学校代码： 10246
学　　号： 15212010038

硕 士 学 位 论 文

（专业学位）

云存储中间件关键技术优化设计与实现

Design and Implementation of Backup-Cloud Oriented Optimized Middleware

院　　系：　软件学院
专业学位类别（领域）：　软件工程
姓　　名：　刘武
指 导 教 师：　韩伟力　教授
完 成 日 期：　2017 年 月 日

指导小组成员名单

韩伟力 教授

目录

摘要

近年来，云存储服务因其海量的存储空间、便携的存取方式、可靠的安全特性，逐渐取代传统的数据存储模式，成为人们首选的数据备份方式。国家“核高基”科技重大专项[1]为顺应这一发展趋势，设立了云存储与云备份子课题，构建了一款自主研发的云存储服务平台。该系统的中间件[2]作为服务器与客户端之间的交互枢纽，实现了数据传输、数据压缩、数据加密以及安全会话等基本功能。然而，持续升温的云存储行业在近几年面临着诸多的挑战，衍生出了如安全性、稳定性、可扩展性等问题，用户对云存储平台提出了更高的要求。在这样的环境下，各大云存储供应商纷纷对云存储的关键技术进行革新，而现有的云备份中间件系统仅实现了基础的数据传输、存储功能，这已无法适应当前错综复杂的云环境，因此亟需对其关键技术进行优化，以提升自身的竞争力。

本论文主要通过对当前云存储行业的分析，研究云储存服务中的关键技术，同时基于前期的成果，对现有云存储中间件进行优化，以适应云存储行业的发展趋势。本文拟在 Linux 平台上，通过物理隔离和数据加密的方式，保证缓存在中间件中的用户信息和会话记录的安全性；同时，借助大文件分块传输和断点续传技术，解决大文件上传过程中遇到网络中断或服务器死机等故障而引发的数据重传问题；利用本地存储和服务器提供的高级接口，减少数据在上传下载过程中产生的冗余数据，并提供快速上传的功能，以提高数据传输性能；此外，优化并完善平台框架和 API 支持机制，解决不同应用和云存储服务之间适配的问题，以支持更多客户端类型使用云备份与云存储服务。

关键词：

Abstract

~~With the rapid development of Internet technology~~, the network has penetrated into every corner of our lives. It needs an efficient way to store the massive resulting data. However, the traditional data storage methods has long been unable to meet the needs of users because of its limited capacity, the use of inconvenient, easy to lose data and other limitations, so cloud storage technology came into being. In just a few years, there are dozens of cloud storage products launch on the market. From abroad iCloud, Google Drive, Dropbox to the domestic Baidu cloud, Ali Cloud, Tencent Cloud, many companies who are optimistic about the emerging markets want to be in this crowd. Continued warming of the cloud storage industry in recent years is facing fierce competition, In order to occupy more market share, the challenge has also spurred the new cloud storage platforms to introduce new technologies to solve the issues derived from cloud storage like security, stability and network problems.

This paper mainly studies the key technology in cloud storage service, and optimizes a self-developed cloud storage middleware. Based on previous results, in the Linux platform, exploiting the local storage and computing power to reduce data upload and download I / O-intensive services, to improve service performance. Through the form of encryption to ensure that user information and data security. At the same time, this paper uses the technology of block upload and breakpoint resume to avoid data retransmission caused by network problem or server problem. In addition, the platform framework and API support mechanisms are optimized and refined to address the adaptation issues between different applications and multiple cloud storage services to support more desktop OS components using cloud backup and cloud storage services.

Keywords：

第一章 引言

1.1 背景介绍

随着互联网技术的飞速发展，网络数据呈现出井喷式的增长，传统的数据存储方式存在太多的局限性，已经无法满足人们的要求。在这样的背景下，云存储服务[3]，逐渐成为主流的数据存储方式。因其海量的存储空间、便携的存取方式、可靠的安全特性，越来越受大众的青睐，大型的机构如政府、医院、企业等也逐步将数据转移到云端。

为适应数据存储的新趋势，国家“核高基”科技重大专项设立了云存储与云备份子课题，拟构建支持云存储服务的底层框架，研制一款高效的云存储服务平台。至目前为止，经过充分的市场调研和系统的需求分析，设计出了一套完整的体系结构，实现了数据传输、数据压缩、数据加密以及安全会话等基本功能。系统以中间件作为服务器与客户端之间通信的桥梁，有效地分离了客户端云存储工具与基础设施之间的绑定，提高了系统的稳定性，并为系统的跨平台性提供了良好的基础[4]。

