创建时间、文件属性、文件修改时间、文件总块数、当前文件大小、文件名称等 8 个字段
   如图 4-3 所示

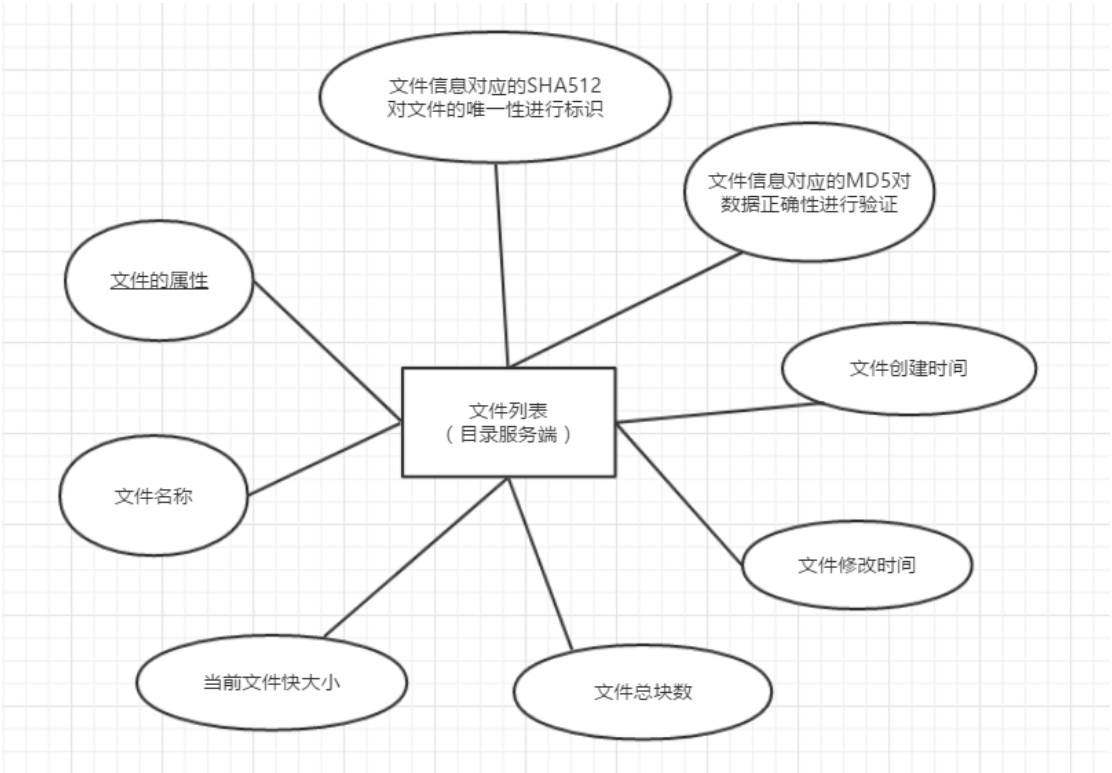


图 4-3 3.文件列表结构

4. 日志信息实体是用来保存各个端操作信息的实体，包括：操作时间、对象 SHA512、用户 ID、操作状态、操作信息等 5 个字段

如图 4-4 所示

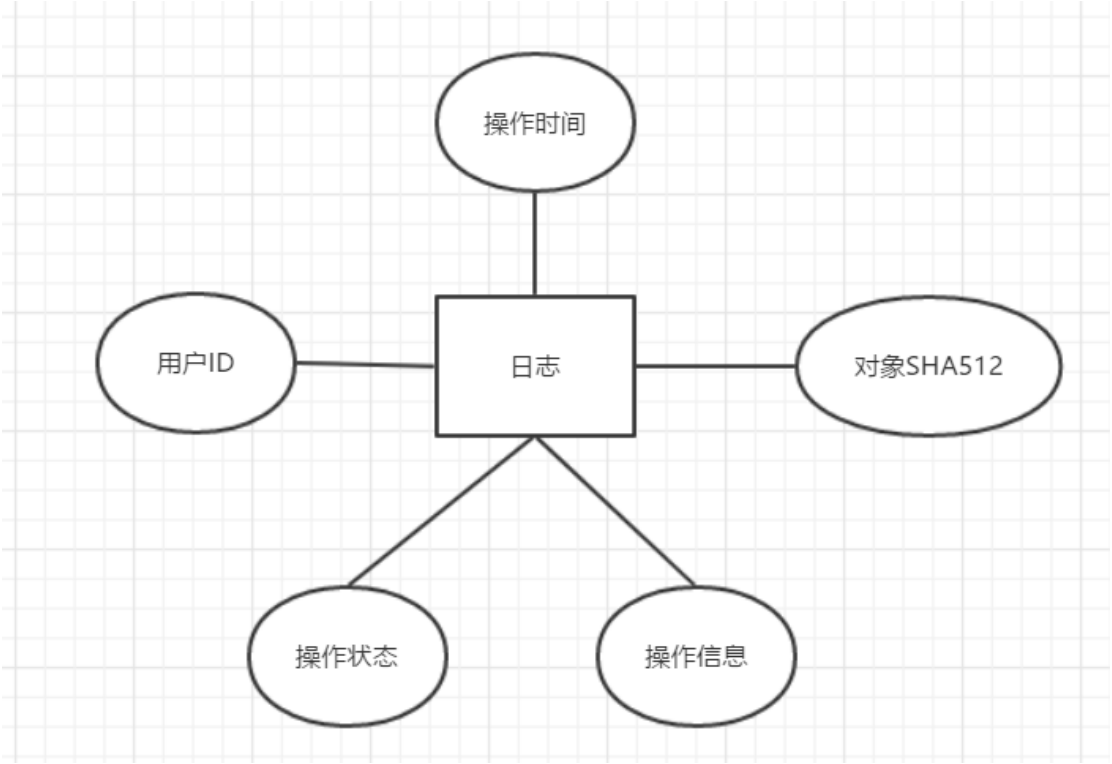


图 4-4 日志信息实体列表结构

5．存储服务器信息实体是用来保存各个存储服务器当前信息的包括：存储服务器 ID，存储服务器 IP、认证信息、认证信息生成的 MD5、总空间大小、剩余空间大小等 6 个字段。

