广东工业大学硕士学位论文

（工学硕士）

基于 Hadoop 与 SSM 的大数据云存储平台设计与实现

袁斯煌

二〇一八年五月

分类号：　　学校代号：11845
UDC：　　密级：　　学号：2111404063

广东工业大学硕士学位论文

（工学硕士）

基于 Hadoop 与 SSM 的大数据云存储平台设计与实现

袁斯烺

指导教师姓名、职称 ：　谢胜利 教授
学科(专业)或领域名称：　控制科学与工程
学 生 所 属 学 院：　自动化学院
论 文 答 辩 日 期：　2018 年 5 月

A Dissertation Submitted to Guangdong University of Technology

For the Degree of Master

（Master of Engineering Science）

Design and implementation of big data cloud storage platform based on Hadoop and SSM

Candidate: Yuan Silang

Supervisor: Prof.Xie Shengli

May 2018

School of Automation

Guangdong University of Technology

Guangzhou，Guangdong，P.R.China，510006

摘要

云计算在近年来逐渐成为国内外关注的热点。当云计算系统中的运算与处理的核心是海量数据的存储时，云计算就衍变成为一个云存储。伴随着云计算的高速发展，云存储也成为当下最为热门的研究领域。云存储作为当前的新兴服务，它把用户的数据都存储到云端的服务器上，用户只需要通过网络登录到云存储服务系统上，就能够随时随地地查看、添加自己的文件，并且再也不用担心数据的丢失了。

Hadoop 是阿帕奇开发的一个开源分布式计算平台。在分布式计算和数据存储方面，Hadoop 表现出优异的性能，并引起了国内外知名 IT 公司的高度关注，各大公司和科研机构纷纷投入大量人力物力进行研究，使得 Hadoop 在云计算和云存储中的应用越来越广泛。Hadoop 包括 HDFS 分布式文件系统。HDFS 拥有强大的数据存储能力，特别适合在作为云存储集群中使用。但 HDFS 在设计上存在一些缺陷和性能上的不足。因此想要大规模推广 HDFS 的使用，还必须对其进行改进。本论文主要研究基于 HDFS 的云存储模型，并针对基于 HDFS 所建立的大数据云存储平台在云数据存储、安全性方面以及并发性能方面的不足对其进行改进。最后使用 HDFS 与当前流行的 SSM 服务端后台开发框架搭建一个高可用的大数据云存储平台。

本论文主要分为四大部分，分别是客户端、传输层、请求处理系统、云存储集群。客户端是用户直接操作大数据云存储平台的工具；传输层提供安全加密的方式传输文件，请求处理系统是后台系统，向上接收用户请求，向下操作 HDFS；云存储集群中文件直接存放的物理介质，提供海量数据存储，其与请求处理系统对接。本论文关于大数据云存储平台的主要工作和特点如下：

一，云存储集群采用 Hadoop 搭建，同时增加备份元数据节点，组成联邦结构。HDFS 的元数据存储在 namenode 节点上，而 HDFS 一般只有单一的 namenode 节点，所以整个 HDFS 的性能、存储容量以及可靠性都受到单一 namenode 的限制。甚至，如果 namenode 宕机，则整个 HDFS 分布式文件系统将无法正常运行。所以我们需要对 HDFS 的 namenode 进行改进，增加一个 backup_namenode 备份节点，以提高 HDFS 的可靠性。

二，客户端增加一层文件系统过滤驱动加密机制。存储在 HDFS 上的文件都是先按一

定的算法分割成多个指定大小的文件块后再存储的，换言之，HDFS 都是采用明文的方式对文件进行存储。所以若 HDFS 被黑客攻击，导致用户数据发生泄漏，那后果将不堪设想。所以我们需要在原来 HDFS 的基础上，增加一层加密机制，对存储在 HDFS 上的文件进行加密，以提高 HDFS 的安全性。

三，在客户端与请求处理系统使用 Netty 框架的非阻塞 IO 方式传输。与传统的云存储系统不同的是，我们这次设计的平台将使用 Netty 框架所支持的非阻塞 IO 方式传输文件，比阻塞式 IO 的性能更好，同时还节约了系统的线程等资源。云存储系统需要考虑的其中一个问题是整个系统的并发性，当用户请求同发数达到一定时就会严重约束云存储系统的性能与市场发展。本论文将使用非阻塞 IO 以增强大数据云存储平台的并发性。

四，传输层使用 HTTPS 安全网络传输协议传输文件。HPPTS 安全协议在当前 IT 行业最为流行和安全性较高的网络 传输协议。因为，在第二部分的基于上，本论文还将进一步采用 HTTPS 协议以加强我们的大数据云存储平台的安全。

五，采用 SSM+Netty+Shiro 框架搭建请求处理系统。请求处理系统使用 SSM 以达到快速搭建的目的，同时还减少了大部分烦琐的问题。这样即可达到处理客户端请求的要求。结合采用 Shiro 框架进行用户权限认证。本论文研究的大数据云存储平台具有用户分层的权限级别。对不同的用户提供不同程度的文件安全级别。另外结合第二部分提及的非阻塞 IO，实现请求处理系统的高并发性能要求。

本论文在最后进行大量实验验证，将采用原始的 HDFS 搭建的云存储系统与改进后的方案进行对比，实验结果证明，本文提出的改进方案具有更好的效果，能够发送 HDFS 的性能。使用改进后的 Hadoop 搭建的云存储集群，开发 Web 应用程序，通过 B/S 模块模拟云存储平台，实现云存储的相关功能。

关键词：云存储；Hadoop；分布式文件系统；SSM；非阻塞 IO；Shiro

ABSTARCT

Cloud computing has gradually become the focus of attention at home and abroad in recent years. When the core of computation and processing in cloud computing system is massive data storage, cloud computing is evolving into a cloud storage. With the rapid development of cloud computing, cloud storage has become the most popular research field at present. Cloud storage, as a current new service, stores the data of the user to the cloud server. Users only need to log in to the cloud storage service system through the network, and can check and add their own files anytime and anywhere, and no longer have to worry about the loss of data.

Hadoop is an open source distributed computing platform developed by Apache. In the aspects of distributed computing and data storage, Hadoop shows excellent performance, and has aroused the high attention of well-known IT companies at home and abroad. Large companies and scientific research institutions have invested a lot of human and material resources in research, making the application of Hadoop more and more widely in cloud computing and cloud storage. Hadoop includes a HDFS distributed file system. HDFS has strong data storage capability, and is especially suitable for use as cloud storage cluster. But HDFS has some defects in design and performance deficiencies. Therefore, we must improve it if we want to popularize HDFS in a large scale. This paper mainly studies the cloud storage model based on HDFS, and improves the large data cloud storage platform based on HDFS in cloud data storage, security and concurrency performance. Finally, we build a high availability large data cloud storage platform based on HDFS and the current popular SSM server side development framework.

This paper is mainly divided into four parts, which are client, transport layer, request processing system and cloud storage cluster. The client is a tool for the user to operate the large data cloud storage platform directly; the transport layer provides a secure encrypted way to transmit the file. The request processing system is a back-end system, receives the request from the user and operates down the HDFS; the file stores the physical media directly in the cloud storage cluster and provides the mass data storage, which is processed with the request processing. System docking. The main works and characteristics of this paper about big data cloud storage platform are as follows:

First, cloud storage cluster is built by Hadoop, at the same time, backup metadata nodes are

added to form a federal structure. The metadata of HDFS is stored on namenode nodes, and HDFS is generally only a single namenode node, so the performance, storage capacity, and reliability of the whole HDFS are limited by a single namenode. Even if namenode is down, the HDFS distributed file system will not function properly. So we need to improve the namenode of HDFS, add a backup_namenode backup node to improve the reliability of HDFS.

Two, the client adds a file system filter driver encryption mechanism. The files stored on the HDFS are stored according to a certain algorithm and then stored in a number of specified size files. In other words, HDFS is used to store the files in a plaintext. So if HDFS is attacked by hackers, causing user data to leak, the consequences will be unthinkable. So we need to add a layer of encryption mechanism on the basis of the original HDFS to encrypt the files stored on HDFS, so as to improve the security of HDFS.

The three is the non blocking IO transmission using the Netty framework on the client side and the request processing system. Unlike traditional cloud storage systems, the platform we designed this time will use the non blocking IO mode supported by the Netty framework to transmit files, better than blocking IO, and save the resources of the system threads. One of the problems that the cloud storage system needs to consider is the concurrency of the whole system. When the user requests the same number, it will seriously restrict the performance and market development of the cloud storage system. This paper will use non blocking IO to enhance concurrency of big data cloud storage platform.

Four, transport layer transfers files using HTTPS secure network transmission protocol. HPPTS security protocol is the most popular and highly secure network transport protocol in the IT industry. Because, based on the second part, we will further use the HTTPS protocol to enhance the security of our big data cloud storage platform.

Five, use SSM+Netty+Shiro framework to build request processing system. The request processing system uses SSM to achieve the purpose of rapid construction, and at the same time, it also reduces most of the tedious problems. In this way, we can achieve the requirement of processing client requests. The Shiro framework is used to authenticated user rights. The large data cloud storage platform researched in this paper has user level privileges. It provides different users with different levels of file security. In addition, combined with the non blocking IO mentioned in the second part, we achieve high concurrency requirement of request processing system.

In the last part of this paper, a lot of experimental verification is carried out, and the original HDFS cloud storage system is compared with the improved scheme. The experimental results show that the improved scheme proposed in this paper has a better effect and can transmit the performance of HDFS. The cloud storage cluster built by improved Hadoop is used to develop Web application program, and cloud storage platform is simulated through B/S module to realize related functions of cloud storage.

Keywords: cloud storage；Hadoop；distributed file system；SSM；non blocking IO；Shiro

目录

摘要 I

ABSTARCT III

目录 VI

CONTENTS IX

第一章 绪论 1

1.1 研究背景及意义 1

1.1.1 研究背景 1

1.1.2 研究意义 2

1.2 国内外研究现状和发展均势 4

1.2.1 云存储现状 4

1.2.2 Hadoop 现状 4

1.3 论文研究内容及组织结构 6

1.3.1 论文主要研究内容 6

1.3.2 论文组织结构 7

第二章 关键性技术简介 8

2.1 Hadoop 的 HDFS 分布式文件系统 8

2.2 SpringMVC+Spring+Mybits 后台框架 9

2.3 文件系统过滤驱动原理 11

2.4 HTTPS 网络安全协议 12

2.5 Netty 13

2.6 Shiro 16

第三章 大数据云存储平台分析与设计 18

3.1 平台要求分析 18

3.1.2 性能要求 19

3.2 平台总体设计 20

3.2.1 平台总体架构设计 20

3.2.2 模块划分原则 22

3.2.3 平台功能模块总体设计 22
3.3 客户端模块设计 24
3.3.1 注册登录模块设计 26
3.3.2 文件系统过滤驱动设计 27
3.4 传输模块设计 28
3.4.1 HTTPS 安全传输设计 28
3.4.2 非阻塞 IO 传输设计 30
3.5 服务器端模块设计 31
3.5.1 存储集群架构设计 31
3.5.2 请求处理系统设计 32
3.5.3 实时监控模块设计 33
第四章 大数据云存储平台实现 35
4.1 平台实现说明 35
4.2 客户端模块实现 36
4.2.1 注册登录实现 36
4.2.2 文件系统过滤驱动实现 37
4.3 传输模块实现 38
4.3.1 HTTPS 连接实现 39
4.3.2 非阻塞 IO 及断点续传实现 39
4.4 服务器端模块实现 40
4.4.1 云存储集群搭建 40
4.4.2 请求处理系统的实现 45
4.4.3 实时监控模块实现 48
4.5 本章小结 52
第五章 大数据云存储平台的性能测试 54
5.1 文件透明加密的测试 54
5.2 用户文件上传和下载的测试与分析 55
5.3 用户登录的响应性测试 56

5.4 文件的同步性能测试 57
第六章 总结与展望 59
6.1 总结 59
6.2 进一步工作展望 59
参考文献 60
攻读学位期间的科研成果 64
学位论文独创性声明 65
学位论文版权使用授权声明 65
致谢 66

CONTENTS

ABSTRACT(CHINESE)..... I

ABSTRACT(ENGLISH) III

CONTENTS(CHINESE) VI

CONTENTS(ENGLISH) IX

Chapter 1 Introduction 1

1.1 Research background and significance 1

1.1.1 Research background 1

1.1.2 Research significance 2

1.2 Research statusnd development balance at home and abroad 4

1.2.1 Cloud storage status 4

1.2.2 Hadoop status 4

1.3 Research content and organization structure of the paper 6

1.3.1 Main research content of this paper 6

1.3.2 Organizational structure of paper 7

Chapter 2 Brief introduction of key technology 8

2.1 Hadoop’s HDFS distributed file system 8

2.2 SpringMVC+Spring+Mybits Backstage frame 9

2.3 The filtering driving principle of file system 11

2.4 HTTPS network security protocol 12

2.5 Netty 13

2.6 Shiro 16

Chapter 3 Analysis and design of large data cloud storage platfor..... 18

3.1 Platform requirement analysis 18

3.1.1 Functional requirements 18

3.1.2 Performance requirement 19

3.2 Overall design of platform 20

3.2.1 Overall architecture design ofplatform 20

3.2.2 Principle of module division 22

3.2.3 Overall design of platform function module 22

3.3 Client module design 24

3.3.1 Design of register and login module....26
3.3.2 Filter driver design of file system....27
3.4 Design of transmission module....28
3.4.1 HTTPS secure transmission design....28
3.4.2 Non-blocking IO transmission design....30
3.5 Design of server-side module....31
3.5.1 Storage cluster architecture design....31
3.5.2 Request processing system design....32
3.5.3 Design of real-time monitoring module....33
Chapter 4 Implementation of large data cloud storage platform....35
4.1 Platform implementation....35
4.2 Client module implementation....36
4.2.1 Implementation of register and login....36
4.2.2 File system filter driver implementation....37
4.3 Transmission module implementation....38
4.3.1 HTTPS connection implementation....39
4.3.2 Non-blocking IO and HTTP implementation....39
4.4 Server-side module implementation....40
4.4.1 Cloud storage cluster building....40
4.4.2 The implementation of the request processing system....45
4.4.3 Realization of real-time monitoring module....48
4.5 Summary....52
Chapter 5 Performance test of large data cloud storage platform....54
5.1 Test for transparent encryption of files....54
5.2 Test and analysis of user file uploading and downloading....55
5.3 Responsiveness test of user login....56
5.4 File synchronization performance test....57
Chapter 6 Summary and Prospect....59
6.1 summary....59
6.2 Prospect of further work....59
Reference....60

Achievements researched during the study for Master Degree 64
Original Creative Statement 65
Copyright License Statement 65
Acknowlegements 66