然而，大范围普及的云存储服务在近几年迎来了巨大的挑战，除了提供云端数据存储这项基本的功能之外，人们对云存储服务平台的安全性、容灾性、稳定性以及可扩展性投入了更多的关注。在传统的数据存储方式中，数据保存在本地，对用户来说是可管控的，而将数据存储在云端则可能产生如个人数据泄露、商业信息遭到窃取、科研成果外泄等问题，这可能对用户产生不可逆的后果。此外，云存储服务以网络作为基础设施，而网络又时常受限于硬件、软件和地域等多方面的因素，一旦出现网络波动或中断，将会影响数据在网络层的传输，尤其对较大的文件传输来说，遇到网络问题只能进行重传，这将极大地降低用户的体验度。同时，随着云端数据不断地积累，产生了大量重复冗余的数据。一方面这些冗余的数据会占用服务器大量的存储空间，另一方面用户存储在服务器的文件只是一个映射的链接，如果重复上传服务器中已存在的文件，将会占用大量的网络带宽。更进一步，介于移动互联网和智能手机的热度持续走高，人们的娱乐、办公环境逐渐趋向于平台化，针对单一应用的云存储服务已不适应当前平台多元化的格局，实现云端数据在多平台的共享，成为了云存储行业的共识。

目前，国内外成熟的云存储服务供应商为应对这一系列新的挑战，纷纷推出新的技术或是对原有技术进行革新，以最大化程度的满足用户需求。我们选取了四个国内外较为知名的云存储服务供应商进行了调研，国内的

百度网盘、360 云盘，国外的 Google Drive、Dropbox，分别对他们的技术特性进行了全面的分析。表 1.1 罗列出了详细的信息。

表 1.1 国内外知名云存储服务平台技术特性分析

	加密算法	数据去冗余	大文件上传	断点续传	平台数
百度网盘	128 位 SSL	全局	20G / 4M	支持	6 种
360 云盘	128 位 SSL	全局	5G / 4M	支持	5 种
Google Drive	128 位 AES SSL/TLS	无	5TB / 字节流	支持	5 种
Dropbox	256 位 SSL/TLS	单一用户	不限 / 8M	支持	7 种

数据加密是一种强有力的安全措施来保护个人信息不被窃取或篡改，根据调研结果，所有的云存储服务供应商都采用了类似的方式对数据在传输过程中及存储在云端时两种状态进行了加密。在传输过程中，百度网盘和 360 云盘使用的是 128 位的 SSL[5]加密技术，Google Drive 和 Dropbox 分别采用了的 128 位/256 位的 AES SSL/TLS[6] 加密技术。数据在云端时，百度网盘和 360 云盘未强制对数据进行加密，而是由用户在上传之前决定是否要将数据以加密的形式存储。反观 Google Drive 和 Dropbox，它们选择了对上传至云端数据进行强制加密，Dropbox 使用的是 256 位的 AES[7]加密方式，而 Google Drive 采取了规模相对较小的 128 位 AES 加密技术。此外，考虑到网络的不稳定因素，各大供应商都对大文件上传技术进行了优化，以避免因网络原因而导致的大文件重传问题。百度云盘支持单个文件上传的大小上限为 20G，360 云盘为 5G，Google Drive 为 5TB，而 Dropbox 不限制单个文件的大小。在处理大文件上传的方案上，它们都将文件以固定大小的文件流形式，通过设定偏移量来对大文件进行分块传输，这样做的好处是遇到网络故障时，只需记录已上传数据流的偏移量，重传时可以根据偏移量进行文件续传而无需从头开始。针对大量数据冗余的问题，国内外的云存储服务平台选择了不一样的策略。减少数据冗余除了能为服务器节省大量的存储空间外，还能为用户提供一种新的功能，这种功能称之为“秒传技术”。所谓的“秒传技术”指的是当服务器存在当前用户待上传的文件时（通过文件的 MD5 进行校验[7]），不会真实地上传该文件，而是会将这个文件的链接拷贝到用户的网盘中，这样能有效地解决冗余数据重复上传的问题。百度网盘和 360 云盘都支持全局的重复文件校验，为用户提供了“秒传”这项功能。Dropbox 采取的是一种相对折中的方式，它不会在全局文件系统中校验该文件是否存在，只针对单一用户进行去重处理，这样能在很大程度上地减少由去重而导致的数据安全问题，如侧信道攻击[8] 。Google Drive 对待“秒传技术”极其谨慎，尽管这样做能够大幅度地提升上传的效率，但出于安全性方面的考虑，Google Drive 未在云盘中使