如图 4-5 所示

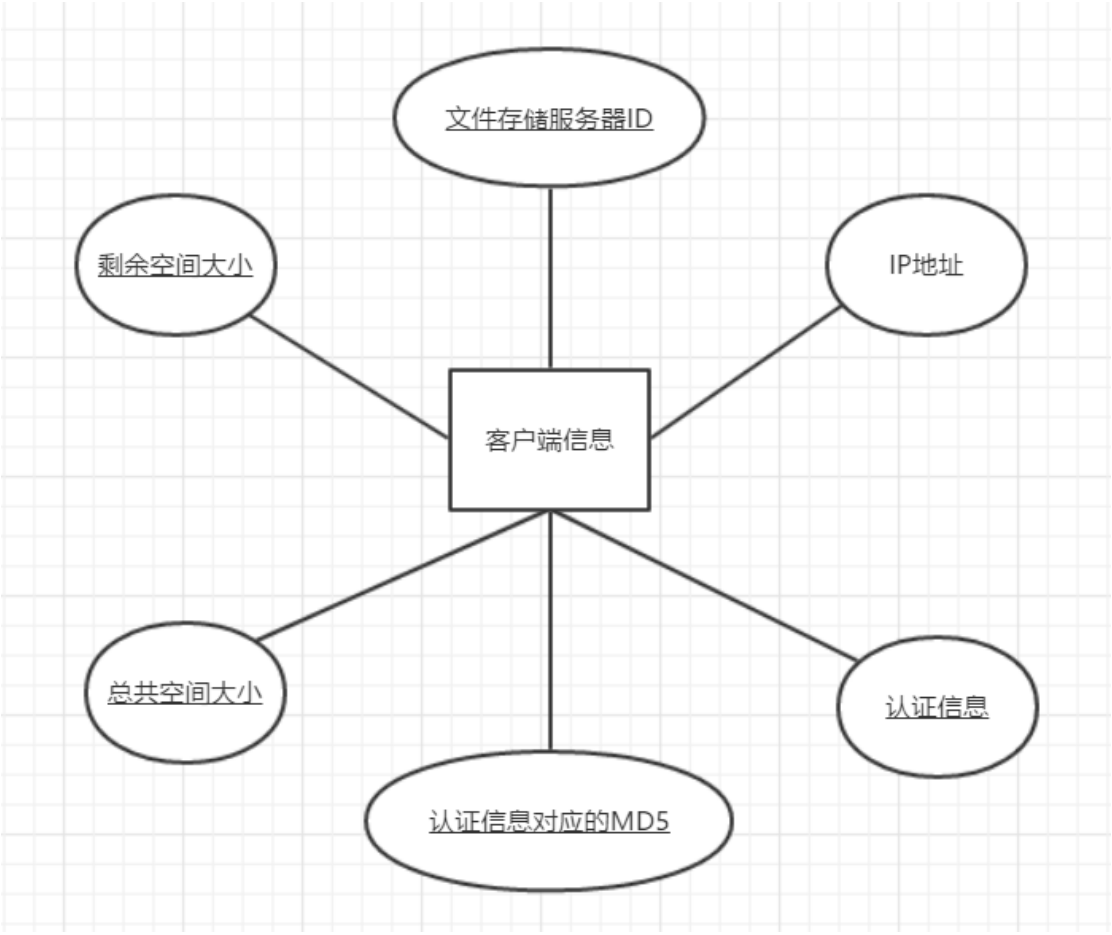


图 4-5 存储服务器信息实体列表结构

6．数据存储块（存储端）实体是用来保存每一块文件所保存的位置的，它包括：文件 SHA512、文件块编号、实际大小、存储文件名、存储位置绝对偏移量字段等 5 个字段
如图 4-6 所示

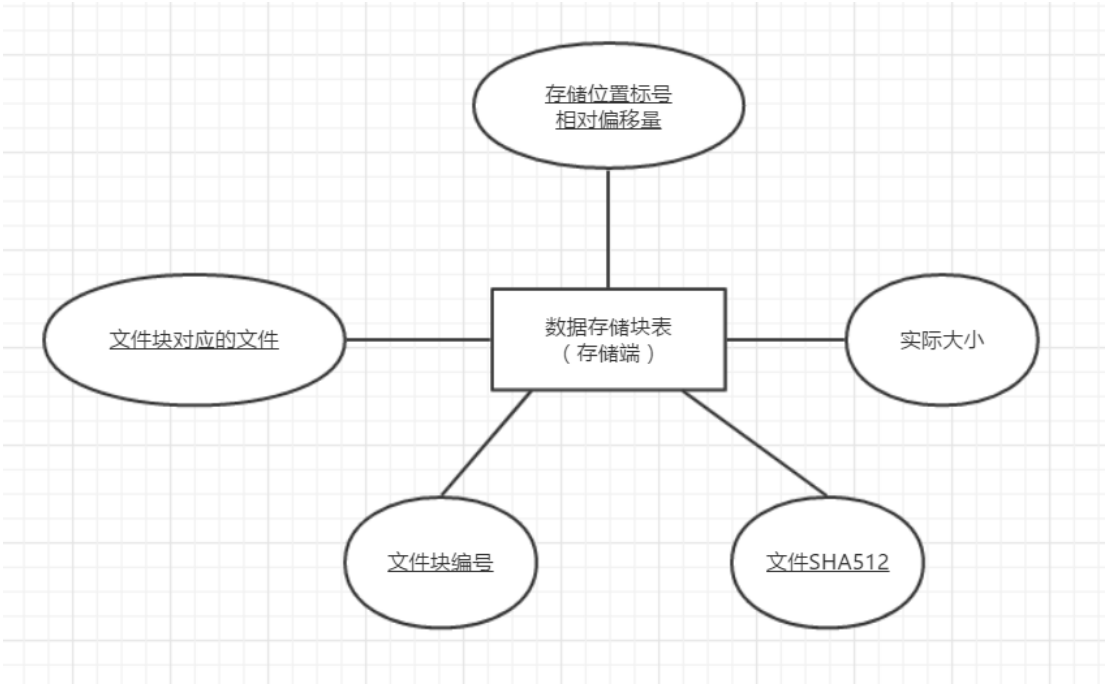


图 4-6 数据存储块（存储端）实体列表结构

7.数据存储块（目录服务器）实体是用来保存每一块文件所保存的位置的，它包括：文件 SHA512、文件块编号、实际大小、存储文件端 IP 等字段等 4 个字段
如图 4-7 所示