第一章 绪论

1.1 研究背景及意义

1.1.1 研究背景

自世界上第一台计算机诞生以后，计算机技术始终处于高速发展阶段。再到 1980 代以后，互联网技术在世界各地引起了极大的变化，对每个国家的经济乃至军事发展都产生的深远流长的影响，同时也植根到我们的日常生活中。随着个人计算机的逐步普及，在当前社会都能看到个人 PC 身影出现在各行各业中，成为了人们必不可少的工具，而不再是以奢侈品的身份存在。互联网虽然给我们的日常生活、学习和工作上带来了前所未有便利，但人们仍然希望互联网可以提供更为体质、更为方便、多样性的服务供人们使用。于是，云计算的理解在顺应潮流的大势下应运而生。它进一步的对互联网的潜在价值进行深入挖掘。云计算可以原来的互联网的基础对他的潜能尽最大限度在发挥。在当代，经过几年的逐步发展，云计算已经成为未来不可阻挡的发展趋势。

人们的生活离不开水，于是就旦生了自来水厂并向人们提供了干净的水资源；人们的生活和日常生产都需要电作为能源，于是就建起了发电厂并提供给我们电力资源。与此相同，云计算的出现也是为了完成人们的某种目的，在互联网上给人们提供相应的服务。云计算的出现，是 21 世纪的新一轮技术革命，给社会带来的翻天覆地的影响与变化。

计算模式因云计算我出现也发生变化。与传统的计算模式相比，云计算则是把计算模型分配到云端的多台计算设备上，共同协助完成同一计算。这些云端的计算机，在物理距离上，它们可能相隔很远，也能相互协调地完成分发给他们的计算任务以对外共同地提供服务。云计算的诞生是在网络计算、分布式计算、并行计算的基础上逐步发展而来的。云计算通过上述技术将相关的软件与硬件资源都组装到一起以共同提供一种服务给人们使用。

云存储是云计算的延伸和发展。传统的存储模式满足不了高速发展的云计算的存储要求，由此衍生出来了云存储。我们可以把一个以数据存储与管理为核心服务的云计算系统看作是云存储系统。云计算的主要功能是计算，而云存储的主要功能是存储。

云存储提供给用户的服务是数据存储、访问和数据管理服务，而传统的存储是指某些具体的存储设备，所以两者有本质上的不同。正因为此，所以云存储的重点是提供一种有别于传统存储方式的服务，而不仅仅只是看作一种存储系统而存在。

互联网的蓬勃发展致使全球整体数据量出现爆炸性增长。在全球数据量以指数增长的背景下，由于物理设备的的约束，如硬盘的容量、服务器性能的限制，使传统的存储方式在扩展其性能与容量上出现瓶颈。随着个人数据的日益增加，传统的存储方式已越来越难以满足用户的日常存储需求。在云存储服务出现之前，用户为了解决大量数据的存储需求，需要把个人的存储系统进行升级扩展，以获得更大存储容量和性能。但这种方式却会导致用户的存储成本增加。另外这种解决方式也只能解决当前的存储问题，治标不治本。云存储技术出现后，用户通过使用云存储服务即可方便地解决大量数据的存储问题。用户完全不必关心云存储服务的物理设备问题，只需要支付比自己升级存储设备少得多的云存储服务费，即可使用海量的云存储服务。用户只需要使用互联网就能够实现随时随地访问、存储个人数据的需求。云存储拥有很多优点，包括成本低、扩展性高、安全性和可靠性高以及使用便捷等优点。正因为此，云存储技术在数据存储领域有着不可估计的使用前景。

当前云存储已经是未来存储领域的趋势。由于一些相关云存储技术的开源性，云存储的应用领域也随之越来越广泛。现有的网络存储架构被云存储的兴起所颠覆，在不久的未来，云存储势必取代传统的数据存储方式。

1.1.2 研究意义

在上一小节中我们也已经提及过，在当前全球数据量大规模爆发的形势下，相比于传统存储模式，云存储在海量数据存储方面有着得天独厚的优势，这决定着它在未来有着不可估量的研究与应用前景。云存储技术是当下国内外最为火热的研究热点之一。

分布式文件系统是云存储的基础，也是其最核心的技术。在当下为数不多的云存储相关技术中，Hadoop 的 HDFS 文件系统是当前所有 IT 公司使用最广泛的分布式文件。把具有高稳定性与可扩展性等优点的分布式计算平台运行在廉价、使用广泛的硬件设备上是 Hadoop 的设计初衷。HDFS 是谷歌文件系统的开源实现项目，在云存储系统的搭建中给各国科研机构与公司提供有效的解决方案，因此当前在云存储领域最重

要的研究对象非 Hadop 的 HDFS 莫属。有根据相关调查分析，一些国内知名公司就是运用 Hadoop 的 HDFS 分布式文件系统来搭建云存储系统，以提供云存储服务的。如中国移动研发的“大云”系统，即 Big Cloud，就是采用 Hadoop 的 HDFS 来搭建的。该平台具有多种功能，除了云存储功能外，还包括弹性计算、并行计算等等。还有中国的华为、搜索引擎巨头百度都是使用 Hadoop 来实现云计算系统的搭建。所以，研究 Hadoop 的 HDFS 分布式文件系统是具有极其重要的意义[1]。

目前 HDFS 分布式文件系统还存在很多缺陷，需要在设计上继续对其进行改进。例如在一般的 HDFS 分部式文件系统中，集群内只有一个 namenode 节点负责元数据信息的存储与管理。只有 namenode 出现故障，那么整个云存储集群将不可以正常运行。此外，单个 namenode 的内存容量很有限，直接影响了整个分式性集群的性能；存储在 HDFS 系统上的文件是没有经过加密处理的，其的安全性也得不到足够保障，一旦 HDFS 上的数据发生泄漏，将造成不可挽回的损失。

综上所述，本论文中研究以 Hadoop 为基础的大数据云存储平台，具有以下重要单方：

在当前为数不多的云计算相关技术中，Hadoop 中一大研究热点，所以通过对 hadoop 的 hdfs 分布式文件系统的研究可以让我们把握与云存储与云计算有关的领域的发展方向和获取最新前沿信息。

HDFS 尚存在很多有待改进的地方。第一，针对 HDFS 的单一 namenode 问题，增加 namenode 的备用节点，可提高 HDFS 的稳定性与可靠性。第二，针对 HDFS 分布式文件系统对数据的安全不加密问题，需要额外附加一层数据加密机制，把将要存储在 HDFS 上的数据先进行加密再存储，可提高 HDFS 的安全性。

HDFS 可在廉价的服务器上运行，对硬件的性能要求也不高。因为对 HDFS 的研究可降低云存储系统的成本。

基于 hadoop 的一系列优点和业界对 hadoop 的热衷，对 HDFS 的研究一旦取得一定的成果，将会推动 hadoop 的广泛用使用。

1.2 国内外研究现状和发展均势

1.2.1 云存储现状

云存储系统相比于传统存储模式具有先天的优势，它的优点包括有易访问、可靠性高、成本低、扩展性好等优点。近几年来，云存储技术广泛受到国内众多企业与用户的青睐，因为采用 hadoop 技术所开发的应用也越来越广泛，促使各大存储厂商由以往的销售软硬件产品的方式转变到以提供存储服务为主[1]。在这种情况下，一些拥有大规模硬件设施的互联网公司也相继转型以提供云存储服务为主。他们投入了大量的人力物力来开发和推广自己的云存储产品和服务。接下来，我们将介绍国内外主流的云存储服务，使我们更好地了解云存储应用的现在和未来的发展方向。

亚马逊是第一家推出云存储服务的公司，也是云存储行业中最成功的公司。亚马逊早在 2006 年初就已经开发了一个以销售云存储服务为主的 S3 服务系统——即简单存储服务，以充分利用公司空闲的硬件设备为公司带来利润。S3 服务凭借其廉价的存储成本与稳定的服务，在短时间内就占据了大量的市场份额。除此之外，亚马逊公司还在后续推出了弹性块存储技术，以支持数据持续性存储。而谷歌公司也有其云存储服务，名为 GSD——全称为 Good Storage for Developers。它拥有多个数据中心以存储用户数据。早在 2007 年，微软公司就已经有自己的云存储服务了，那就是人们熟悉的网络硬盘服务。它给用户提供的存储空间在 50G 。2009 年，为了增强自己在云存储服务方面的实力，IBM 公司推出了名为全频级智能云存储的云计算领域存储战略计划，通过虚拟化存储、归档和其他相关技术，为海内外企业提供应用程序层面级的服务。而在 2010 年，惠普公司为了应对云存储的发展趋势，成功收购了 3PAR 存储厂商，并在原有的存储产品的基础上对其进行了扩展，使公司的云存储实力得到极大增强，从而使惠普的云存储服务更加强大。

云存储服务也在中国受到广泛关注。华为的第一个互联网产品就是华为网盘。国内一些 IT 公司也为了应对云存储在全球的发展趋势，纷纷推出了自己的云存储产品和服务。其中，包括国内搜索引擎巨头百度的百度云网盘、杀毒软件的国内领头 360 的 360 云盘与金山的金山快盘等等。

1.2.2 Hadoop 现状

业界和学术界对 Hadoop 密切关注，并将 Hadoop 广泛应用于互联网行业。而随着新的业务模式不断推陈出新，未来 Hadoop 必定会被运用到更多的领域中，以提供更多样化的优质服务。

雅虎美国的全球知道 IT 公司，也是第一个对 Hadoop 进行深度研究和使用的公司。直至目前为止，雅虎在公司里面搭建的 Hadoop 集群服务节点数已经超过惊人的 10000 个节点，为搜索引擎与广告系统的研究提供了硬件方面的支持。即使如此，目前雅虎依然进行着 Hadoop 有关的科研实验项目。Facebook 是全球知道的社交巨头，有着全球第二大的 Hadoop 集群规模。Facebook 采用 Hadoop 来搭建一个可扩展的、高效的系统来处理由于网站使用率高速增加而出现指数式增长的日志和数据。除此之外，Facebook 使用 Hadoop 作为引擎，以推动其社交网站的信息传送和处理网站属性。采取 Hadoop 以实现数据分析，现已成为 Facebook 处理海量数据的选择之一。

在国内，不少企业选择 Hadoop 的 HDFS 作为文件系统是因为 Hadoop 的开源性。2011 年 12 月，以“各级党委数据掘宝”为主题的会议 Hadoop in China 2011 在中国北京成功举办。此次会议吸引了世界各地的 Hadoop 开发者、使用者和专家。这一举动足以说明 Hadoop 已经在国内得到了足够的重视。包括华为腾讯阿里巴巴等等国内知道 IT 企业出席了此次会议并热烈的讨论了有关 Hadoop 的应用。阿里巴巴与百度是当前国内使用 Hadoop 规模最大的两个 IT 企业。

由于淘宝购物平台的用户数量与并商品种类信息量巨大，因此普通文件系统根本对这些海量数据进行存储和处理。鉴于此，淘宝根据自身需求对 Hadoop 进行改造以处理上述的海量数据。Hadoop 集群经过淘宝改造后能够依照自身业务需求进行动态伸缩。淘宝的 Hadoop 集群由 1100 个服务蘠组成，其总存储容量达利惊人的 9PB。此外淘宝的云技术团队在服用 Hadoop 的同时还对其做大量的个性化修改和优化。例如对成为 Hadoop 集群瓶颈的单一 namenode 问题开设我专门的项目小组。经过修改后 namenode 的吞吐量在性能上提升了整整八倍以上。另外淘宝对 MapReduce 也做大量的性能优化。此后，淘宝把有关 Hadoop 的大部分科研成果在其官网上公布，这对国内、外 Hadoop 的研究与改进给予了很大帮助。

Hadoop 的 C++版本（Hadoop C++ Extension）是由百度公司自主研发的。而且在 Hadoop 中国 2010 大会上，百度公司把该 C++版本的 Hadoop 源码和自己的调度器分享出来。C++与 Java 的最大区别就在于内存资源的申请与释放上，因为 Hadoop C++

Extension 成功绕开了 Java 在内存释放与资料申请上的弊端，在很大程序上提高了 Hadoop 的计算效率。直至当前为止，百度公司在内部搭建了规模达 4 千多个服务节点的 Hadoop 集群，主要用于数据存储业务，同时还负责线下挖掘计算以及线上实时性计算相关工作。

当前有关 Hadoop 的科研研究在国内外正如火如荼地进行，相信未来人们取得丰硕的成果，让 Hadoop 提供更多样性的高质量服务。

1.3 论文研究内容及组织结构

1.3.1 论文主要研究内容

针对 Hadoop 的 HDFS 分布式云存储 namenode 可靠性问题和数据安全问题[5]。本论文从安全角度和可靠性角度为立足点，搭建一个包括服务器端、传输模块、客户端三大模块的大数据 云存储平台，并通过相关辅助技术的技术与使用最终完成平台的实现。其中，服务端的功能上我们还增加实时监控机制，以便于开发人员实时掌握云存储平台的实时运行状况；传输模块使用多线程传输与断点续传技术以及在这基础上引入 HTTPS 加密，可以进一步确保数据在传输过程中的可靠性与安全性。客户端包括虚拟硬盘与文件过程驱动加密机制，确保文件在本地的安全性。

本主主要研究以下内容：

一级加解密机制的研究

该大数据云存储平台的客户端采用虚拟硬盘与文件过滤驱动两层加密。为确保客户端模块的实现，我们需要对文件过滤驱动加解密技术与虚拟硬盘进行深入研究，合理的选择实现方式。

基于 HDFS 的云存储群集研究

大数据云存储平台属于云存储领域，最终目的是为用户提供海量的数据存储服务。基于 Hadoop 的多个优点，本论文采用 Hadoop 的 HDFS 分布式文件系统作为云存储集群。因此，要论文需要深入研究 Hadoop 的集群搭建、HDFS 原理以及其提供的读写操作 API。

网络传输与安全策略的研究

文件传输是每款互联网产品都必须有的部分。该平台中为保障数据在整个传输过

程的安全性与可靠性，我们需要深入研究多线程、断点续传以及 HTTPS 等相关技术，为这部分功能模块的实现打下扎实的理论知道基础。

基于 Hadoop 与 SSM 的大数据云存储平台的实现

在前三我们对相关技术进行深入研究的基础上，我们再加入 SSM 后台开发框架作为我们的实际直接操作 Hadoop 集群的后台 Web 模块。添加 SSM 后台的两个作用，一是接收网络用户的请求，二是根据不同的请用完成对 hadoop 集群的操作或进行其他相关操作。这部分的最终目的是搭建一个最可用的、安全性高的云存储平台，提供优质云存储服务。