用该技术。同时，这些成熟的云存储供应商都会针对主流的操作系统和平台，提供不同类型的客户端，这样使得用户可以跨平台地使用云存储服务。常见的平台有 Windows、Mac、Android、iPhone、Web，以上四个云存储供应商都提供了这些平台的客户端，其中百度云盘还支持 Windows Phone，Dropbox 是唯一支持 Linux 和 BlackBerry 智能手机的平台。

由此可见，各大供应商为迎合市场的需求，纷纷对云存储的关键技术做了重要的革新。而由国家“核高基”科技重大专项自主研制的云存储服务平台目前尚不具备以上技术特征，因此，亟需对原有系统中的关键技术进行优化，以保证产品的竞争力。

1.2 研究内容及意义

云存储关键技术作为产品的核心竞争力，已成为了各大供应商的研究的重点。但是，由国家“核高基”科技重大专项自主研制的云存储服务平台还停留在提供基础的云端数据存储服务上，为顺应云存储行业的发展趋势，本文主要通过对现有的中间件系统进行全面分析，同时借鉴业内成熟产品的优势，从以下几个方面进行优化。

1. 持久会话管理：云存储中间件在客户端和服务器之间工作，多个客户端的命令通过中间件提交给服务器。用户的命令会以会话的形式保存在中间件，当中间件系统出现异常时（如网络中断、中间件死机），会在下次重启中间件时自动恢复执行未完成的会话。此外，对会话信息需要进行了物理隔离，并且以加密的形式存储在数据库中，以提高用户信息的安全性。
2. 大文件上传：利用服务器提供的高级功能，实现了大文件分块上传功能，在受到网络状况受到影响的情况下，若文件上传中断，会自动启用断点续传功能，以避免数据重传。
3. 本地缓存：利用现代计算机普遍较大的存储空间以及高性能的计算能力，实现了本地缓存管理，进一步减少了系统 I/O 操作，提高了中间件性能。同时，利用服务器提供的个人存储在云端的数据信息，实现了针对单一用户的“秒传”功能。
4. 多应用适配：原有的中间件系统只支持单一的 CLI 客户端，为实现对不同客户端请求的适配，支持更多的客户端类型，对系统的跨平台性做了优化。此外，开发了一款基于 Chrome 浏览器的插件，用户可以利用插件使用云存储服务，并提供了 Java 平台的软件开发包（SDK）[9]，Java 开发者可以利用该开发包提供的应用接口 API 来集成我们的云存储服务。

1.3 论文组织结构

本文将分七个章节对论文进行阐述。第一章介绍本文研究内容的背景以及相关的研究工作。第二章介绍本文在研究和实现过程中使用到的核心技术，包括中间件技术、加密算法、Java SDK、WebExtension、大文件上传/下载技术、断线续传技术以及本地缓存技术。第三章将分析原有云备份中间件的不足，指出本文需研究和实现的内容。第四章详细阐述云备份中间件系统的整体设计，包括优化后的系统框架、类图设计、流程设计以及接口设计。第五章介绍云备份中间件实现的细节。第六章将对云备份中间件优化后的结果进行评估，以实际的测试结果作为评估的标准。第七章对全文进行总结，并对未来进行展望。

第二章 核心技术

2.1 AES 加密算法

2.2 SQLite

2.2.1 SQLite 简介

SQLite 是 D.Richard Hipp 用 C 语言编写的开源嵌入式数据库引擎。它可以在所有的主流操作系统上运行，并支持大多数的 SQL92 标准，且源代码不受版权限制。SQLite 由四大部分组成：SQL 编译器、内核、后端以及附件，其整体结构如图 2-1 所示：