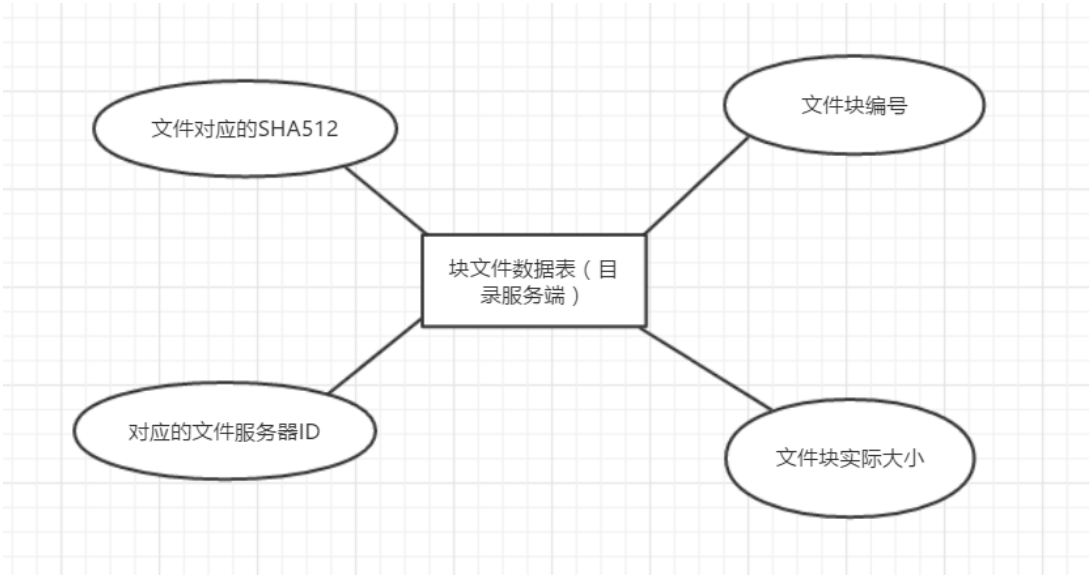


图 4-7 数据存储块（目录服务器）实体列表结构

8．文件传输信息实体是用来保存请求下载或上传文件块信息的，包括：文件 SHA512、文件块编号等 3 个字段
如图 4-8 所示

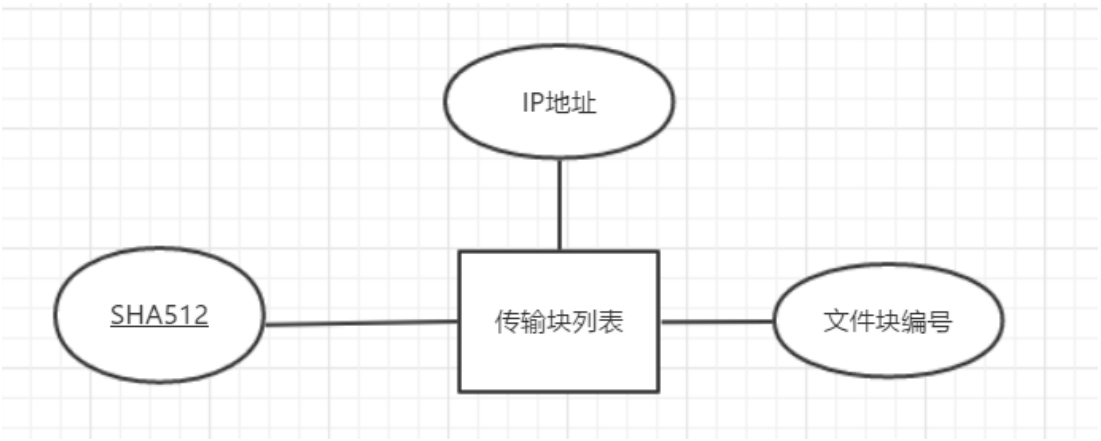


图 4-8 文件传输信息实体列表结构

9．断点续传实体是用来保存和记录文件下载的全部信息的，它包括：文件SHA512、下载时间、文件的 MD5、文件总大小、文件当前所保存的临时文件、文件已下载的块数，文件的位图存储标记等共 7 个字段
如图 4-9 所示