1.3.2 论文组织结构

本论文的组织结构如下：

结论。这章主要对论文的研究背景、意义、当前国内外云存储现在和发展趋势的介绍，在这章的结尾部分我们还将对本论文的组织结构进行简要阐述。

本论文采用的关键技术的研究。这章是对搭建大数据云存储平台需要采用的技术进行分析，包括 Hadoop 的 HDFS 分布式文件系统、虚拟硬盘、文件过滤驱动加密、多线程断点续传以及 HTTPS 安全协议等技术的研究。

大数据云存储平台的整体设计。这章中根据功能需求与性能需求，对平台进行整体架构设计，然后再详细对服务端、传输模块、客户端三大模块进行设计。

大数据云存储平台的实现。根据在第三章中的详细分析与设计的基础，阐述这个大数据云存储平台的各个模块开发过程。

平台测试。这章中对大数据云存储平台进行多项测试并展示测试结果与分析。

总结与展望。在这章中我们将总结与讨论一下本论文中的研究工作和取得的实验成果，并对平台目前尚存在的问题与需要改进的地方给出合理意见与展望。

第二章 关键性技术简介

本单主要研究大数据云存储平台相关的技术，包括 Hadoop 的 HDFS 分布式文件系统、SSM 后台框架、透明加解密技术、虚拟硬盘、HTTPS 安全传输协议等技术。研讨 Hadoop 中的 HDFS 系统的组织结构；探究使用 SpringMVC+Spring+Mybits 后台框架实现的 Web 技术；还有透明加解密、虚拟硬盘、HTTPS 的工作原理等待

2.1 Hadoop 的 HDFS 分布式文件系统

Hadoop 是以 HDFS 为基础层的。HDFS 是以谷歌的布式文件系统的思想而设计出来的。HDFS 是一个可以安装在廉价的机器上的分布式集群，优点是拥有极高容错性。HDFS 能够读写海量数据的重要原因是它拥有极高的数据吞吐量；HDFS 的系统扩展性很强，能依据系统的实时需求，动态调整分布式文件系统集群的规模与数量。HDFS 具有以下几个特点：

HDFS 可支持超大规模的数据集合,规模可达几百 GB，甚至几百 TB，而整个文件系统承受的数据量已经达到了惊人的几十 PB 了。HDFS 不支持修改，即文件存入到 HDFS 后，就不可再被修改了。而随着各种应用对文件操作的需要，目前 HDFS 文件系统在原有的基础上新增加了对文件的追加功能，即用户可以向存储在 HDFS 上的文件的尾部追加新的内容，但 HDFS 还是未能实现对文件的任意修改和添加操作。

由于 HDFS 能够在廉价的 PC 的部署集群，所以会经常发生集群内某些服务节点宕机或某些原因不能正常工作等。为确保整个 HDFS 集群不因单个节点的损坏也导致运行的严重局面，HDFS 拥有自己备份机制，HDFS 中文件，实现上会按照一定的算法被分割成多份，每份共有多个副本（可通过配置文件设置），存储集群上的不同节点上。

HDFS 的可扩展性强。HDFS 可以动态地向集群中增加或移除服务节点，以适应不同的应用场合需求，同时对整个集群的正常运行不产生任何负面影响，HDFS 会对自动更新集群内的节点状态，以达到新的稳定状态。

HDFS 的容错机制。HDFS 对单个文件的存储也是有限制的，HDFS 可以通过设定，可设置允许存储的单的文件的大小最大为 64M。因为 HDFS 对于大文件会进行切割并对每个文件块进行多个备份，然后再存储到不同的节点上。

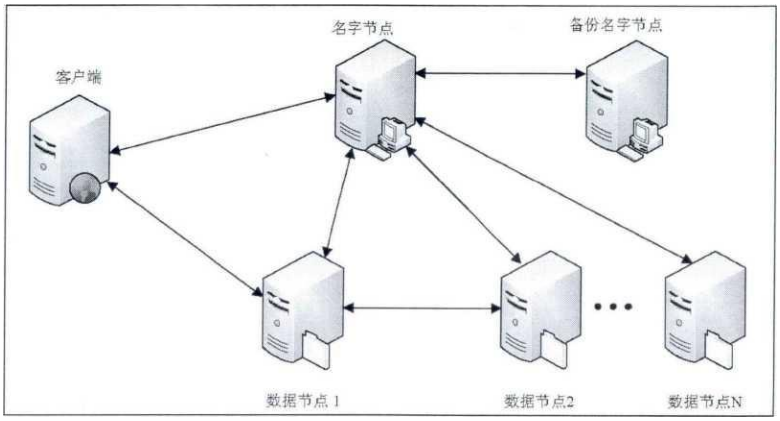
HDFS 的系统架构如下图所示。HDFS 属于典型的主/从结构。一个完整的 HDFS

集群应该包含三个部分：用于管理元数据的 namenode 主节点、真正负责存储数据的 datanode 从节点、用于元数据合并的 secondary namenode 辅助节点。用户可以通过 HDFS 提供的 DFSClient 客户端登录到 HDFS 并对其进行读写操作。

图 2-1 HDFS 分布图

Fig.2-1 Distribution of HDFS

namenode 节点是集群中的大脑，其主要作用是负责对集群中元数据存储和以整个集群的控制[5]。Secondary namenode 是辅助节点，其主要作用是对 namenode 上的 editlogng 与 Fileimage 进行合并，以提高 namenode 的执行效率，防止存储在内存中的元数据丢失。datanode 是实际存储的物理介质，由 namenode 节点控制对 datenode 的读写和容错控制操作。DFSClient 是 HDFS 提供给用户进行 HDFS 读写操作的客户端。



2.2 SpringMVC+Spring+Mybits 后台框架

SSM 框架是 SpringMVC+Spring+Mybits 的首字母缩写，它是 SSH（即 Struct2+Spring+Hibernate 框架）后，当前最流行的 JavaEE 框架，适用于搭建各种 Web 应用系统[12]。下面分别对 SSM 的三大框架进行简介。

SpringMVC 框架简介：

SpringMVC 是 Spring 家族的一员，与 Spring 完美整合，与 Spring 完成结合。SrpingMVC 原生的支持 Spirng，与 Struct2 相比，它使原来的 Web 开发更加规范[13]。SrpingMVC 也是使用 MVC 模块的思想设计的，把控制器、模型对象、分派器进行分离。下图为 SrpingMVC 的工作原理图。

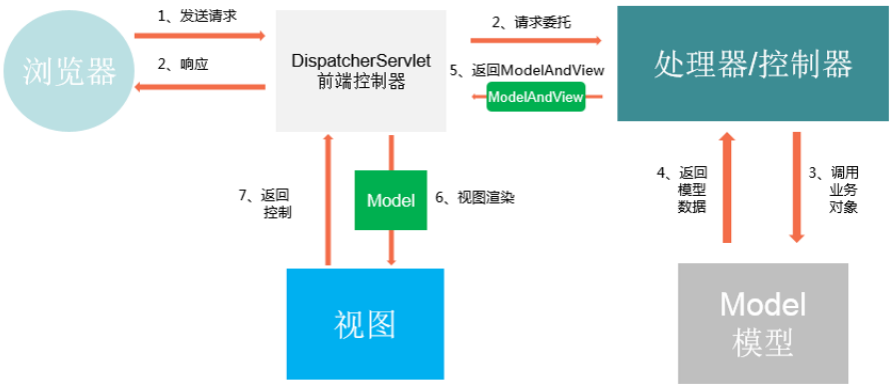


图 2-2 SpringMVC 架构图

Fig.2-2 SpringMVC architecture diagram

Spring 框架简介：

Spring 是一个 javaee 开发框架，具开源性与轻量级的特点，其极大的降低企业级应用的开开发难度[10]。 之前只能由 EJB 来完成的事情，也可以通过 Spring 来完成了。Spring 还可以在其他应用开发中使用，以实现程序的松耦合、降低开发难度和方便开发人员调试。Spring 框架具有两个很重要的特性，分别面向切面编程、控制反转[1]。

控制反转(IoP)。控制反转主要是通过工厂模式扩展而来的。开发人员把 java 对象交给 Spring 托管。spring 在启动时会初始化好 java 对象，接着在我们需要时，Spring 会把这些 java 对象作为属性，分配到对应的其他 java 对象里面。控制反转是通过 setter 方法注入的，这样开发人员将不再需要 new 对象了。

面向切面(AOP)。面向切面是 spring 的特性，与 OOP 一样都是一个编程思想，而不是一种技术。OOP 是从上往下的思维方向定义程序结构，却无法定义从左往右的程序结构。例如与权限相关的功能，一般都分布在部分对象中，但又与这些对象的主要功能不相干。如果只使用 OOP 的编程思想，则会导致很多相同或类似的冗余代码，不利于以后应用程序的维护与升级。而采用 AOP 设计思想，把需要被嵌入到其他代码中的业务逻辑包装成一个切面再穿插到指定的对象中。

主要有两大类技术可以实现 AOP。第一种实现技术是使用设计模式中的动态代理，预先截取信息，把原对象实行封装，以增加原对象的行为。第二种实现技术是静态织入。

MyBatis 框架简介：

MyBatis 是阿帕奇公司的一个开源项目，由 java 语言编写。它是一个典型的持久层框架[15]。MyBatis 是对 JDBC 的封装，让对数据库的操作变得透明化。Mybatis 通过设

置映射关系，把关系型数据库中的每一行与 java 对象一一对应起来。下图为 MyBatis 的关系图。

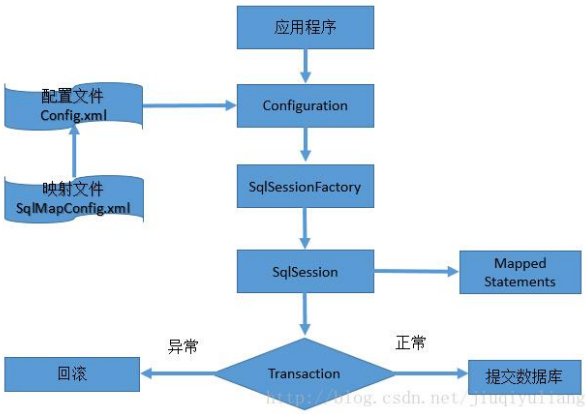


图 2-3 MyBatis 框架图

Fig.2-3 MyBatis architecture diagram

2.3 文件系统过滤驱动原理

应用层发出的每一个 I/O 请求，从 I/O 管理器开始出发，按顺序从设备栈的最上面往下传送。请求每经过一层，与当前设备对象相关的程序会由系统调用来对该请求进行处理。这个分层的驱动程序模型，可以实现让驱动程序创建一个设备对象粘附到另一个设备对象上面。之后如果 I/O 管理器在传送请求到下一级前发现有一个设备对象粘附在其上面，则请求会先传送到此设备对象并由它发送到下一级。于是可以可以在这个设备对象中实现我们预设的处理逻辑，对特定的请求进行预定的处理，对其他请求则直接放行。我们称这个粘附的设备对象为 FiDO 对象，即我们俗称的过滤设备对象，而与这个粘附设备对象有关的驱动就叫做过滤驱动程序了。基于这个原理，就可以在操作系统的文件系统设备对象上增加一个包含粘附设备对象的过滤驱动，把所有访问文件系统的请求都拦截下来，对特定的访问请求执行预先的处理逻辑，其他请求就直接放行[29]-[35]。下图为其运行原理图。

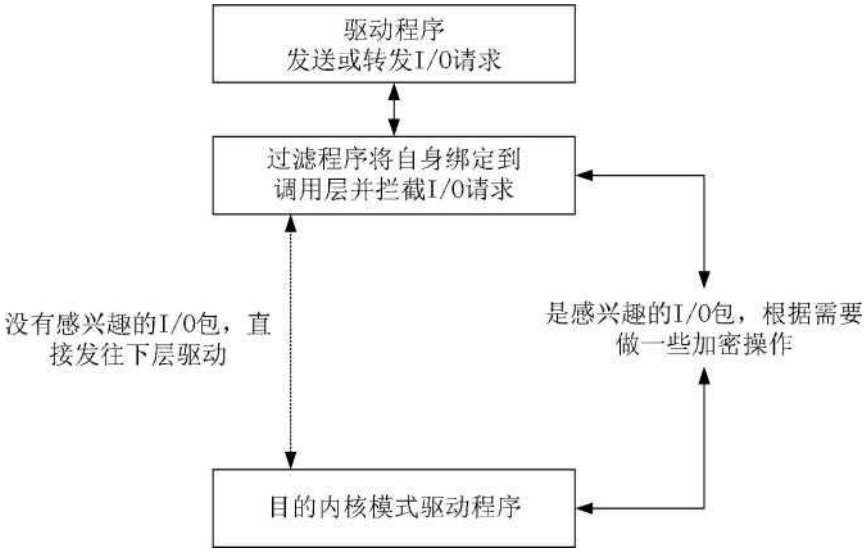


图 2-4 文件系统过滤驱动原理图

Fig.2-4 File system filter driver schematic diagram

2.4 HTTPS 网络安全协议

HTTPS 是一个传输协议。HTTP 是以明文的方式传输数据，换言之没有经过加密处理，因为其安全性很差。相比之下，HTTPS 安全性非常高，因为 HTTPS 是 HTTP 与 TLS 或 SSL 的组合，即其经过 TLS 或 SSL 加密[24]。

HTTPS 需要在两端间成功握手后才可以进行数据传输。在这个过程中它们会确定用于数据加密的密码信息。TLS/SSL 协议是一套采用多种加密算法的传输协议[25]。下面为 HTTPS 握手的具体步骤。

第一步，加密规则由浏览器发送给服务端。

第二步，服务端从中筛选加密算法与 HASH 算法，再把证书发送回浏览器。其中证书里包括加密公钥、网站地址以往此证书的颁发机构等等。

第三步，取得证书后，浏览器执行下列操作：1，检查证书是不是合法的，包括当前访问的网址与证书内的网址是否匹配，颁发机构是不是安全的。2，随后浏览器在接受证书的提前下随机产生一串数字密码对公钥加密。3，握手信息会采用双方指定的 HASH 算法进行计算，计算结果也用上面的数字密码加密，再把全部信息都发回服务端。

第四步，服务端收到浏览器发送回的信息执行下列操作：1。对客户端发来的信息，采用私密解密出密码并用其对双方握手的信息执行解密操作和验证服务端 HASH 是否与

浏览器发送来的 HASH 相同。2，给浏览器发送一段采用此密码加密的信息。

第五步，浏览器把信息进行解密。其后再对信息的 HASH 计算，若与服务端的一致，则成功并结束握手过程。之后，所有数据都采用此数字密码加密再通过 HTTP 协议传输[28]。