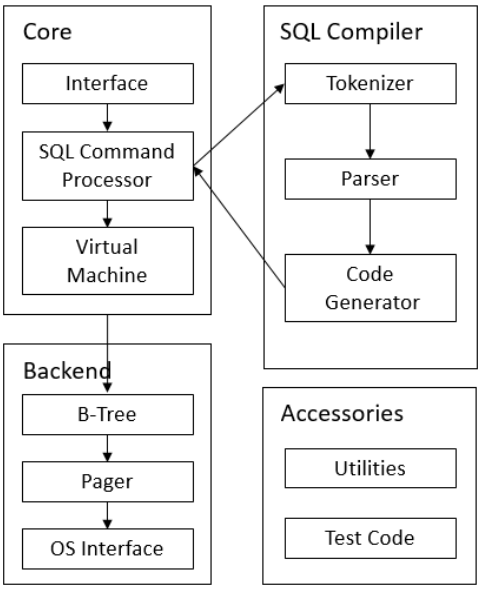


图 2-1 SQLite 系统结构图

2.2.2 SQLite 功能特性

在存储容量方面，SQLite 虽然是一款轻量级的数据库，但是最高可支持 2TB 大小的数据存储，并且每个数据库都是以单个文件的形式存在的，数据都是以 B-Tree 的数据结构形式存储在磁盘上。

在事务处理方面，SQLite 支持多个进程可以同时读取同一个数据的，但只有一个进程可以执行写操作。通过数据库级上的独占性和共享锁机制，使得某个进程或线程想要往数据库写入数据之前，必须获得独占锁，而其他进程则无法执行写操作。

在数据类型方面，SQLite 支持 NULL、INTEGER、REAL、TEXT 和

BLOB 数据类型，分别代表空值、整型值、浮点值、字符串文本、二进制对象。同时，SQLite 采用动态数据类型的机制，当某个值插入到数据库时，首先会对它的类型进行检验，如果该类型与关联的列不匹配，SQLite 则会尝试将其自动转换为对应列的类型，如果不能转换，则将该值将作为本身的类型。

2.3 断点续传

2.3.1 断点续传简介

断点续传技术指的是在下载或上传时，将文件以固定的大小，划分为若干部分，或以固定长度的字节流的形式，对每部分都单独采用一个线程进行上传或下载，如遇到网络故障，则可将此时的断点记录到数据库中，下次可以根据断点所记录的已经上传或下载的部分开始继续上传下载未完成的部分，而没有必要从头开始上传下载。通过此项技术，可以帮助用户可以节省大量的时间，提升上传下载的效率。由于网络的不稳定因素，断点续传技术在云存储行业应用十分广泛。

2.3.2 断点续传实现过程

一般在使用断点续传时，需要用到 HTTP 请求中的 Range 和 Content-Range 实体头，Range 用于在请求报文中指定起始字节的位置，Content-Range 用于在响应报文中，指定整个实体中的一部分的插入位置，他也指示了整个实体的长度。它们的一般格式如下：

Range : (unit=first byte pos)-[last byte pos]

Content-Range : bytes (unit first byte pos) - [last byte pos]/[entity length]

请求报文如下图所示：

```
GET /test.docx HTTP/1.1
Connection: close
Host: 10.131.1.63
Range: bytes=0-1024
```

响应报文如下图所示：

```
HTTP/1.1 200 OK
Content-Length: 256823
Content-Type: text/html
Content-Range: bytes 0-1024/256823
```

在断点续传过程中，首先需要将上传的文件拆分为固定大小的部分，随后逐一发送 PUT 请求到服务器。该请求中需要包含 Range 字段，表示当前请求上传数据的字节范围。一旦发生网络中断，下次连接时首先发送一个 GET 请求到服务器，在请求的 header 中放一个 Content-Range 字段，通过服务器返回的信息，从 Content-Range 字段中获取已上传的字节数，之后再发送剩余字节的上传请求。

2.4 WebSocket

2.4.1 WebSocket 简介

WebSocket 主要用于 Web 浏览器和服务器之间进行双向数据传输。其协议基于 TCP 协议实现，包含初始的握手过程，以及后续的多次数据帧双向传输过程。其目的是在 Web 应用和后台服务器进行双向通信时，可以使服务器避免打开多个 HTTP 连接进行工作来节约资源，提高了工作效率和资源利用率。

2.4.2 WebSocket 协议分析

WebSocket 是一种类似 TCP/IP 的 socket 技术，其协议为应用层协议，定义在 TCP/IP 协议栈之上。WebSocket 连接服务器的 URI 以"ws"或者"wss"开头。ws 开头的默认 TCP 端口为 80，wss 开头的默认端口为 443。具体来说，共分为握手阶段和数据通信两个阶段。