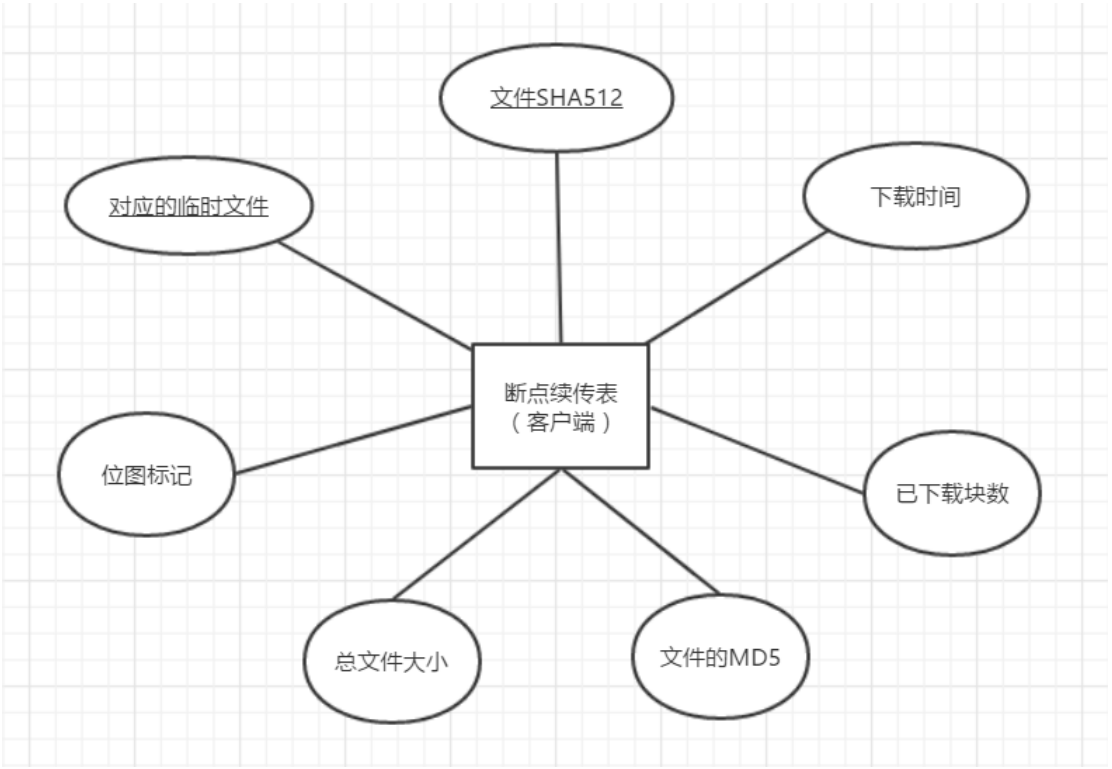


图 4-9 断点续传实体结构

10．私人文件实体是用来保存和记录个人所拥有的文件，包括：文件名称，文件的 SHA512、文件状态和文件名等 4 个字段
如图 4-10 所示

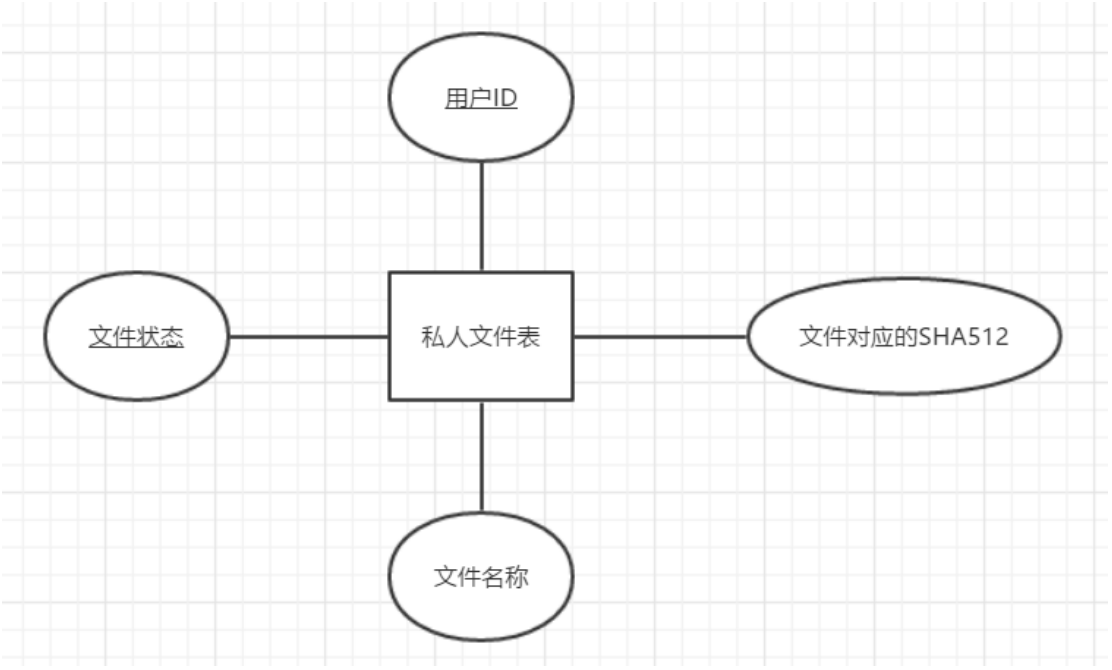


图 4-10 私人文件实体列表结构

11. 分享链接实体是用来保存分享文件的，包括：分享链接、文件 SHA512、和验证信息共 3 个字段。
如图 4-11 所示

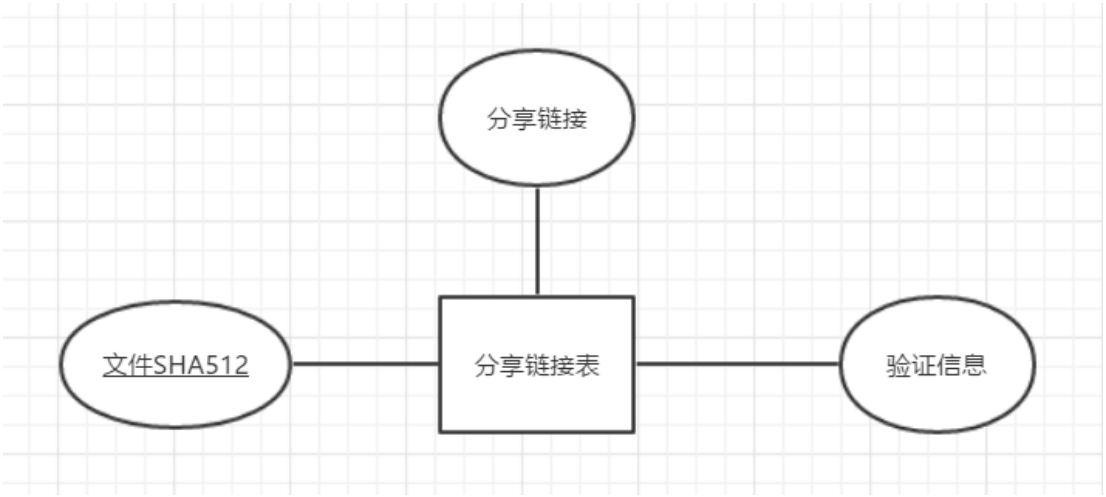


图 4-11 分享链接实体列表结构

12.信息校验实体用来保存不同端的校验信息，包括：文件存储端 ID、Key 对应的 MD5 等共 2 个字段
如图 4-12 所示

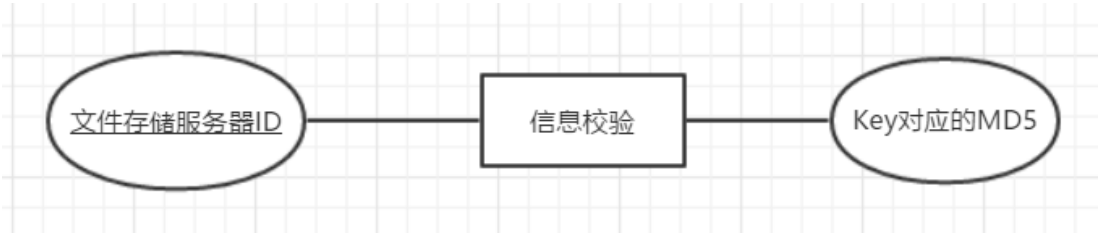


图 4-12 信息校验实体列表结构

13. 传输文件列表用于传输文件的 SHA512，在文件完成时校验，包括文件的 SHA512 以及文件的 MD5 等 3 个字段

如图 4-13 所示

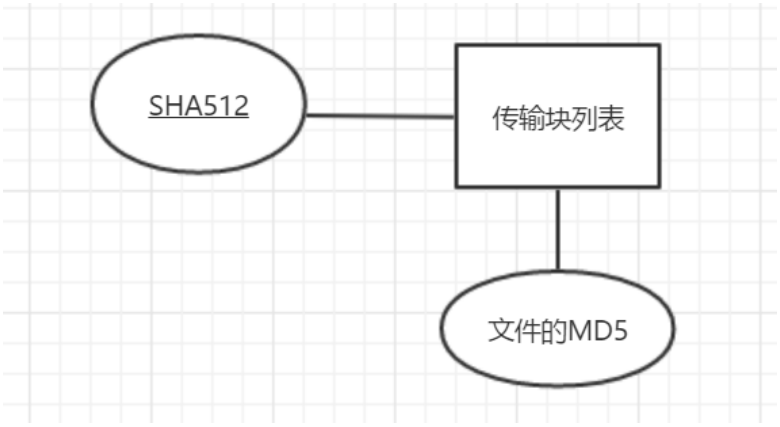


图 4-13 传输文件列表结构

4.5 数据字典

文件 = 文件 SHA512+文件信息+文件大小+文件存储偏移量+文件 MD5

分享链接 = 文件 SHA512+分享 URL+验证信息+用户

日志 = 操作时间+操作内容+操作对象+操作类型

文件下载 = 文件块存储列表+存储端 IP+文件 SHA512+文件 MD5+认证信息

文件分享 = 文件 SHA512+Url

认证信息 = 用户密码+用户 ID+密钥的 MD5

断点续传 = 文件 SHA512+文件位图下载状态+文件 MD5

私人文件 = 文件 SHA512+用户 ID

存储服务器 = 文件的 SHA512+文件块的 ID+文件偏移量+文件大小+文件名

4.6 实体对应的数据结构

1、用户表是保存用户基本信息，如表 4-1 所示

表 4-1 用户认证表

列名	类型	长度	是否主键	是否外键	描述
UserID	Uint32	--	是	否	用户 ID
UserPassword	String	32	否	否	用户密码

2、个人文件表是保存个人私有文件信息的表 如表 4-2 所示

表 4-2 个人文件表

列名	类型	长度	是否主键	是否外键	描述
UserID	Uint32	--	是	否	用户 ID
FileSHA51	String	128	是	否	文件的唯一标识
FileStatu	Uint32	--	否	否	文件状态
FileName	String	--	否	否	文件名称

3、文件操作日志表是保存和记录事件的变更信息的表，如表 4-3 所示

表 4-3 文件操作日志

列名	类型	长度	是否主键	是否外键	描述
LogDate	Uint64	--	是	否	日志时间
LogSha512	String	128	否	是	操作对象的唯一标
LogUserID	Uint32	--	否	是	事件发生者 ID