下图为 HTTPS 对应的通信原理时序图。

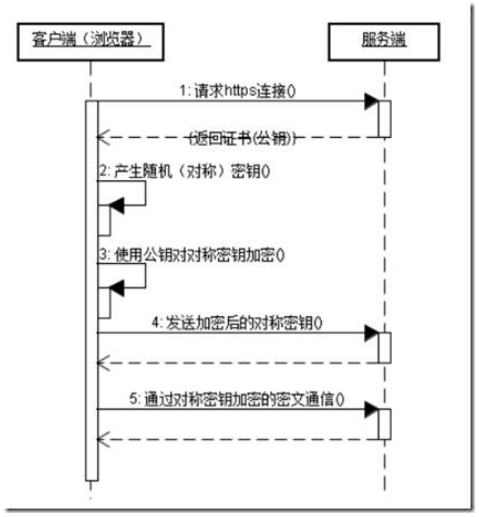


图 2-5 HTTPS 通信时序图

Fig.2-5 HTTPS communication sequence diagram

2.5 Netty

Netty 是一个性能极高、异步事件驱动的框架。它是由 JBOSS 公司所开发的项目，是一个 NIO 框架[17][20]。这个框架支持多种通信协议，包括 UDP、TCP 以文件传输协议等。Netty 是一个很优秀的框架，它的所有 IO 操作都是异步非阻塞的。通过使用 Netty，对于所有的 IO 操作，开发人员都可以主动地或由它内部的机制取得结果。 Netty 是目前 IT 行业内最火热、使用最广泛的 NIO 框架，在众多 IT 领域都能看到它的身影，包括去计算领域、互联网领域、通信领域以及当前最火的游戏领域（可提供更高性能的游戏服务器）等领域[18]。下图为 Netty 的框架服务器。

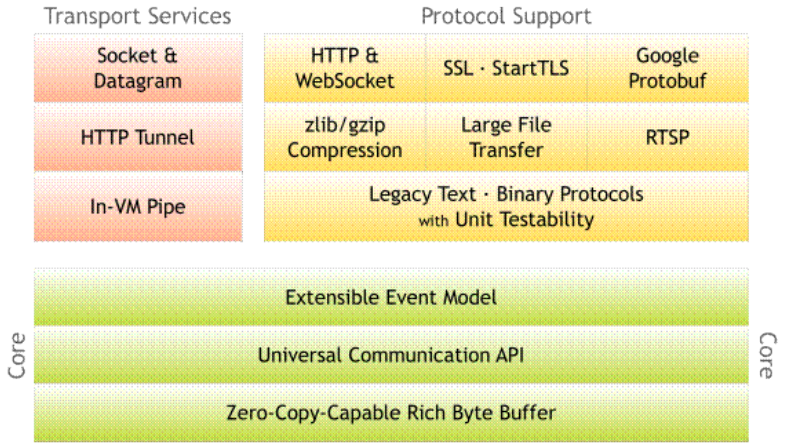


图 2-6 Netty 框架结构图

Fig.2-6 Netty Frame structure diagram

此框架采用典型的 Reactor 模型设计，包括三种线程模型，它们分别是 Reactor 单线程、主从 Reactor 多线程、以及 Reactor 多线程。下面分别简述这三种 Reactor 线程模型[19]。

Reactor 单线程模型意思是说，只在同一个 NIO 程序上完成所有的 IO 操作。这 Netty 可以作为 TCP 连接的 NIO 客户端，也可以作为 TCP 连接的 NIO 服务端，以发起或等待连接。Netty 的所有 IO 操作却不会因阻塞而导致出现性能问题的原因它的 Reactor 模式也是异步非阻塞 IO。因此，从理论层面来说，Reactor 单线程模块对所有 IO 操作，都可以只使用一个线程来处理就已经足够了。下图为该模型的操作图。

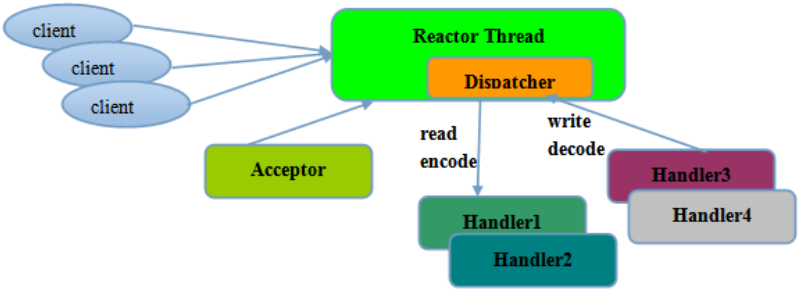


图 2-7 Reactor 单线程模型图

Fig.2-7 Single-threaded model diagram

Reactor 多线程模型：

上一种模块的单线程完全有足够的能力应对一些需求容量相对比较小的使用场合。然而对于一些要求高并发、高负载的场合，那么单线程就表现得无能为力了。所

以，基于此原因就衍生出了 Reactor 多线程模型。该模块与 Reactor 单线程模型中最大的不同在于 Reactor 多线程模型是使用一组 NIO 线程来处理所有 IO 操作的，而只是使用一个。

Acceptor 线程是这个模块是一个专门用于监听服务端的 NIO 线程，它的主要作用就是负责客户端请求的接收。线程与链路的关系为，一个链路只能对应于一个 NIO 线程，而一个 NIO 线程却可以与多条链路对应，这样的设计方式可以解决链路并发操作而出现混乱的问题。此模块采用一整个线程来处理所有的网络 IO，关于线程池的实现标准，可以采用 JDK 内的线程标准实现即可，其含有 N 个线程和一个被等待处理的任务队列，所胡信息的读取、解编码和发送操作都是由此线程池内的 NIO 线程负责的。下图为该模式的操作图。

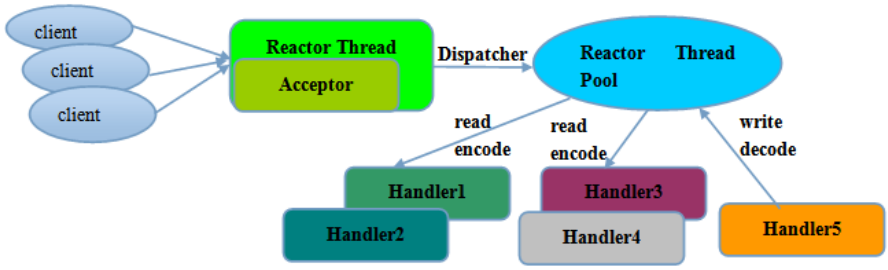


图 2-8 Reactor 多线程模型图

Fig.2-8 Multithreading model diagram

主从 Reactor 多线程模型：

从使用场合的高并发要求的角度比较，Reactor 单线程 < Reactor 多线程 < 主从 Reactor 多线程。主从 Reator 多线程模式是专门为一些对并发要求极高、极苛刻的场合而设计的。在极高并发场合，Reactor 多线程模式中出现问题的地方在于它只使用一个 Acceptor 线程负责分发网络 IO 请求，同时如果此时正在连接的客户端请求需要认证，如 HTTP 和 HTTPS，则严重消耗服务器的性能。所以，从 Acceptor 线程入手，对 Reactor 多线程模型进行改进，就衍生了主从 Reactor 多线程模型了。该模式与 Reactor 多线程模型相比，最大的区别在于，在该模式中使用一个独立出来的 NIO 线程池负责与客户端连接通信，而不是只用一个 NIO 线程了，线程池中的每个 Reactor 都中单独存在于自己的 NIO 线程中的。服务器端的 CPU 是多核的话，那在该模型中，可以实现真正的同时并发处理多个客户端连接请求。该模式的具体执行步骤如下。首先服务端的

Acceptor NIO 线程接收到客户端的 TCP 连接请求（一般还可能伴随有认证信息等等）。Acceptor 收到请求后会马上新创建一个 SocketChannel 对象并使用客户端的信息对其初始化，然后把这个对象注册到 NIO 线程池——这个线程池称为 sub reactor 线程池，的某个具体的线程中，即把 SocketChannel 对象与 NIO 线程绑定，由这个线程负责之后所有关于此 SocketChannel 对象的 IO 操作。而在这个过程中，Acceptor 线程的角色仅仅是用于服务器端与客户端之间的握手、认证等相关连接工作。双方间的通信链路一旦建立，就再没有些 Acceptor 的事了，剩下的 IO 工作都由 sub reactor 线程池上的某个具体与该 SocketChannel 对象绑定的 NIO 线程处理了。下图为该模式的操作图。

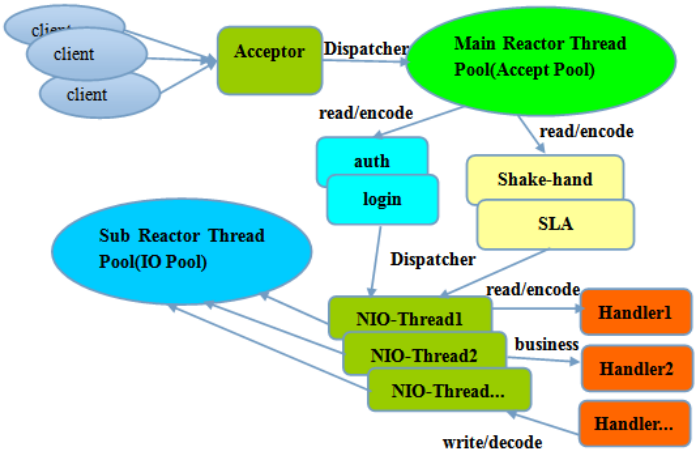


图 2-9 主从 Reactor 多线程模型图

Fig.2-9 Master-slave multithreading model diagram

上面提到 Netty 线程模型并不是固定的，我们使用 Netty，在其启动时，可以通过适当的配置，就可以让 Netty 实现对上面三种线程模型的支持了。基于 Netty 在线程模型方面存在灵活性高等优势，可以适应不同的使用场合，使得在当前 IT 行业中，Netty 框架被广泛使用。

2.6 Shiro

Shiro 是由阿帕奇公司的一个开源项目，是一个安全框架。当前，有很多项目和开发人都是使用 Shiro，其中一个原因是 Shiro 使用起来简单且功能强大。Shiro 给开发人员提供了一个解决文案，以直观的形式设为权限认证等问题[21]。在实际开发中，它的目的是对程序中与安全有关的问题进行管理，以此同时还提供了许多种解决此类问题的

方式。Shiro 支持开发人员自己定义各种行为，因为它是以面向对象与接口驱动为原则。

Shiro 使用起来是非常简单的。与 Spring 自家的安全框架相对，即使没有它强大，但也可以应对实际工作中的绝大多数安全问题。因为，采用这个小巧的 Shiro 框架就可以了。

Shiro 可以完成很多工作。其中最包括最常用控制用户访问和身份验证。另外，Shiro 还可以在没有任何附加条件的情况下，使用 Session 的 API。在 Shiro 中提供了许多中信息数据源，这也是 Shiro 使用方便的原因之一。最后值得一提的一点是，Shiro 是相对独立的，它并不需要依赖于某些框架、服务或容器，就可以实现上面的功能，相对于相当框架来说，方便多了[22][23]。

下图为 Shiro 的架构图。

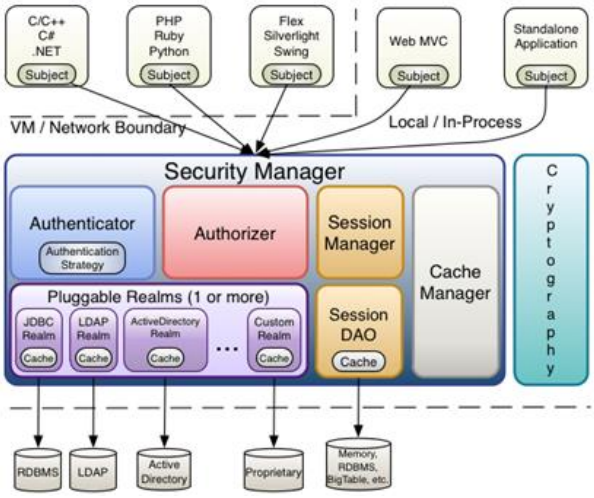


图 2-10 Shiro 框架结构图

Fig.2-10 Shiro fram structure diagram

第三章 大数据云存储平台分析与设计

通过对中国互联网公司的初步研究和分析可知，我们了解大数据平台相关技术，这对我们的设计及其实现许可性垫下深厚的技术基础。在本章中，我们会在其核心技术方面，先分析功能和性能要求，再整体构建架构，最后再仔细设计各模块。

3.1 平台要求分析

大数据存储平台领域中，我们通过对国内的公司进行调研，如百度云盘、360 云盘等等，可知，一般的大数据存储平台都由三部分构成：1，客户端：这部分通常都以公司个性订制的软件形式存在，如手机 APP，web 网页客户端等等。个人用户一般可以此客户端可以实现与系统的数据存储交互。2，大数据系统的具体业务功能模块。这部分是客户端与大数据平台的中间逻辑交互层，对上负责与客户端交互、对下负责操作数据存储平台，实现文件的存储。此部分功能包括有注册与登录、数据的安全性要求。3，大数据云存储集群。在三在部分中，此部分是平台的核心部分，其是文件具体存储的物理介质。基于对中国云存储公司的分析，此部分我们最终选用 hadoop[2]。

近年关于云存储的安全问题屡见不鲜，企业和个人用户对大数据存储系统的安全性出了最高的需求。针对于此，下面分析功能和性能要求。3.1.1 功能要求

功能需求有：1，普通用户与 vip 用户功能权限认证；2 对于已注册的用户，系统对于一定数据的免费存储空间供其体验；3 安全机制，确保传输安全。

客户端要求

每一件成功的产品（即使是以云存储为核心的客户端 APP），都需要注册登录界面。这里我们要求这两个界面越简单越好。

虚拟硬盘的使用方式与我们在个人 PC 上的硬盘（如 C 盘、D 盘）的使用很相似。其能够将文件复制到如“我的网盘”里面，或通过以拉动文件到文件夹的形式，实现本地文件的上传与云盘内文件间的移动。

文件加密需求

自 21 世纪互联网调整发展以来，个人资料信息以指数增长，且一般都以网络为媒介，存储其资料。与此伴随而来就是其数据的安全性问题，造成了很多麻烦。当代用户对其数据量和数据价值的升高以及云存储技术的高速发展，个人用户的数据遭受到

前所未有的安全性威胁。如果网络黑客一旦成功盗取到个人或企业超大数量的数据，负影响将是空前的，后果无化估计。所以数据安全很重要。

我们可以从应用安全的角度为出发点，可采用安全策略对数据信息进行加密，以提高大数据云存储系统的安全性。根据实际情况，提高大数据云存储平台的安全性可以从心下两个方面入手：1 在本地通过透明加密技术加密文件；2 文件在网络传输过程中，而对其再一次进行加密，一般常见的加密方法有 SSL[26]，从而用户数据的整体安全性得到提高。