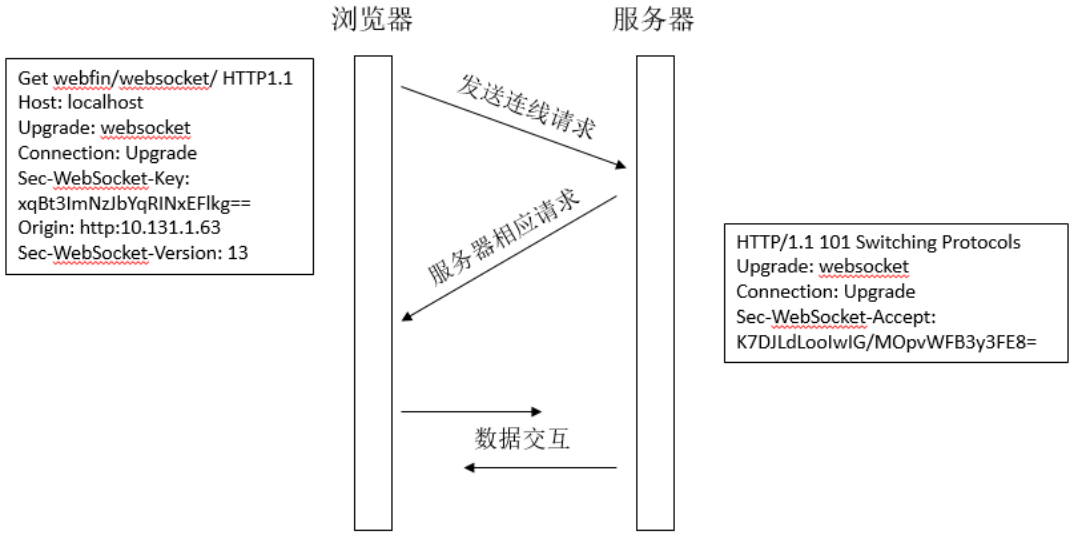


图 2-2 握手阶段示意图

在握手阶段，户端和服务器建立 TCP 连接之后，客户端发送握手请求，

随后服务器发送握手响应，如图 2-2 所示。

在数据通信阶段，数据被组织为依次序的一串数据帧(data frame)，然后进行传送。帧的类型分为两类：数据帧(data frame)和控制帧(Control frame)。数据帧可以携带文本数据或者二进制数据；控制帧包含关闭帧和 Ping/Pong 帧。如要关闭连接，只需任何一端发送关闭数据帧给对方即可，关闭连接以状态码进行标识。

2.5 LRU 替换策略

2.6 Java SDK

2.7 WebExtension

第三章 云存储中间件系统优化分析

本章主要通过对现有的中间件系统进行全面分析，详细阐述其已实现的功能，并在此基础上结合对国内外云存储服务平台的调研结果，指出当前中间件系统的不足之处，由此分析出需要优化的具体内容。

3.1 现有中间件系统分析

现有的云备份中间件系统是基于 Linux 系统的，它提供了客户端与存储服务器之间的数据交互，有效地分离了客户端和服务器之间的绑定。同时，云备份中间件也提供给云备份客户端与云备份服务器，身份认证服务器以及密钥管理服务器之间稳定可靠的数据传输[4]。

3.1.1 功能分析

目前，云存储中间件在功能上主要包括四个模块：数据传输模块、数据加密模块、数据压缩模块、安全会话管理模块，各模块的详细说明如下：

1. 数据传输模块：实现了服务器与客户端之间的数据交互，主要提供数据的上传与下载功能，这也是云存储中间件系统的基本功能。
2. 数据加密模块：为了保证中间件系统能够抵御各种恶意网络攻击，中间件系统会将数据以加密的形式进行网络传输。使用的是高级加密标准（Advanced Encryption Standard，AES）对称加密算法，该算法是当前最流行的对称加密算法。
3. 数据压缩模块：支持对大文件的压缩功能。当上传大文件时，用户可以通过 COMPRESS 参数指定对文件进行压缩，以减少数据传输的大小，降低所需的网络带宽，提高传输的效率。此外，整个压缩过程是透明，用户通过压缩上传的文件，下载后是解压的格式。
4. 安全会话模块：中间件作为服务器与客户端之间的桥梁，需要保存用户的登录信息，以方便客户端与服务器之间建立持续的连接。原有的中间件系统采取一种安全的方式以确保用户的信息的安全性。