LogStatus	Uint32	--	否	否	事件类型
LogInform	String	--	否	否	日志操作详细信息

4、存储端列表是保存和记录事件的存储端信息的表，如表 4-4 所示

表 4-4 存储端列表

列名	类型	长度	是否主键	是否外键	描述
SavePointID	Uint32	--	是	否	存储端的 ID

SavePointIP	String	15	否	否	存储端 IP
PrivateKey	String	128	否	否	认证信息
Key2MD5	String	32	否	否	认证信息的 MD5
AllSize	Uint64	--	否	否	总空间大小
RemainSize	Uint32	--	否	否	剩余空间大小

5、块文件数据表（存储端）表是保存文件对应的数据块所存储服务端 ID 的完整信息表，如表 4-5 所示

表 4-5 块文件数据表（存储端）

列名	类型	长度	是否主键	是否外键	描述
FileSHA512	String	128	是	否	文件的唯一标识
FileBlock	Uint32	--	否	否	文件块的编号

6、文件分享表表是记录分享链接和文件的基本信息，如表 4-6 所示

列名	类型	长度	是否主键	是否外键	描述
SharedURL	String	30	是	否	分享链接
FileSHA51	String	128	否	否	文件唯一标识
FilePS	String	8	否	否	链接密码

表 4-6 文件分享表

7、文件信息表是记录文件的基本信息，如表 4-7 所示

表 4-7 文件信息表

列名	类型	长度	是 否 主	是否外键	描述
FileMD5	String	32	是	否	文件完整性验证
FileSHA512	String	128	否	否	文件唯一标识
CreateTime	Uint64	--	否	否	文件创建时间
ChangeTime	Uint64	--	否	否	文件修改时间
FileSize	Uint64	--	否	否	文件大小
BlockNum	String	128	否	否	文件总块数

FileName	Bits	--	否	否	文件名
FilePro	String	--	否	否	文件属性

8、断点续传表表是记录文件当前正在下载文件的所有状态的基本信息，如表 4-8 所示

表 4-8 断点续传表

列名	类型	长度	是 否 主	是否外键	描述
FileMD5	String	32	是	否	文件完整性验证
FileSHA512	String	128	否	否	文件唯一标识
DownTime	Uint64	--	否	否	下载时间
DownLoadBlock	Uint32	--	否	否	文件已下载块数
FileSize	Uint64	--	否	否	文件大小
FileTmpName	String	128	否	否	文件存储的临时文
BitMap	Bits	--	否	否	文件位图下载标记

9、文件块列表基本信息，如表 4-9 所示

表 4-9 文件块列表

列名	类型	长度	是否主键	是否外键	描述
FileSHA512	String	128	是	否	文件唯一标识
FileBlockNu	Uint32	--	是	是	文件块编号