存储和监控要求

数据存储是服务端的一个主要功能。现通琮下面两方面加以阐述具体需求：

数据副本数量配置：集群内服务器节点宕机是时有发生的事情，所以备份多个副本到不同的节点是一个解决办法。同时对于不同的用户级别，如普通用户、vip 用户、svip 用户，我们将提供不同级别的安全存储等级，即提供不同和用户数据存储副本数。

数据异地存储配置：为使用户存储在大数据云存储平台的数据的安全性足够可靠，在不同的节点上存储用户数据和副本，方可确保用户数据的安全性。

合理搭建好大数据云存储平台后，为其提供相应的监控功能，能够实时监控存储在系统里的文件及其同步状况。具体有以下内容需要进行监控：1，文件的同步情况：以直观简洁的方式展示当下正在进行数据同步的用户和用户具体同步中的文件，且能够实时监控指定文件的同步状况；2，元数据节点运行情况：内存、cpu 使用率和线程数；3，文件数据变化情况。

3.1.2 性能要求

功能要求完成后开始分析性能要求，以提供更好的用户体验。基于以 hadoopd 而搭建的大数据云存储平台，下面为性能要求。

响应要求

短于 3 秒的登录时间。在平台访问高峰期，服务端线程性暴增，处于高负荷状态，此时我们要求登录时间应不超过 3 秒。

可靠性要求

此大数据云存储平台的可靠性还依靠于具体的网络环境。可靠性要求如下。

峰值并发数：>5000；

响应失败数，每 1 千个请求中应小于或等于 20 个用户请求失败；

如若对大数据云存储平台提出更亢的性能需求的话，可以在原来的大数据服务集群的基本上选用性能更高的机器或横向增加更多的服务器节点。

可用性需求

在可用性需求方面，大数据云存储平台需要满足以下的性能需求：

大数据云存储平台需要满足 24 小时不间断的网络服务；

在同步操作完成后，人个用户客户端里面的虚拟硬盘中的文件数据必须与大数据云存储平台服务集群里的文件保持一致。

同时，即使客户端断电或断网后，客户端虚拟硬盘里的文件资料也不会因此而丢失；

大数据云存储平台允许多同一账号多客户端登录，即 pc 网页端、pc 客户端、安卓客户端、ios 客户端都能同步其虚拟硬盘里的文件，使其在多平台客户端下都保持文件一致性。

3.2 平台总体设计

3.2.1 平台总体架构设计

近年来，关于企业用户数据泄漏的新闻层出不穷，为搭建可靠性高、安全性强的大数据云存储系统，我们通过在客户端用户层和传输过程中加密数据，设计满足要求的平台。

通过初步分析百度云盘和 360，我们他用 c/s 方式，是因为使用我们大数据云存储平台的客户都是特定的群体，其考虑的主要是用户体验性大于便捷性，极少会通过 web 网页来登录个人账号来下载文件。服务器的反映速度是良好响应决策的直接表现。使用这种设计方式对系统以后的维护有很大的保障。

基于安全问题也是我们首要考虑的问题之一，因为数据在传输方面我们需要单独作为一个独立的模块加以研究。因为，从整体上来说，我们把平台分为三个模块。

客户端模块

只有注册登录个人账号的前提下，才能像使用个人 PC 的 C 盘、D 盘那样，快捷高效地操作客户端的的虚拟硬盘。我们在客户端选取一级加密的方案对个人用户的资

料进行加密。所谓一级加密就是采用一层加密方案。它是使用文件过滤驱动技术对文件进行加密，是一个透明高效的文件加密技术。可以保证通过二级加密后的文件再存储大数据云存储平台是唯一的，只有用户本人可以在客户端可见——文件在客户端使用私钥被解码后才能被正确打开。这样即使数据不幸被泄漏，他人获取到的都是乱码的文件。另一方面，第二级文件加密是完成透明的，整个文件的加密过程，用户几乎完全察觉不了，因此提升了用户体验[33]。

安全可靠传输模块

客户端与服务器端需要相互传输数据。客户端上传文件到服务器，另一个方面客户端也要下载文件到服务器，这就要求在双向的传输过程中，数据的传输都是安全的。我们的大数据云存储平台将采用消息中间件的方案，数据在传输过程中，使用多线程和断点续传技术；同步还将使用 SSL 加密协议和 HTTP 传输协议相结合，通过 HTTPS 安全传输方式提升用户文凭数据的安全性[27]。

服务器端模块

这个模块由两部分部组成，第一部分是请求处理系统，这里我们决定采用当前 IT 行业最流行的 SSM 框架，即 SrpingMVC+Spring+Mybatis“全家桐三件套”。使用 SSM 最重要的原因是它的开源性和快速开发能力。通过 SSM 框架，在大大减少我们的工作量的情况下，可以快速开发出一个企业级别的 Web 应用。同时基于 SSM 的开源性和超高的使用率，我们很方便的在网上解决我们在开发中遇到的各种问题。此请求处理系统有以下几个主要功能：一，通过 Mybatis 持久层框架操作关系就数据库，这里我们使用免费的 MySql[37]，以存放用户相关的信息；二，通过 Shore 权限认证框架，实现不同的用户提供不同的安全级别；三，响应用户请求并操作存储集群。

第二部分就是存储集群。这部分我们使用 Hadoop 来实现，一般包含三个角色，分别元数据节点，数据节点和元数据合并辅助节点。元数据节点负责管理整个大数据云存储平台。数据节点负责实现数据的存储。而元数据合并节点则是 Hadoop 为了减轻元数据节点的性能负担所设计出来的专用来合并元数据的辅助节点。由于考虑到平台的可靠性，如果元数据节点宕机，那整个平台都处理瘫痪存储，为此我们在设计上增加了一个备用元数据节点，组成一个联邦共同管理元数据。这样即使无数据节点出现问题，那么备用节点也能及时运行起来，代替原元数据节点继续工作，及时将集群 从故障中恢复起来[3]。

基于以上的三个模块的陈述，我们的平台的架构如下。

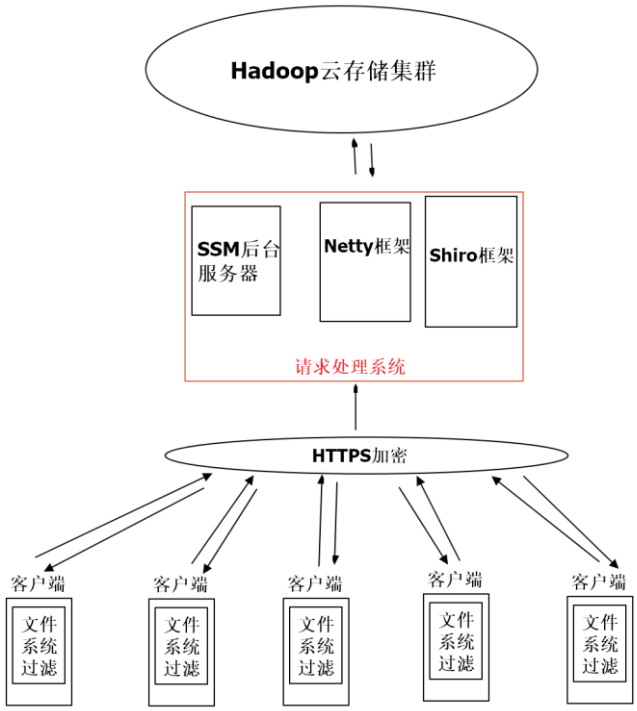


图 3-1 基于 Hadoop 与 SSM 的大数据云存储平台结构图

Fig.3-1 The structure diagram of big data cloud storage platform based on Hadoop and SSM

3.2.2 模块划分原则

由上一节的陈述可知，我们基于 hadoop 而搭建的大数据云存储系统是由三大模块组成，因此各个模块间必定还有很多细分和功能有待完善，所以我们应遵循下列原则来进行相应地开发。

子功能模块必须要尽可能的相对独立，降低其耦合性。多个相互独立的功能子模块共同构成我们的大数据云存储平台，在设计时就可以充分地使它们之间相互独立，各模块间如果不是迫不得已，都不应该存在模块间的调用，功能间的逻辑耦合要尽可能的减少，提高系统的可维护性。

3.2.3 平台功能模块总体设计

由上面的分析可得，我们的平台整体上由三大功能构成，分别是客户端、传输模

块以及服务端。各模块详细设计如下。

客户端

客户端是由大数据云存储平台提供给注册用户直接使用的功能模块，从系统的实现角度出发，有下列几个小功能：

注册登录：我们的用户在注册账号我们提供的信息包括注册的用户名（具有唯一性）、登录密码以及独立的文件数据加密密钥。用户在登录时需要正常填写账号名和登录密码，之后再填写正确的加密密钥才能打开虚拟硬盘使用我们的大数据云存储服务。

文件过滤驱动：上文提到的客户端第二级加密就是文件过滤加密。我们可以在本地文件系统的驱动上添加一屋过滤机制来完成对用户虚拟硬盘中指定文件的透明加密，它的具体设计和实现将在后而的章节中详细介绍[35]。

安全可靠传输

大数据云存储平台的中间模块是安全可靠传输层。这个是中间模块的安全性与可靠性具有极其重要的意义，因为他决定的用户是否会长久地使用此平台。这部分应包括下列功能：

HTTPS 网络传输协议：为提升数据在传输中的安全性，本大数据云存储平台使用 HTTPS 协议以实现数据的传输。在下一章中我们对这个功能的实现详细详述。

非阻塞 IO 传输：请求处理系统的线程数是有限的，如果我们采用传统的阻塞型 IO 通过多线程传输文件，那必定会造成请求处理系统的线程源利用率极低。而进程的 IO 类型是操作系统层面上的，所以为解决文件传输过程中阻塞型 IO 引起的性能问题，我们决定使用非阻塞型 IO，以单个线程传输文件，达到比多程序传输文件更快更好的效果。

服务器端

服务器端是整个平台的服务提供模块。如之前所述，它包含一两部分：一是存储集群；二是请求处理。云存储集群主要负责海量数据存储。而来自客户端的请求则是由请求处理系统来响应。下面我们仔细分析：

存储集群：这部分由我们采用 Hadoop 实现，重点分析集群内各个角色与机器的分配，合理布置整体框架，尽最大限度为发挥集群的性能。

数据处理：服务器端搭建好云存储集群后，还必须要有用于监听来自客户端请求

的数据处理模块。对于客户端文件的请求操作，我们将采取不同的请求处理方式 。

实时监控：提升整个服务器集群的可操作性，还必须增加个实时监控。此部分有三个主要方面的工作：1 文件同步情况：监控所有个人用户的文件的同步情况；2 云存储集群的运行情况：对云存储集群的运行发问我们必须有采取相对就的策略，实时对其进行监控，主监控的监控内容有集群的硬件利用率、存储空间的使用率和个人用户数据的存放情况；3 元服务器的运行情况：元服务器是服务器集群中的大脑，对它的监控是不可或缺的，这部分的内容包括系统的硬盘、CPU、内存以及线程并发数等等的使用情况。

分析以上大数据云存储平台中各部分功能模块的详细划分，我们能得出整个系统功能模块的部休划分图如下所示。

3.3 客户端模块设计

我们搭建的以 hadoop 为基础的大数据去存储平台中，最能体现整个平台安全性的无疑是安全客户端模块了。对客户端一级加密机制的提出与实现是这个模块最大的创新点与亮点。用户的个人文件数据在这一层保护下能极大的提升用户数据整个的安全性。而在用户体验性方面，我们运用的数据透明加密技术的思想，既确保了用户数据的安全了，同时又提高了用户体验，可谓是一举两得。另一方面，我们的客户端也是必须要有注册登录功能。只有成功注册并登录的用户才能拥有本大数据存储平台的基本使用权。所有基于以上分析，可得出总结，客户端将包含四个功能模块：文件过滤驱动、注册登录。这些功能模块之间的关系与整个运行操作流程如下所示。

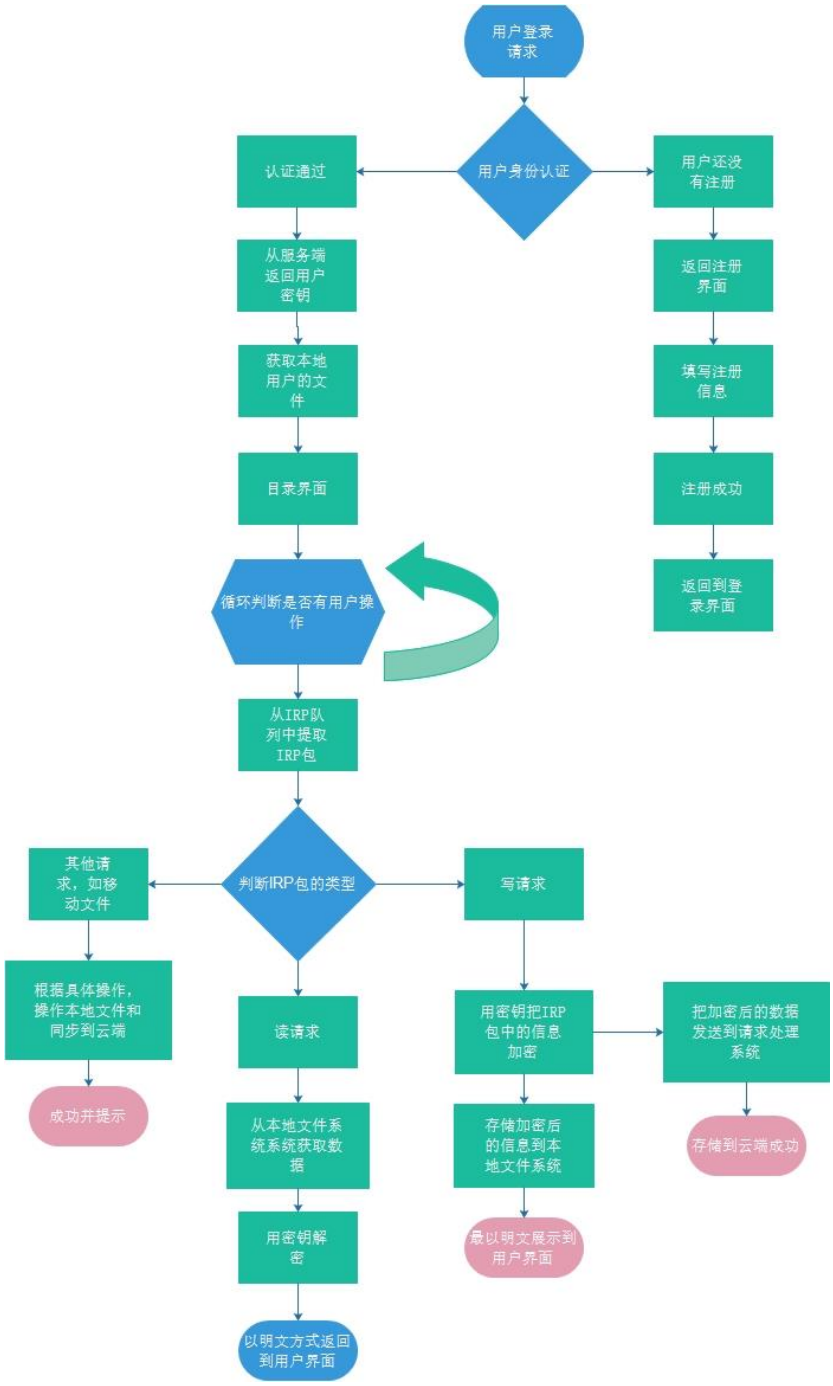


图 3-2 客户端运行流程图

Fig.3-2 Th flow chart of client operation

上图为客户端的原理图，下面我们来仔细分析一下。用户运行客户端然后发起登录请求。此请求信息被请求处理系统的 Shiro 仅限认证机制拦截到，然后 Shiro 通过 Mybatis 框架查询 MySql 数据库，查看此用户是否在 MySql 数据库表中。如果 Mybatis 返回结果集为空，则用户不存在，即此用户还没有注册。此后请求处理系统返回信息，要求客户端跳转到注册界面。如果该用户在 MySql 数据库中，则获取这个用户存入在