3.1.2 缺陷分析

从功能上看，现有的中间件系统实现数据传输、加密、压缩和安全会话等功能，能够有效地将用户的数据传输到云端，并且具备一定的安全机制。

然而，根据目前行业的发展趋势，现有的中间件系统存在很多的不足之处，具体来说有以下几个方面。

首先，现有的中间件系统没有考虑到数据传输模块是以网络作为基础的，默认网络在数据传输过程中是绝对稳定的。但是，在实际生活中，我们的网络状况常常受限于硬件、软件以及地域等多方面的因素，网络波动或网络中断是常见的问题。由于我们的网络带宽是有限的，因此上传文件，尤其是较大的文件，通常需要花费一定的时间，在此期间存在因网络故障而导致文件传输失败的问题。在现有的中间件系统里，用户只能重传文件，倘若多次遇到网络故障，则可能导致无休止的重传，这对用户来说是极其不友好的体验。

其次，为保证用户数据的安全性，现有的中间件系统在文件传输过程中，系统采用了 128 位的 SSL 加密技术，在文件的存储形式上，让用户主动选择是否加密，使用的是 AES 加密算法，这样的加密策略与主流的云存储服务平台使用的方式基本一致。然而中间件作为服务器与客户端之间的桥梁，需要保存一定的用户信息，以作为交互的保障。用户的登录信息、会话信息等敏感数据在现有中间系统中是以缓存的形式保存的，这些信息以明文的形式保存在 SQLite 数据库中，同时也未设置数据库的访问权限。因此，用户的私密信息很容易被窃取。此外，由于会话等信息是以缓存的形式存储在中间件系统中的，若中间件系统出现故障，可有可能造成用户会话信息的丢失。

再者，现有的中间件系统未对冗余的数据进行处理。当用户上传一个已有的文件到服务器时，服务器不会对文件进行去重检验，而是会重命名该文件，并以文件的原始形式保存。当冗余的文件过多时，会占用服务器大量的存储空间。去重技术虽然有一定的安全隐患，但是我们可以借鉴 Dropbox 针对单一用户去重策略，这样既能缓解服务器的存储压力，又能为在一定程度上保证数据的安全。

最后，现有的中间件系统只支持 Linux 平台下的 CLI（Command Line Interface）客户端。考虑到主流的云存储供应商都提供了多平台数据共享的服务，我们需要对现有的中间件系统进行重构，解决来自多平台的不同客户端与云存储服务器的适配问题，使其具备可扩展性以支撑更多的客户端类型。

3.2 中间件系统优化需求分析

通过上文对现有中间件系统的缺陷分析，并结合业内同类产品的调研结果， 本文将从持久会话管理、大文件上传、本地缓存、多应用适配四个方面进

行优化，具体分析见下文。

3.2.1 持久会话管理

云备份中间件在客户端和服务器之间工作，多个客户端的命令通过中间件提交给服务器，用户在访问存储服务器上的资源时，需要提供身份和授权信息，因此中间件也需要保存用户的相关私密信息。由于存在多用户同时将命令提交给中间件的可能，因此用户的命令不会马上被执行，而是以缓存的形式存储在数据库中，只有被处理请求的进程轮询到时才会生效。在原有的系统中，用户的会话信息是以明文的形式保存的，因此需要设计一种行之有效的策略来保证用户私密信息的安全。具体来说，可以从以下几个方面进行优化：

1. 不同用户的数据，按照会话进行逻辑隔离；一个用户只能访问自己所属会话的数据。
2. 明文形式的秘密信息，只能作为会话数据在内存中存在。
3. 用户的持久私密信息以加密的形式存储，通过会话管理模块只能获取到加密形式的数据。

此外，客户端可以批量提交任务给中间件，即使用户注销后，中间件仍要在后台继续执行任务。更进一步，在中间件系统重启后，中间件也需要自动执行未完成的任务。因此，中间件需要持久化保存多个客户端的请求，以可靠地执行客户端提交的任务。同时，我们需要设计一套基于会话信息的接口，便于用户查询已提交的请求，如果请求没有执行，用户可以撤销该任务。

3.2.2 大文件上传

在很多情况下，客户端和云备份服务器分布在不同的地域，网络传输速度受各种条件制约。在现有的中间件系统中，一个文件如果传输不成功，下一次只能重传，这浪费了大量的带宽和时间。事实上，服务器支持大文件分块传输的功能，也就是说客户端可以传输指定大小的文件数据。因此，中间件可以根据网络数据传输的质量，确定文件分块传输的大小，并在每次上传时记录文件以上传的块数，即已上传文件的偏移量。如果遇到网络故障，可以根据偏移量实现断点续传的功能。这样做能够很大程度上地减少因网络因素导致的数据重传问题，提高大文件的传输成功率。