10、数据存储块表（存储端）是用于记录当前文件快所存储的位置的基本信息 如表 4-10 所示

表 4-10 数据存储块表（存储端）

列名	类型	长度	是否主键	是否外键	描述
FileSHA512	String	128	是	否	文件唯一标识
FileBlockNum	Uint32	--	是	是	文件块编号
BlockFileSize	Uint32	--	否	否	文件块实际大小
FileBlockOffset	Uint64	--	否	否	文件块偏移量

FileSaveName	String	128	否	否	存储文件名称

11、传输文件列表是用于传输文件完成后验证文件正确性的基本数据结构，如表 4-11 所示

表 4-11 传输文件列表

列名	类型	长度	是否主键	是否外键	描述
FileSHA512	String	128	是	否	文件的唯一标示
FileMD5	String	32	否	否	文件的 MD5
IP	String	15	否	否	存储端的 IP

12、信息校验表是用于保存各种校验的基本信息 如表 4-12 所示

表 4-12 信息校验表

列名	类型	长度	是否主键	是否外键	描述
ServerID	Uint32	--	是	否	服务端 ID
Key2Md5	String	32	否	否	认证信息

13、文件分享对应的数据结构 如表 4-13 所示

表 4-13 分享链接表

列名	类型	长度	是否主键	是否外键	描述
FileSHA512	String	128	是	否	文件唯一标识
UserPasswo	String	32	否	否	用户密码
SharedUrl	String	--	否	否	分享链接

第五章 系统实现

5.1 系统整体的实现

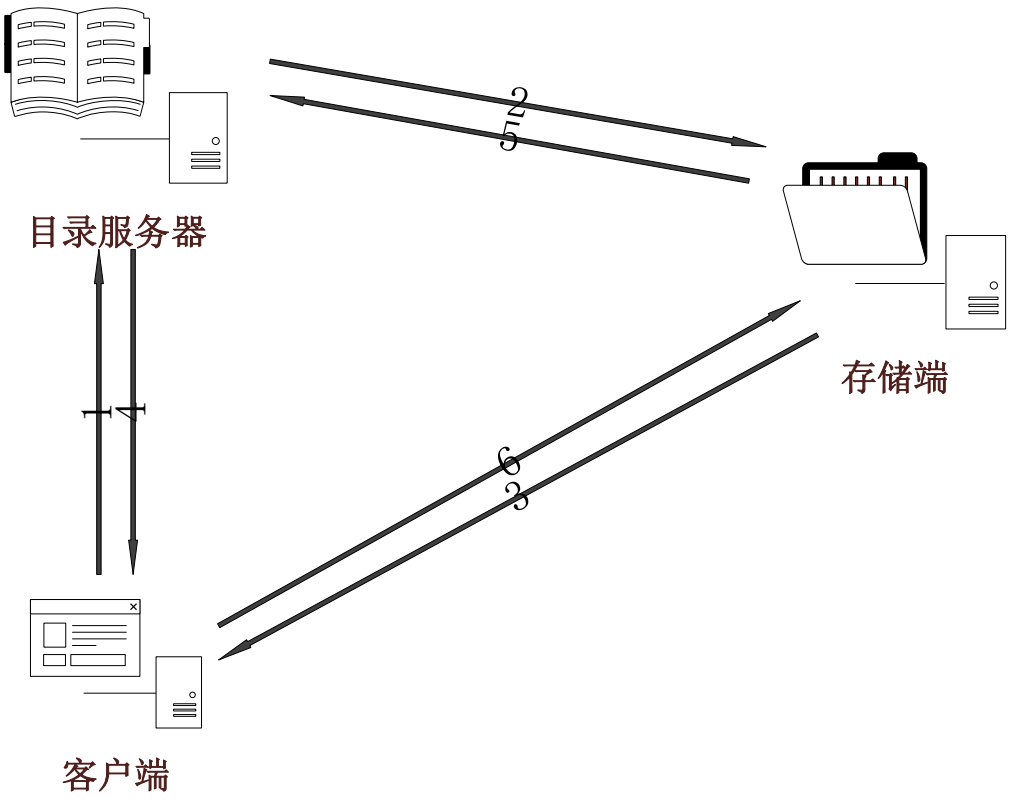


图 5-1 系统数据流向图

5.2 数据流向

本上传与下载系统，与传统的下载是不同的传统的文件下载方式是单个文件下载，虽然都有块的验证，但是必须按照顺序上传或下载。

而本系统上传时虽然也是采用顺序上传，但是也是有所不同：就是上传时，当对应的链接传输完成时，就立刻获取对应要传输的数据块。由于网络带宽的影响这些存储到不同服务器的文件块可以是乱序的，但是整个文件是完整的。

由于采用断点续传的下载方式，下载时不用担心数据块的下载顺序，客户端会下载存储服务端不忙的文件块，如果服务器响应超时，可以请求重传没有下载的数据块，因为下载时会记录当前文件的下载状态。

由于系统采用的时 SHA512 的消息摘要作为唯一序列，若用户没有对应的文件则，其碰撞的概率很低，所也可以不采用权限管理，以简化逻辑结构。

5.2.1 文件的上传：

如图 5-1 ：1→4→6（1）→3→6→5→2

1： 客户端先计算出文件的 SHA512 后，将本客户端的用户名和密码与文件的 SHA512 进行序列化后发送给目录服务器，并根据附带参数，判断请求类型。

2：目录服务器接收到消息后，对消息进行解析先验正用户的用户名和密码，若密码错误返回验证消息。若为正确则，查询文件是否已经存在，若已经存在，则返回文件已进行秒传。若服务器没有该文件则，返回可以进行上传的文件存储服务器中在线的 IP 地址以及对应的密钥的 MD5 值。

6：（1）若文件不存在时，则将文件的信息发送到目录服务器。

（6）当需要进行文件上传时，对可进行存储操作的服务器，先进行验证，将用户的 ID 以及目录服务器返回的密钥所生成的 MD5 值发送给存储服务器（Port：12345），进行验证。