MySql 中的密钥，并把此密钥与登录成功信息一并发送到客户端。客户端接收到登录成功信息后，取出 respone 对象用中密钥。之后客户端会不断的查看 IRP 队列，并对不同的包执行不同的操作。如果此包为 read 操作，则使用请求处理系统回返的密钥对包进行解密，再返回到客户端。如果此包是 write 操作，则对此包加密再存储到指定的目录中。如果此包是其他的操作，如移动路径、复制等操作，则直接执行对应操作即可，无需要使用密钥。

经过上面的初步分析，我们对客户端有了清晰的了解。下面我们开始仔细设计。

3.3.1 注册登录模块设计

前面我们已经说过，登录注册界面要越简单越。所以界面的设计风格这里我们不作过多介绍。下面对客户端的逻辑部分仔细说明。

用户可以直接通过注册界面进行注册。当用户发起登录请求后开始阻塞，以等待请求处理系统返回的信息。SSM 请求处理系统查无此用户，会随机生成验证码并要求客户端跳到注册界面。在这个界面里，用户需要填写个人信息，还有验证码。至于密钥，并不需要用户操心，会由请求处理系统随机生成并存储。用户点击注册请求按钮后，会先通过 MD5 算法把登录密码加密，再发送注册请求到请求处理系统。如果请求处理系统返回的是登录成功信息，则直接执行前文原理图中的步骤即可。

有了上面陈述的关于客户端注册登录的流程分析，我们对客户端整个注册登录的流程都有了清晰的了解。在确保这部分功能完成的前提下，还要充分考虑些登录模块与系统其他功能模块的偶合。综合上面的分析，登录的流程如下。

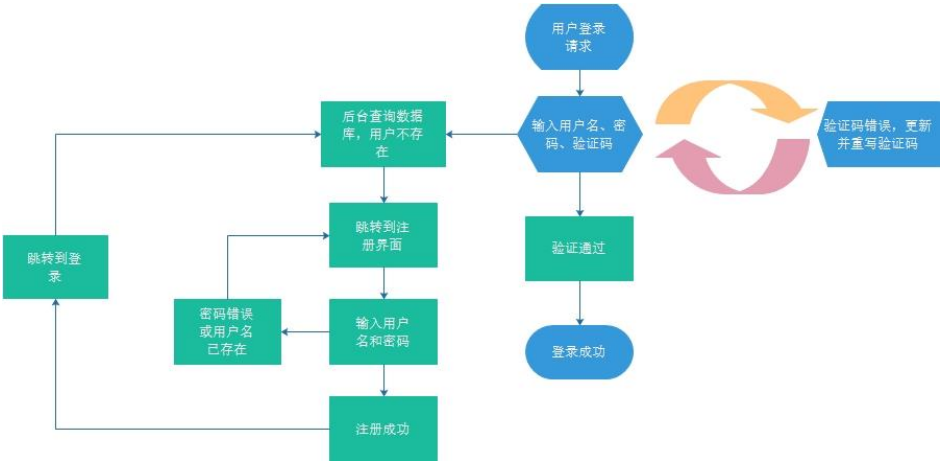


图 3-3 注册登录流程图

Fig.3-3 Th flow chart of register and login

3.3.2 文件系统过滤驱动设计

一级加密机制中的是文件过滤驱动模块。为更一步地确保用户的文件数据在客户端的安全性，我们选择在构建了一层加密解密机制。它的作用域是在操作系统内核层的，其会拦截所有的 I/O 操作，对符合要求的执行指定处理流程，对不符合要求的则释放。

综合之前分析，以下为此模块的操作操作流程图。

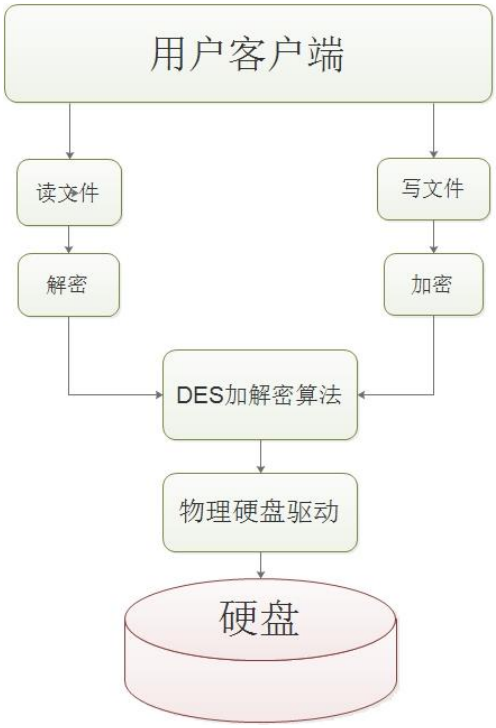


图 3-4 文件过滤驱动操作流程图

Fig.3-4 The flow chart of file filter driver operation

在开始分析上面的流程图前，我们需要讨论一个很重要的问题，那就是到底我们应该加密那些文件呢，换句话说，那就是对说程序来说，需要加密的文件应该具有什么特征？有两种文件的加密逻辑：一种是通过识别不同的后缀名来加密，另一种是不区别后缀名，而是针对指定的进程加密。这两种方式各有利弊。第一种方式，只要识别预先设置的后缀名就可以加解密。优点是实现起来相对第二种要简单一些，而缺点则是存在一定的安全隐患。这种方式若文件被另存为其他类型的后缀的话，就会发生泄漏。举个例子，比如你有一个后缀名为.pdf 的文件，同时模块会加密此后缀名。现在你把这个文件另存为.xxx，则此时存储后的这个.xxx 文件是没有经过加密的，然后你

再把这个.xxx 文件通过 U 盘拷贝到另一台电脑，再修改后缀名为.pdf，那么你就可以看到这个文件的内容了。所以说这种加密逻辑是有一定弊端。而第二种方式相对来说就比第一种要安全很多了，它史针对指定程序加密。只是要指定程序操作文件，都会经过加解密程序，包括中间产生的文件，即如果指定程序想另存为其他类型的文件，则也会加密这个文件。而这种方式也有一定弊端，就是加密难度相对来说要大一点。考虑到安全时，这里我们使用第二种加密逻辑，即只对我们的客户端进程有效。

首先客户端发起请求，然后此模块拦截所有的请求并分析出请求来自那个进程，如果请求来自我们的客户端进程，则进行预先的加密处理，这里我们使用 DES 算法[39]-[42]。接着此模块判断这个请求是什么类型如果是读请求，则调用下一层系统驱动读取数据，并把读到的数据使用 DES 算法解密，再返回给客户端。如果是写请求，则对请求中的数据使用 DES 算法加密，再调用下一层系统驱动存入到硬盘；如果请求是其他的进程，则此模块会直接释放并传递给下一层系统驱动。

3.4 传输模块设计

3.4.1 HTTPS 安全传输设计

由上一章对此模块的分析，能充分掌握其具体的传输原理。从两端的角度分析，可以得出此模块的原理图如下。

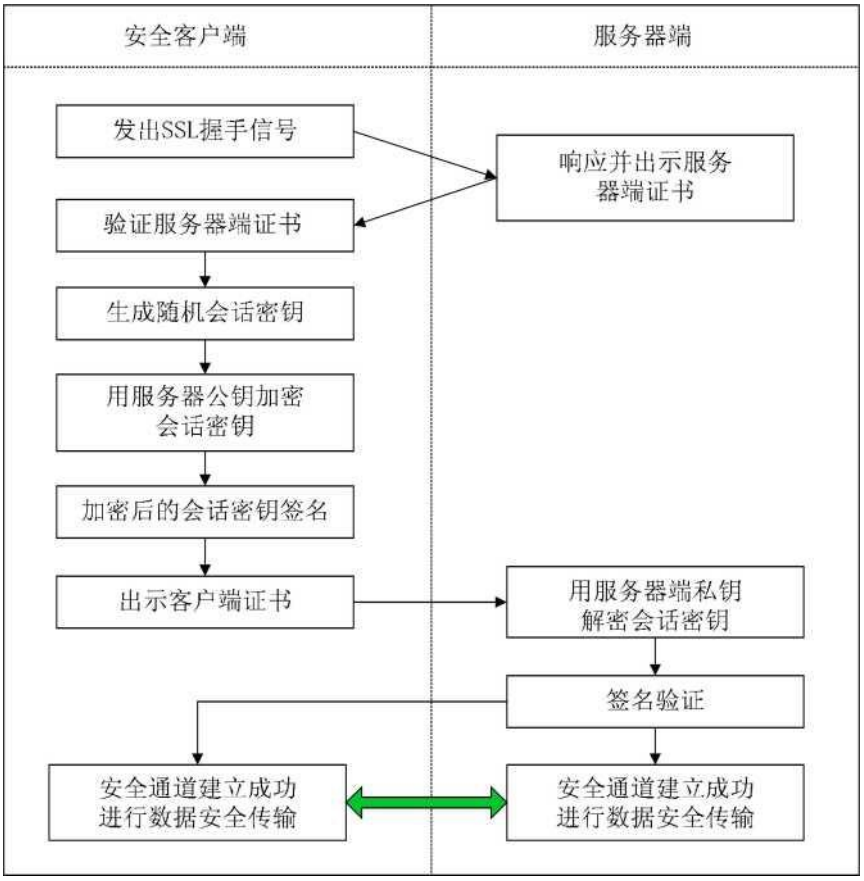


图 3-5 HTTPS 安全传输流程图

Fig.3-5 The flow chart of HTTPS secure transmission

TLS/SSL 协议是一套采用多种加密算法的传输协议。首先加密规则由浏览器发送给服务端。然后，服务端从中筛选加密算法与 HASH 算法，再把证书发送回浏览器。其中证书里包括加密公钥、网站地址以往此证书的颁发机构等等。接着，取得证书后，浏览器执行下列操作：1，检查证书是不是合法的，包括当前访问的网址与证书内的网址是否匹配，颁发机构是不是安全的。2，随后浏览器在接受证书的提前下随机产生一串数字密码并用公钥加密。3，握手信息会采用双方指定的 HASH 算法进行计算，计算结果也用上面的数字密码加密，再把全部信息都发回服务端。之后，服务端收到浏览器发送回的信息执行下列操作：1。对客户端发来的信息，采用私密解密出密码并用其对双方握手的信息执行解密操作和验证服务端 HASH 是否与浏览器发送来的 HASH 相同。2，给浏览器发送一段采用此密码加密的信息。最后，浏览器把信息进行解密。其后再对信息的 HASH 计算，若与服务端的一致，则成功并结束握手过程。之后，所有数据都采用此数字密码加密再通过 HTTP 协议传输。

3.4.2 非阻塞 IO 传输设计

非阻塞 IO 传输模块是本论文的亮点之一，在这个模块设计中，我们是把 Netty 实现非阻塞 IO 的,关于 Netty 的绍介,在第二章的平台关键技术中已经详细阐述过。Netty 是 java 的非阻塞框架，NIO 的底层原理是操作系统级别，其实现原理很复杂，而 Netty 把非阻塞 IO 封装得很完美，其提供了非常简单易用的 API 供开发人员使用。开发人员只需要调用 Netty 里的类与 socket 接口绑定，即要建立非阻塞 IO 的长连接。

在这部分实现是，我们采用 SSM+Netty+HTTPS 框架，即 SSM 框架整合 Netty，建立非阻塞 IO 长连接，并在此基础上采用 HTTPS 安全传输协议实现文件的传输，同时在文件传输过程中使用断点续传技术，边传输边记录断点，以实现一个高性能的非阻塞型 IO 传输模块。下图为非阻塞 IO 传输的流程图

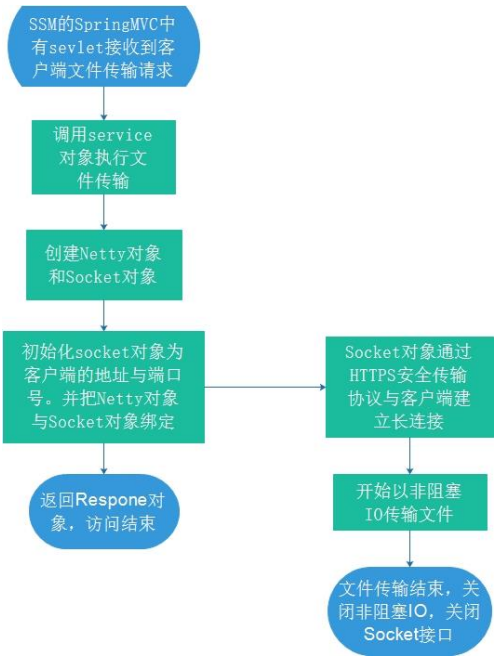


图 3-6 非阻塞 IO 操作流程图

Fig.3-6 The flow chart of operation of non-blocking IO

如上图所示，首先 SSM 请求处理系统的 SpringMVC 中的 sevlet 前端控制器接收到客户端的文件传输请求，并把请求交给 controller 类的相关方法处理。controller 的相关方法会调用 service 对象执行文件传输。在 service 类的实现对象中，其底层会先创建好 Netty 相关对象和 socket 对象。接着再以客户端地址和端口号初始化 socket 对象，并把 Netty 相关对象与 socket 对象绑定。接着再使用 Netty 相关对象与 socket 对象开启新线程执行文件传输后，controller 就可直接返回 respone 对象并访