图 3-1 详细描述了大文件上传的过程。

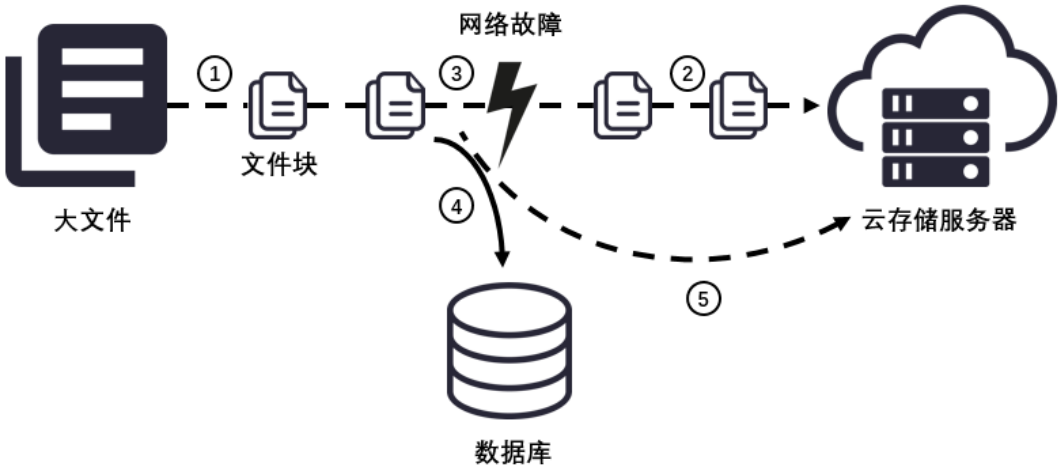


图 3-1 大文件上传及断点续传过程分析图

1. 将大文件分割成固定大小的文件块；
2. 依次将文件块上传至服务器；
3. 遇到网络故障；
4. 将当前已传送文件块的信息记录到数据库；
5. 查询数据库中的断点信息，实现文件续传，并调用合并文件接口组合文件。

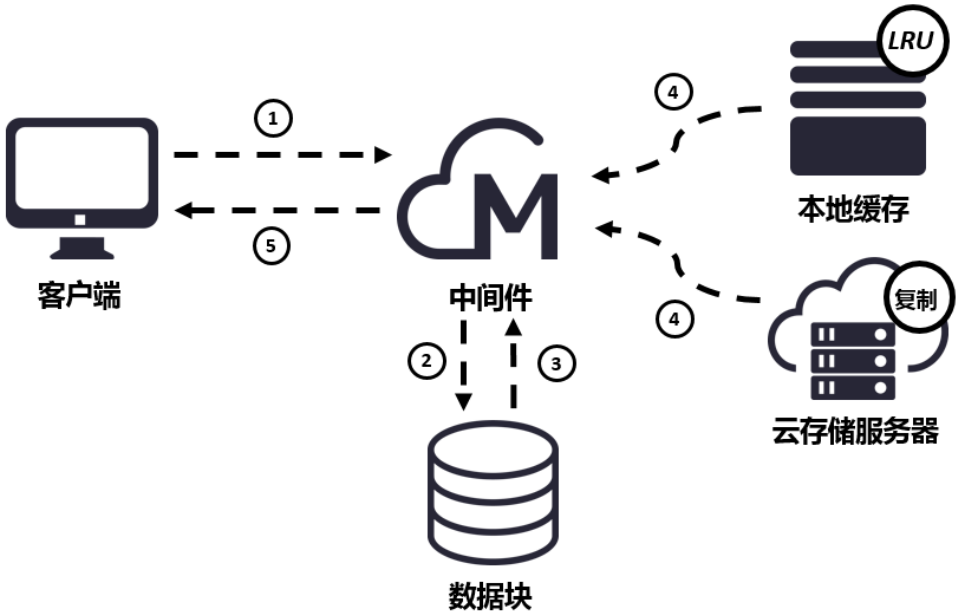
3.2.3 本地缓存

云存储是存储和网络 I/O 密集的服务[10]，大量的 I/O 会引起机器性能下降，也可能会干扰其它的应用。现代计算机通常有很大的本地存储空间，并且计算能力普遍过剩。因此，可以通过利用本地存储和计算能力，如优化本地缓存的利用，进一步减少 I/O，提高服务性能。