3：若验证成功，则由存储服务器发起文件传输链接请求。

6：然后客户端进行数据的传输，对应的文件块附带 MD5 验证信息，文件块传输策略： 当不同的存储端建立起请求时，对文件进行竞争读取模式，当一个文件传输进程开始或者是刚刚的文件块已经传输完成时，对文件进行加锁保证每次只有一个进程读取文件（防止多个进程同时读取相同的文件块（同时读取文件块时会产生数据的冗余）），将读取的数据进行计算其 MD5，组成（文件的 SHA512+MD5+Data 块）的数据结构进行传输，若传输进程发现已经没有文件块可以进行传输则关闭对应的传输链接，当所有传输进程关闭时则表示文件的数据传输已完成。并在数据完成时将文件信息发送给目录服务器，请求服务器验证文件是否传输完整。

5：在文件传输过程中，对文件进行完整性验证，若数据块正确则可以将数据块存到对应的文件中，并在块存储列表中添加记录（文件的 SHA512、原始文件中对应的偏移量、存储文件时对应的偏移量、存储的文件名）。

当文件客户端的文件上传链接关闭时，存储端将文件的块号、对应的文件 SHA512

并附带存储服务器的 ID 发送给目录服务器。

2：目录服务器接收到数据时保存为临时数据，当收到客户端的文件完整性验证请求时对接收到的数据进行验证，若完整则将他们保存到对应的序列化文件中。

否则返回失败。

5.2.2 文件下载

如图 5-1 ：1→4→3→6

1：客户端先与目录服务器建立验证链接，发送用户 ID 和密码以及对应的文件 SHA512，并根据附带参数，判断请求类型。

4：服务端收到客户端文件下载请求时，先验证客户端信息的有效性，若有效则查询文件是否存储在，若文件存在则，查询文件的对应存储位置验证总数据块后，若文件块书完整则将查询结果发送给客户端，若文件不完整、不存在或者是用户的信息错误时返回对应的消息错误码。

6：若返回错误信息则进行提示，若返回的是文件对应的存储信息，与对应的文件存储服务器建立连接请求，并发送对应的验证密钥的 MD5。

3：若验证信息正确则，与客户端建立文件传输链接，发送所请求下载的文件块，当文件传输完成时，关闭传输链接并发送文件以传输完成的结束信息。

客户端接受文件快的过程中，验证文件的 MD5，若文件正确是写入到文件的对应位置，并将接受到的文件块的映射为文件的 MAP 映射，便于文件的断点续传。当文件传输完成时，验证整个数据的完整性。

5.2.3 文件分享

如图 5-1 ：1→4→6→3

1： 客户端先计算出文件的 SHA512 后，将本客户端的用户名和密码与文件的 SHA512 进行序列化后发送给目录服务器，并根据附带参数，判断请求类型。

2：若文件存在，服务端则生成一个不重复的 URL 发送给客户端。

若文件不存在，则提示文件不存在需要先进行上传。

1：当文件上传完成时，向目录服务器发送请求，再次生成不重复的 URL。

2：当数据完整性验证完成时，并生成一个不重复的 URL 发送给客户端。

生成 URL：将文件的 SHA512 与用户 ID 进行连接之后计算其 MD5 值作为文件的 URL。

5.2.4 文件删除

如图 5-1 ：1→4→3→6

文件的删除是通过配置文件，由目录服务器定期执行，目的是删除一些长久不用的文件，以减少磁盘占用和提高文件查询效率。

1：根据目录服务器的文件保存策略，对目录进行筛选，对要筛选的信息进行，文件块的查询、查询完成时连接对应的文件存储进行链接验证。

2：存储服务器验证链接信息，对数据进行筛选，删除对应的数据块和碎片空间的整合。结束时给文件服务器发送完成消息。

3：当所有文件服务器都发送完才能消息时，对目录进行目录数据的清除。

5.2.5 分享链接的删除

如图 5-1 ：1→4

分享链接参数格式：8 位+128 位

若为 URL 格式为：网站链接并附带分享链接参数

前 8 位时与用户相关的验证信息，后 128 位是文件的 sha512 消息摘要

1：先进行将本客户端的用户名和密码并附带上对应的分享链接，序列化后后发送给目录服务器，并根据附带参数，判断请求类型。

2：目录服务器先进行数据的反序列化，再进行参数的解析，确定请求了类型，并进行验证。若要进行分享链接的删除，则先检查 URL 的合法性（是否符合要求），在查找对应的 URL 是否存在，若存在则删除，并返回对应的处理消息。

5.2.6 URL 的下载

如图 5-1 ：1→4→6→3

将“URL+验证消息” 发送给发送给目录服务器，若正确服务器返回对应的文件块存储列表信息，若不存在则返回相应的错误信息。

若返回消息为文件块列表存储信息，则对文件列表进行遍历，依次相对应的存储端发送文件块下载请求，等待建立连接。

文件存储端，接受到请求时验证请求和客户端的有效性，若有效则提供下载服务。

5.2.6 日志服务

当任何段进行任何操作时，都将操作的详细信息，输出到日志文件，便于为题的查找。

第六章 总结和展望

6.1 总结

几个月的时间匆匆而逝。从最开始的迷茫，到和导师讨论构想和相应技术的实现，慢慢的从构想到实现的整个过程，都离不开老师的指导。此次毕业设计我的收获是颇丰的，这也离不开老师的指导和监督。

从开始到实现的过程中，既有兴奋也不乏一些困难，使得我们五味杂陈，但是最终的构想到现实的转变是我受益匪浅。首先我一开始是再用百度云盘时有想法的，对其文件存储的方式比较好奇，很高兴的是，我从百度中得到了答案，分布式文件系统。我明白一个企业的应用的成功，比不知是依靠一些逻辑代码就可以了，而是需要大量的技术支撑的，虽然不能达到企业的效果。但是作为学生的我有一种探索的精神，所以选择了一个有挑战性的任务，就是实现一个简单的分布式文件存储与分享软件。

对技术进行筛选后我决定使用 Google 的 ProtoBuf 作为文件目录的存储结构，使用 C++开发语言，并使用 Boost.asio 网络框架，来完成自己的构想，其中最难的一点就是实体对象关系的处理和要学习使用这些框架。因为一开始不太了解网络程序的编写所以开始使用 Asio 网络框架时一头雾水，但是 boost 库提供了，不错的 Demo 开发者去使用和了解 Asio 框架。当遇到 ProtoBuf 时，由于是中文文档比较少，只能慢慢的啃官方文档，所以学习使用 ProtoBuf 用了将近三周的时间，到目前为止只学会使用最近基本的用法，构课程设计使用的了。