问此访问了。而在新开启的文件传输线程中，先通过 HTTPS 安全传输协议与客户端建立连接，然后采用 Netty 相关对象创建的非阻塞 IO 线程，其底层会创建多个线程通过 selector 调度器传输文件，并采用断点续传技术边传输边记录断点。最后在文件传输完成后，先后关闭 Netty 相关对象创建的线程和 socket 接口。

3.5 服务器端模块设计

3.5.1 存储集群架构设计

存储集群的架构选择会直接影响整个大数据存储平台的整体性能。鉴于以 Hadoop 所搭建的服务器集群中存在最在安全隐患的部分在于元数据节点服务器，它的损坏或宕机很容易给用户文件数据造成不能恢复的损失。因为在整个服务器后端的集群架构设计过程中，必须重要考虑存在在元数据节点服务器上的数据的备份问题。

此模块我们使用 7 台机器搭建集群。之后，我们还可以继续扩展 Hadoop 云存储集群的规模。本论文主要讨论整个大数据云存储平台的初步探索。我们把云存储集群中的服务器依次编号为 hadoop01、hadoop02、hadoop03、hadoop04、hadoop05、hadoop06、hadoop07，其现在搭建的集群架构如下图所示[51]。

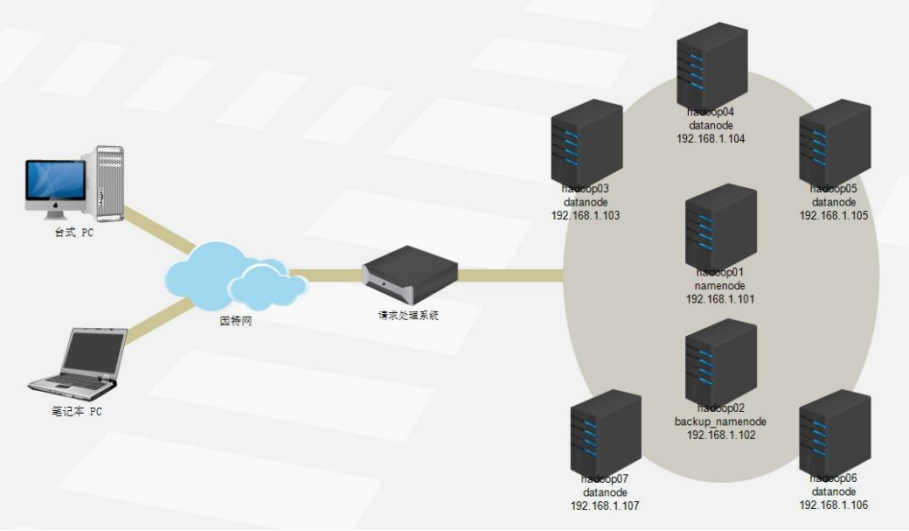


图 3-7 Hadoop 存储集群架构图

Fig.3-7 Hadoop storage cluster architectrue diagram

该集群可分为下面三大部分：

NameNode：负责中元数据存储。它还负责管理整个集群的存储。用来担任

NameNode 的服务器节点应该选用配置最高的机器来担任。这里我们选用 hadoop01 为 NameNode。

DataNode：据实际数据的存储节点，集群中应包含多个 DataNode 数据存储节点。这里我们选用 hadoop03，hadoop04，hadoop05，hadoop06，hadoop07 作为 DataNode 数据存储节点。

BackupNameNode：此节点的存在是为了增强个个 Hadoop 服务器集群的容灾能力。主节点能否正常运行是整个基于 Hadoop 为基础的服务器存储集群的关键所在。为此我们需要对 NameNode 服务器节点建立备份节点。使用 Hadoop 生态中的 Zookeeper 集群，实现 NameNode 与 BackupNameNode 的同步。当 NameNode 出现故障而把工作时，可以通过 Zookeeper[49]实现 BackupNameNode 的快速切换。

3.5.2 请求处理系统设计

这部分模块我们将使用 SSM 框架完成，即当前 IT 行业主流的 SpringMVC+Spring+Mybatis 三件套，另外，还会整合 Shiro 权限认证框架。由于这个模块是整个平台中最重要的部分之一，所以它的性能会决定平台的并发性。通过使用 SSM 框架，我们可以搭建出一个高并发性的请求处理系统，以完成相关逻辑业务和对管理、操作存储集群。这部分的有几个主要功能。第一是处理客户端请求和实现高并发。第二是通过整合 Shiro 框架实现权限认证，提供不同的用户安全等级。第三则是与存储集群的无瑕对接。请求处理系统以 Web 应用的形式存在，内部分为 controll 层、service 层、dao 层，预先在这三层中通过 JDBC 操作存储集群。下图为此模块的功能流程图。

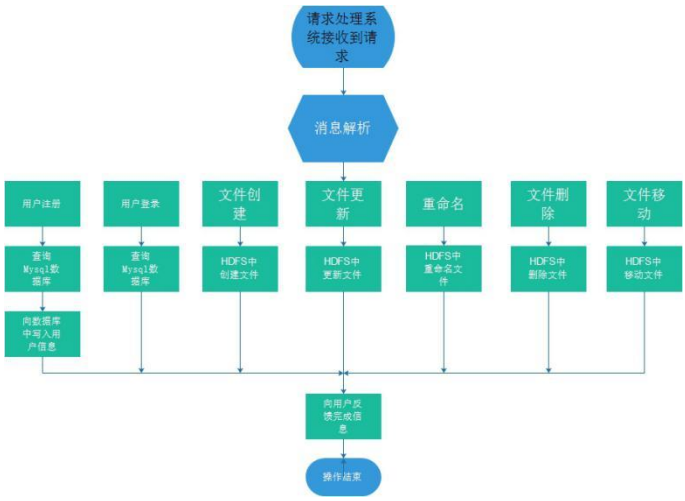


图 3-8 请求处理系统流程图

Fig.3-8 The flow chart of request processing system

3.5.3 实时监控模块设计

在大数据云存储平台中增加实时监控模块可以更进一步提升服务器端的可用性。对 Hadoop 云存储集群、服务器节点性能以及文件同步状况进行实时监控，能够对当前云存储集群的整体运行状况有直观的了解，并对整个平台的维护都带来极大的便利。在实际实现过程中这部分模块主要是由第二方组件实现的，鉴于此，该模块在设计过程中将重点分析其划分与将要采用的实现技术。下面将详细叙述这部分模块的三个主要部分：

Hadoop 监控

在整个大数据云存储平台中最重要最核心的部分就是服务器存储集群。它是整个大数据云存储平台中的数据具体存储的物理介质。这部分主要是对基于 Hadoop 搭建的云存储集群的监控，主要包括各个节点存储、运行状况等等。在上面的章节中已经对个服务器云存储群集进行整个架构设计。

服务器性能监控

上面我们也讨论过，大数据云存储平台的核心部分是基于 Hadoop 的云存储集群。而 NameNode 元数据存储服务器又是 Hadoop 云存储集群的核心服务节点，因为这部分监控功能其实就是对 NameNode 元数据存储服务器节点的监控。基于些，我们对 NameNode 服务器的整体性能和其上的多线程并发状况。

综合以上叙述的三大监控部分，我们继续将这三大监控部分作进一步的细分，以

便于大数据云存储平台的实现，其具体划分情况如下图。

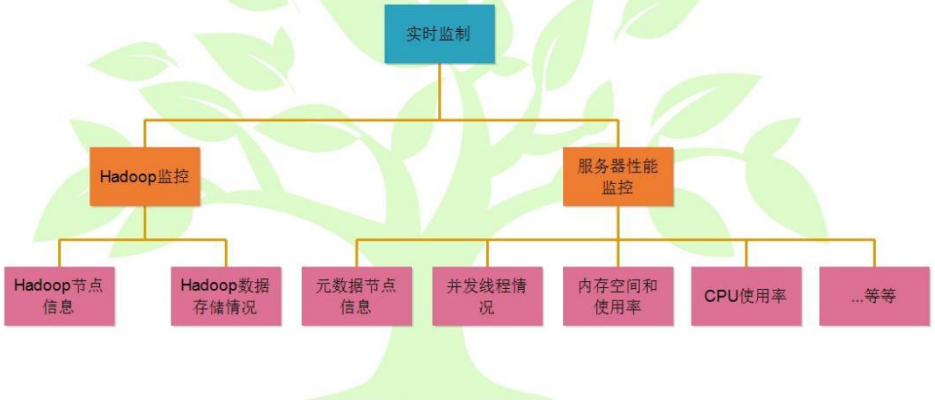


图 3-9 监控模块分层结构图

Fig.3-9 Monitoring module hierarchical structrue diagram

第四章 大数据云存储平台实现

在上面的两章中我们先对大数据云存储平台所要使用的核心技术作了分析，接着对平台的功能需求和性能需求又提出了新的要求以及对大数据云存储平台作了总体架构设计和各功能模块作详细分析设计。在这章中，我们在前两章我基础上对大数据云存储平台的实现进行搭建，主要对核心功能模块的实现方式进行详细陈述。

4.1 平台实现说明

在第二章和第三章中，我们对大数据云存储平台的核心技术进行研究，同时也对平台进行的详尽的设计，所以我们对整个平台的架构和各模块也有了较为深入的掌握。本章我们将进一步对大数据 云存储平台的具体实现 进行阐述，以进一步加深对平台的了解。从层次上来划分，基于 hadoop 所搭建的大数据云存储平台可分为三大模块，分别是安全客户端、安全可靠传输模块和服务器端。所以我们也将从这三个方面对大数据去存储平台的实现作深入分析。

注册登录子模块、文件过滤驱动子模块共同构成了客户端模块。文件实时监控子模块和注册登录子模块都在操作系统的用户态层，属于应用程序方面的，所以我们对这两小部分的实现会使用主使且成熟的 java 语言来实现。而对于文件过滤驱动和虚拟硬盘，鉴于它俩都倾向于操作系统的底层，所以我们将使用更偏向机器层的 C 语言来实现，且这两子模块使用 C 语言实现后再封装成 exe 格式的程序模块，这样就可以供使用 java 实现的文件实时监控子模块和注册登录子模块调用了。

用户的文件数据实际存储在大数据云存储平台的服务器端。服务器端由数据处理子模块、实时监控模块以及 Hadoop 云存储集群三大子功能模块组成。在 hadoop 云存储集群的搭建中，鉴于安全稳定性是我们对此云存储集群再关心的问题，所以我们也将选取 hadoop 的稳定版来搭建我们的云存储集群，我们最终选用 hadoop-0.2.6 版。而数据处理子模块主要是对来自安全客户端的请求进行响应而根据具体的请求进而调用 hadoop 的 API 来操作 hadoop 云存储集群。所以对于数据处理模块的实现我们也将使用 java 语言来实现，同时也选用 SpringMVC 和 Spring 来搭建服务端后台的数据处理子模块。当前 hadoop 被广泛使用的一个重要原因是 hadoop 自带一套完善的监控模块，所以为实现大数据云存储平台的快速开发，我们对实时监控子模块的实现也将采用

Hadoop 自带的控制功能模块；而对负责元数据节点性能的监控，采用 Jconsole 来实现，以进一步降低大数据云存储平台的整体开发难度。

4.2 客户端模块实现

4.2.1 注册登录实现

这部分包括界面设计与相关逻辑实现。安全客户端的注册登录界面，我们将使用 Java Swing 来实现，而对于注册登录的操作流程，我们则编写多个 Java 类相互调用来共同实现。我们在 platform.Cli.register 包实现注册界面，在 platform.Cli.login 包中实现登录界面；而 platform.Cli.action 包中，则主要是结合上面两个 java 包，实现注册登录子模块的操作流程。与此同时，在 cloud.server.main 包中的 ReceiveControl 将响应来自安全客户端的注册登录请求。

RegController 类中的 regPost()方法的执行相关的注册操作。而在 SSM 搭建的请求处理系统中，SpringMVC 的 allRequestsController 控制类来处理请求信息，最后把处理结果返回给安全客户端。

而 LogController 类中的 logPost()的主要用户是提交当前用户的登录信息给服务端进行相关的登录操作。而在请求处理系统，SpringMVC 的 allRequestsController 控制类处理请求信息，最后把结果返回给安全客户端。安全客户端同样也会调用 SendControl 类来接收服务端返回的处理结果信息。如果成功登录，则安全客户端会在本地创建虚拟硬盘，为用户提供相关服务。下图为登录界面。

图 4-1 客户端登录界面截图

Fig.4-1 The screenshot of the client login screen

4.2.2 文件系统过滤驱动实现

客户端二级加密机制中的第二级是文件系统过滤驱动[36]，它是建立在文件系统驱动层的上面。此模块能实现文件的加解密，且对于用户而言，其加解密过程是透明的。同时，在虚拟硬盘之上增加文件系统过滤驱动，可以提升存储在客户端的文件数据的安全性。

在第三章时我们已经分析过，鉴于 Minifilter 可以快速实现该模块的实现，所以在这里我们将基于 Minifilter 的原理来实现文件系统过滤驱动模块。在文件系统过劲驱动模块的程序编写完成后，我们会将它封装成 engine.sys 和 SCrypt.sys 两个系统文件。因为文件系统过滤驱动是安装在操作系统底层的，所以在用户首次使用客户端时，需要先把这两个.sys 文件安装到操作系统之中。与此同时，我们还开发了用作收发中间消息的 S_DrvSendMsg.dll 动态链接库。客户端程序会通过 java Runtime 机制来调用 SCrypt.sys 主程序。下图为文件系统过滤驱动开发完成后所产生的全部文件。

图 4-2 开发完成的文件过滤程序模块截图

Fig.4-2 The screenshot of the filter module developed

对要执行加解密操作的进程，我们能够在加解密中进行配置。当这些进行被配置进加解密选项中，系统过滤驱动以进行透明的加解密处理[38]。下图为文件系统过滤驱动的配置界面图。

图 4-3 加密配置界面截图

Fig.4-3 The screenshot of the encryptedconfiguration interface

4.3 传输模块实现

这个模块的实现包含两方面。一是 HTTPS 网络安全传输连接，它是安全可靠传输模块的安全性的保证。二是非阻塞 IO 断点续传技术，它是安全可靠传输模块的可靠性的保证。在这个模块编程的具体实现中，HTTPS 网络安全传输技术是包含到非阻塞 IO 之中的，即非阻塞 IO 断点续传中使用 HTTPS 协议来传输文件数据。具体实现步骤是先在服务器端与客户端之间采用 HTTPS 安全协议建立起一条安全的通道，然后再在两端使用非阻塞 IO 线程传输多个文件块，同时在文件块传输的过程中还采用断点续传技术，在两端持续记录已经传输完成的文件块。