现有的中间件系统在接受到用户文件上传和下载时，首先将文件存储在临时区域，该区域只能被中间件访问。由于客户端远程访问文件的功能，所以用户可能会反复访问同一个文件，如果每次都从远程下载该文件，则是对网络带宽资源的浪费。因此，我们需要高效利用临时存储区域，以降低下载文件时所需要的 I/O 请求。考虑到本地缓存的具有一定的容量限制，当缓存容量达到上限时，可以采用 LRU 替换策略[11]以保证缓存文件的时效性。

此外， 同一个用户备份数据时会有相同的文件。在上传文件时，同一用户的相同文件是可以复用的，我们可以利用服务器提供的复制文件接口，直接在服务器操作，以减少上传文件所需的网络带宽，从而实现“秒传”功能。

图 3-2 详细描述了本地缓存的过程。



1. 用户发送上传/下载请求至中间件；
2. 中间件查询服务器/本地缓存中是否存在用户待上传/下载的文件；
3. 返回数据库查询结果；
4. 上传文件上，若服务器存在该文件，则将服务器中的文件复制到用户的上传路径；下载文件时，若本地缓存存在该文件，则将缓存中的文件复制到用户的下载路径；若不存在缓存文件，则执行正常的上传/下载流程。
5. 将请求结果返回给用户。

3.2.4 多应用适配

作为中间件，其主要的目标之一是完成不同客户端和服务器之间的适配，以使平台具备一定的可扩展性。由于平台和操作系统之间的差异性，加上不同客户端在设计时考虑的情形和目标不一样，客户端和服务端通常会定义不同的接口，导致服务器和客户端不匹配。

为了减少客户端和服务端之间的差异，可以通过中间件来对双方的调用方式进行转换，适配分为两种情况：

1. 提供一个标准的接口，客户端调用中间件给出的接口；
2. 在中间件中引入新的适配模块，以较小的代价将不同的客户端和服务端整合在一起。

第一种情况适合于规范新开发的客户端，从用户使用的角度来统一定义一个访问接口。第二种情况，实际上是第一种情况的强化形式，需要中间件来设计更灵活的机制来支持更一般化的适配。在这里，我们选择了第二种方式作为中间件

多应用适配的方案。

为了适配客户端发出的请求（输出形式）和服务端的接收请求（输入形式）之间的差异，中间件需要对来自不同客户端的请求进行适配，以服务器的标准接口规范作为基准，转化为统一的、服务器可识别的形式。同时，由于目前版本的中间件只有 CLI 一个客户端，因此我们需要开发其他类型的客户端来支撑多应用适配，在这里，我们选择了开发基于 Chrome 浏览器的插件和 Java SDK 开发工具。

第四章 云存储中间件系统优化设计

4.1 中间件优化系统框架设计

4.2 类图设计

4.3 关键流程设计

4.3.1 系统总流程设计

4.3.2 持久会话管理流程设计

4.3.3 大文件上传流程设计

4.3.4 本地缓存流程设计

4.3.5 多应用适配流程设计

4.4 系统接口设计

第五章 云存储中间件系统优化实现

第六章 云存储中间件系统优化结果分析

第七章 总结和展望

7.1 总结

7.2 展望

参考文献

致　　谢

复旦大学
学位论文独创性声明

本人郑重声明：所呈交的学位论文，是本人在导师的指导下，独立进行研究工作所取得的成果。论文中除特别标注的内容外，不包含任何其他个人或机构已经发表或撰写过的研究成果。对本研究做出重要贡献的个人和集体，均已在论文中作了明确的声明并表示了谢意。本声明的法律结果由本人承担。

作者签名：刘武　　日期：2017.07.02

复旦大学
学位论文使用授权声明

本人完全了解复旦大学有关收藏和利用博士、硕士学位论文的规定，即：学校有权收藏、使用并向国家有关部门或机构送交论文的印刷本和电子版本；允许论文被查阅和借阅；学校可以公布论文的全部或部分内容，可以采用影印、缩印或其它复制手段保存论文。涉密学位论文在解密后遵守此规定。

作者签名：刘武　导师签名：韩伟力　日期：2017.07.02