在实现软件的过程中 BUG 不断的出现被解决，最终实现了最初构想发的大部分逻辑功能。当软件完成时，可以说与最初的设想相比是：构想是西施，却写出来个东施。这就是软件专业的现状 2:8 定律“做完一个软件的逻辑结构需要 20%的时间，但是要完善一个软件却要 80%的时间”。

总之到最后软件的顺利完成后离不开老师的指导和帮助，使得我也从中学到了很多技术可以在以后的工作中用到。

6.2 不足及未来工作展望

虽然软件的基本功能已经实现但是，要想成为一个安全运行的软件还有很多路要走。首先就是，要保证功能的稳定，因为用户用软件是不希望出现 bug 是导致文件的丢失，如果这种情况出现较多的话，会使用户对软件失去信任。其次是本软件的实现是基于 Windows 平台的相对于 Linux 系统来说，效率稍显低，并且没有完成跨平台部署，虽然可以进行跨平台的文件传输，但是具体的客户端没有实现到各个平台，无法全面推广，拓展性欠缺。还有一根问题就是当系统长时间运行而不进行清理时会出现硬盘空间不足，可能会导致无法系统崩溃。当系统中有文件服务器宕机时，会造成大量的数据不完整，因此在未来需要对该功能进行解决：对文件存储服务器采用 RAID 1[19]存储架构，并实现服务器的跨平台部署，解决当用户量比较大时，对认证机制进行系统分离，以减少目录服务器的压力提高服务效率。

分布式文件存储与分享系统作为一种共享式的文件系统，不但节约存储空间，还可以减少长距离的带宽占用，提高整个网络贷款的利用率，以及降低文件传输的时延，提高文件下载的并发度，充分利用剩余带宽。而且若所有的文件存储系统采用分布式，并提高一定量的冗余度，可以对用户提供已删除实现文件的还原的服务，以提高产业价值。

其次是分布式文件存储系统可以进行的微服务[20]，使得在软件中集成分布式文件分享利用文件的冗余度，查找附近要下载的文件并下载，或者是为附近的文件提供文件下载服务等。

致谢

时光冉冉，岁月如梭。在我的论文即将完成之际，也预示着将要与生活四年的大学校园、同学、老师们告别，在此由衷的感谢四年的时光里所有老师与同学给予的专业上、生活上以及精神上的帮助。

首先要感谢的是鲍春波老师，是他是我对 C/C++ 产生了兴趣，并建议我们多了解新的技术，这也是我从大一到大四一直学习在 C/C++的原因。

其次要感谢那些为boost 库等一些列优秀的作品的人致敬，表示衷心的感谢。

同时还要特别的感谢郭方导师在我做毕业设计的这大半年时间里给予的关怀。从课题的方向初定，到系统的完整实现和论文的撰写。郭方导师经常给予我正确的方向指导。从研究方向的探讨，到具体数据结构和功能的设计与实现，再到论文的撰写，导师都给予了诸多指正和建议。在于老师的交流过程中使我受益匪浅。整个设计从初定到完成，一路走来导师始终亦师亦友，同时也感谢老师多我的包容。由于自己年轻气盛，不经常与老师交流自己的完成进度，在此向郭老师致歉。再此表示由衷的感谢！

此外，还要感谢开源社区。如果没有诸多优秀的框架（例如 Boost，STL，ProtoBuf 等库）的开源项目作为参考和基石。我们之所以能有所现在的成果（很快的完成毕业设计）是因为我们站在巨人的肩膀！饮水思源，我必将不断学习，力争开发出优秀的项目回馈开源社区。

参考文献

[1] 胡文波, 徐造林. 分布式存储方案的设计与研究[J]. 计算机技术与发展, 2010,20(4):65-68.

[2] 刘琦. 浅谈P2P网络文件传输[J]. 科教文汇旬刊, 2016(3):182-185.

[3] 马连超. CDN-P2P架构下的文件下载策略的研究与实现[D]. 北京邮电大学, 2014.

[4] 葛瑞格尔. C++高级编程(第3版)[J]. 电脑编程技巧与维护, 2015(15).

[5] Duffy D J, Kienitz J. 13. The Boost Library: An Introduction[M]. John Wiley & Sons, Inc., 2015.

[6] Radchuk D. Boost.Asio C++ network programming cookbook[J]. 2016.

[7] 李兰. 分享经济:未来十年的风口[J]. 决策, 2015(5):62-63.

[8] 兰翔. 基于Nginx的负载均衡技术的研究与改进[D]. 华南理工大学, 2012.

[9] Williams A. C++ concurrency in action[J]. Planned Products, 2012.

[10] 罗剑锋. C++11/14高级编程——Boost程序库探秘(第3版)[M]. 清华大学出版社, 2016.

[11] 软件开发技术联盟. VISUAL C++开发实例大全,提高卷[M]. 清华大学出版社, 2016.

[12] 波卢欣卢涛. 深入实践Boost : Boost程序库开发的94个秘笈 : Boost C++ application development cookbook[M]. 机械工业出版社, 2014.

[13] 高静, 段会川. JSON数据传输效率研究[J]. 计算机工程与设计, 2011,32(7):2267-2270.

[14] 威廉逊. XML技术大全[M]. 机械工业出版社, 2002.

[15] Yu S, Lei W, Aoki K. Preimage Attacks on 41Step SHA256 and 46Step SHA512[J]. Iacr Cryptology Eprint Archive, 2012,2009.

[16] 辛运帏, 廖大春, 卢桂章. 单向散列函数的原理、实现和在密码学中的应用[J]. 计算机应用研究, 2002,19(2):25-27.

[17] 王红霞, 陆塞群. 基于HMAC—SHA1算法的消息认证机制[J]. 山西师范大学学报(自然科学版), 2005,19(1):30-33.

[18] 陈庆章, 赵小敏. TCP/IP网络原理与技术[M]. 高等教育出版社, 2006.

[19] 王红. 磁盘阵列的RAID技术[J]. 信息技术与信息化, 2005(3):104-106.

[20] 张英. 微服务:开创图书馆服务的“蓝海”[J]. 图书馆建设, 2011(7):51-53.