综合第三章中对安全可靠传输模块的研究，现对上面的部分代码实现作进一步的

说明。

4.3.1 HTTPS 连接实现

我们将采用 SSLSocket 方式来实现 HTTPS 的安全传输。由于 HTTPS 网络安全传输协议已经在当今互联网中得到广泛应用，因为在 java JDK 中早已经对 HTTPS 安全传输协议进行了完善、成熟的封装和实现。因为 HTTPS 安全传输的实现将采用 JDK 的 SSLSocketChannel 类来实现。客户端预先保存有服务器端的安全证书所产生的公钥。HTTPS 安全连接的具体步骤如下描述。首先客户端下载好由服务器端的安全证书产生的公钥。然后客户端使用此公钥把客户端产生的会话密钥 key 进行加密并传输给服务器端。服务器端接收到这个会话密钥 key 后会使用安全证书产生的私钥把 key 进行解密并进行验证。接着再把验证验证结果传输回给客户端。客户端再对验证结果信息进行处理，如果成功，则两端间成功建立起安全的通信信道。于是客户端与服务器端就可以采用多线程断点续传技术，在刚才建立好的通信上传输文件数据了。鉴于 JDK 对 HTTPS 的封装已经很完善，所以这里不再对其进行过多叙述。

4.3.2 非阻塞 IO 及断点续传实现

这部分功能需要在两端采用 Netty 框架实现。在客户端直接导入 Netty 包，把 Netty 与 socket 绑定即可，其具体实现代码在 platform.Cli.tarnsmission 包的 platform.Cli.tarnsmission.postAction 类、platform.Cli.tarnsmission.getAction 类以及 platform.Cli.tarnsmission.Netty.CliTransferHandler 类中。其中 postAction 是上传文件，getAction 是下载文件，这两个类是都通过调用 CliTransferHandler 实现非阻塞 IO 断点续传。

是面是 CliTransfer 类的部分实现代码。

```
Public class CliTransferHandler extends ChannelHandlerAdaper{
@Override
Public void channelRead(ChannelHadnlerContext context,Object message) throws Exception{
try{
```

```
byteBuf buffer = (byteBuf)message
Byte[] filepastion = new byte[buffer.readableBytes()];
Buffer.readBytes(filepastion);
...
}cathch(Exception e){
System.err.println("非阻塞 IO 在下载文件时出错了，具体错误为："+ e);
}finally{
MyUtil.release(message);
}
}
...
...
...
}
```

在请求处理系统端，我们把 Netty 整合到 Spring 中，通过 Spring 容量来管理 Netty 的创建与销毁。具体操作是，把 Netty 的相关类，如 platform.Server.tarnsmission.Netty.ServerTransferHandler，当成是 javabean 类，配置到 spring.xml 文件中。然后在 platform.server.transmission.service 类和 platform.server.transmission.dao 中，调用 platform.server.tarnsmission.Netty.ServerTransferHandler 类，该继续了 ChannelHandlerAdaper，可实现与 CliTransferHandler 类相似的功能。

4.4 服务器端模块实现

4.4.1 云存储集群搭建

服务器端中最重要的子模块无疑是云存储集群的搭建了，它的具体集群结构将直接影响差整个大数据云存储平台的性能，进而影响着用户的体验感。因此云存储集群的搭建过程极其重要。鉴于 Hadoop 有着十分优越的性能和友好的使用方式，同时在当

前互联网有着非常广泛的应用，因而网上关于 Hadoop 的资料也十分多。所以关于云存储集群的技术选取，我们最终决定使用 Hadoop 来部署我们的云存储集群。

关于 Hadoop 云存储集群的搭建，这样我们分成三部分来分析与实现[51]。第一部分是关于 hadoop 集群内的所有服务节点的前期准备的部署。第二部分是关于元数据备份机制的搭建。第三分部才是关于 hadoop 云存储集群的搭建。鉴于在研究大数据云存储平台时的硬件设备资料有限，同时关于云存储集群的搭建准则是最好选用奇数节点进行搭建，所以我们选择采用 7 点机器来搭建我们的云存储集群。另一方面，由于 hadoop 优越的横向扩展性能，所以在以后的平台升级方面可以视情况而进一步横向扩展云存储集群的服务节点数据。

hadoop 集群服务节点前期准备

在这一小节中，我们主要的工作就是在搭建 hadoop 集群前,把所有服务器节点环境都配置好，以便于上面阐述的第二第三步骤的进行。这样我们需要配置的准备工作包括把所有服务节点的 IP 地址都设置成统一标准的静态 IP 地址、关闭所有服务节点的防火墙、在所有服务节点上安装 JDK、在所有服务节点的本地 DNS 服务、在所有服务节点的设置 SSH 免密钥登录。这部分工作比较烦琐，下面我们将一一阐述。

服务节点设置统一标准的静态 IP 地址

要想 hadoop 服务集群之间能相互相识和通信，那么它们在内网的 IP 地址就不可能是动态的、每次开机都重新分配 IP 地址了。具体设置方法是在 Centos7 系统的 /etc/sysconfig/network-scripts/ifcfg-ens33 文件下设置 IPADDR 项，而 GATEWAY 设置为 192.168.1.1，ONBOOT=YES。然后再重启系统即可。下表为所有节点设置 IP 地址详情。

表 4-1 云存储集群内各节点网络设置概况图

Tab.4-1 The profiles of network configuration of cloud storage cluster nodes

机器名称	IP 地址	网关	系统
Hadoop01	192.168.1.101	192.168.1.101	CentOS-7
Hadoop02	192.168.1.102	192.168.1.101	CentOS-7
Hadoop03	192.168.1.103	192.168.1.101	CentOS-7
Hadoop04	192.168.1.104	192.168.1.101	CentOS-7
Hadoop05	192.168.1.105	192.168.1.101	CentOS-7
Hadoop06	192.168.1.106	192.168.1.101	CentOS-7
Hadoop07	192.168.1.107	192.168.1.101	CentOS-7

关闭所有服务节点的防火墙

这里先说明一下，hadoop 服务集群是在公司内部一个单独内网，环境相对来说很安全，同时它受到其所在公司的防火墙的庇护之下，不会受到公司外的网络攻击。另一方面，在服务节点上的防火墙会拦截集群内节点之间的通信，所以这里我们选择关闭所有服务节点的防火墙。具体设置方式为，首先执行命令 systemctl stop firewalld.service 关闭防火墙，然后再执行命令 systemctl disable firewalld.service 让防火墙开机禁止启动。到此服务节点关闭防火墙完成。

在所有服务节点上安装 JDK

几乎只要涉及到编写 java 程序的操作都必须要安装 JDK，所以这部分的安装教程在网上都能找到。所以关于 JDK 的安装就不作过多详述。有兴趣的可在网上查看资料即可。

设置服务节点的本地 DNS 服务

在上面中我们给 7 给 Centos 服务器设置名字为 hadoop01 至 hadoop07。为方便集群内的节点间相互通信，我们将使用系统名而不是 IP 地址来通信。所以我们需要修改 /etc/hosts 文件加入面内容即可。加入的内容如下：

设置服务器无密钥登录

在安装好 hadoop 后，即使在启动时，namenode 进行所在节点就需要通过 SSH 程

序登录到其他节点上启动 datanode 进行等操作了。如果不设置无密钥登录，则 namenode 每程序登录一个节点就需要输入该节点的登录密码，当集群稍为比较大时，其工作量是极大的。所以我们必须要设置服务节点无密钥登录。设置方法如下。先在需要远程登录到其他节点的服务节点上执行 ssh-keygen -t rsa 命令产生自己的公钥和私钥。然后再执行命令 ssh-copy-id 用户名@目标节点机器名，即可把自己的公钥传输到需要登录的服务器上。到此设置无密钥登录完成。

到里我们已经完成了安装 hadoop 前需要的做的所有准备工作了。

元数据备份机制的搭建

在这里我们所指的无数据备份机制可以理解为专指 hadoop 集群中 namenode 服务节点与 backup_namenode 备份节点的元数据同步机制。hadoop 集群的元数据相当于整个数据的目录，而 namenode 负责管理这些目录，其功能相当是 hadoop 的大脑。只有通过 namenode 所管理的这个“目录”，才能为用户在 datanode 集群中找到用户自己的文件[6]。

基于我们在第二章中对 hdfs 文件系统原理的详细分析，有两种方式可备份 namenode 元数据。一是基于 Rsync 通信机制；二是使用 zookeeper 集群作为 namenode 与备份 namenode 之间的监控机制[46]。这里鉴于 zookeeper 是 hadoop 生态系统中的组成部分，而且是 hadoop 生态中的管理员，同时 zookeeper 有着强大的稳定性和可靠性——即使 zookeeper 集群中有多个节点损坏（只有不超过半数节点）都仍然可以工作，所以我们再终采用 zookeeper 集群技术作用元数据备份机制的搭建[47][48]。

由于 zookeeper 需要提供足够的可靠性与稳定性，所以我们选择当前 zookeep 的稳定版本 zookeeper-3.4.10。具体搭建可分为下面所述几步。一下载并解压 zookeeper 到指定文件夹。二更改/etc/profile 文件，把当前 zookeeper 的环境变量追加到 PATH 环境变量中。三配置 zoo.cfg 文件，设置 zookeeper 里的所有节点和通信端口。四新建 myid 文件并设置里面的 id 号。五把 zookeeper 分发到其他服务节点。

通过以上几步就可以成功搭建起 zookeeper 集群。关于 zookeeper 更为详细的搭建过程，可查看网上相关资料。

hadoop 云存储集群搭建

根据上面的阐述，我们将使用 7 台安装有 Centos7 系统的机器来搭建 hadoop 集群。其中我们在 hadoop01 启动 namenode 进程，在 hadoop02 上安装 backup_namenode 进程，

在 hadoop03 至 hadoop07 共 5 台机器上安装 datanode 进程。再结合上一小节关于 zookeeper 集群的安装（zookeeper 将在 hadoop01 至 hadoop05 上共 5 个节点上安装），所以关于整个 hadoop 的集群内进程的整体分配情况如下表所示[6]-[9]。

表 4-2 云存储集群内各节点进程安装概况图

Tab.4-2 The profiles of installation process of cloud storage cluster nodes

应用程序	hadoop01	hadoop02	hadoop03	hadoop04	hadoop05	hadoop06	hadoop07
QuorumPeerMain	√	√	√	√	√		
JournalNode	√	√	√	√	√		
NameNode	√	√					
ZKFC	√	√					
ResourceManager	√	√					
DataNode			√	√	√	√	√
NodeManager			√	√	√	√	√

下面对上面中的相关配置作进一步解析。

QuorumPeerMain：它为 zookeeper 集群相关的进程，这里只要用作 hadoop 的 namenode 的备份机制。

JournalNode：它为 hadoop 集群自动创建的且在 hadoop 上搭建的另一个集群，其主要作用是存储基于第二章所阐述的 HDFS 中的 editlog 文件，使 editlog 文件不再为 namenode 所有，而是存储在 journode 集群中，用于元数据在 namenode 与 backup_namenode 之间的一致性快速同步。

ZKFC：全称为 zookeeper FailOver Controller，即 zookeeper 失效转移控制器，它是与 namenode、backup_namenode 安装在同一机器上的，作用是监控 namenode 进程的存活情况，且与 zookeeper 集群保持心跳通信。一旦 namenode 宕机，zookeeper 集群就可以通过 ZKFC 进程快速设置 backup_namenode 从 standy 状态转换到 active 状态。

Datanode：用户文件数据的实际物理存储节点，它的主要作用就是提供文件的存储功能。关于这个进程的具体使用情况，这里不作过多分析。

下面我们对 hadoop 的具体实现步骤作简要阐述。

第一步，这里我们也使用 hadoop 的稳定版 hadoop-2.6.4，先在/etc/profile 文件的 PATH 环境变量是所加 hadoop 的安装目录，并使/etc/profile 文件生效。

第二步，以下的配置文件都是的 hadoop 安装目录的 conf/hadoop/目录下设置的。先设置 core-site.xml 文件，主要设置 hdfs 的 nameservice 名和 zookeeper 集群。

第三步，配置 hdfs-site.xml 文件，这个配置文件主要是用户对 hdfs 分布式文件系统的配置。这里最重要的一项就是 namenode 的备份配置，由 namenode 与 backup_namenode 共同组成 nameservice，提供 namenode 服务。另外还有 journalnode 集群的配置、备份 namenode 的启动方式等等。

第四步，配置 slaves 文件，设置需要 datanode 进程的节点有那些。这个文件配置比较简单，加入下面内容即可：

hadoop03

hadoop04

hadoop05

hadoop06

hadoop07

第五步，通过 ssh，把 hadoop 安装的整个目录颁发给其他服务节点，并启动即可。

到此，关于大数据云存储平台服务端的 hadoop 云存储集群搭建完成。

4.4.2 请求处理系统的实现

数据处理模块作为整个大数据云存储平台的辅助系统存在，它的主要功能有两个，上面负责处理来自客户端的请求，向下通过 hdfs 提供的 API 负责操作 hadoop 集群。由于我们的数据处理模块需要解决高并发等问题，同时为了能快速实现此模块的实现，所以最终决定使用 SSM 模块来实现我们的数据处理模块，即当前互联网最为流程的 JavaWeb 框架：SpringMVC+Spring+Mybite。

这本小节中，我们的工作主要有三个。一是实现 SSM 平台的搭建。二是实现数据处理模块对来自客户端的对 hadoop 云存储集群操作的请求的处理。三是实现数据处理模块对客户端的注册登录请求的处理。下面对这三个主要开发工作进行实现。

SSM 框架的搭建

关于 SSM 框架的搭建，我们是在 Eclipse 平台平台的实现的[14]。另外，由于 SSM 框架技术已经非常成熟，国内大多数中小型互联网公司都广泛使用 SSM，所以对于 SSM 的详细搭建过程，在这里我们不作过多阐述。下面只对主要的配置文件简要阐述。

配置 springmvc.xml，开打 springmvc 的注解功能。扫描 bigData 包下的所有 java 文件。部分 xml 代码如下。

```
<context:component-scanbase-package="bigData.controller"/>
```

配置 web.xml，设置 springmvc 的前端控制器为 SpringMVC 的 DispathcherServlet。

注册登录请求与 hdfs 存储请求的实现

综合第三章对这两小功能的分析，我们决定把它们放在一起实现。这部分的实现是由 platform.server.controller 包下的 allRequestsController 类实现。该类中的具体代码如下。

```
@Controller
Public class allRequestsController{
OracleService oracleservice = new OracleService();
hdfsService    dhfsservice = new hdfsService()

@RequestMapping(“/register”)
Public ModelAndView register(HttpsServletRequest request) throws Exception{

userName = request.getParameter(“userName ”)
...
boolean flag= oracleservice .register(userName);
if(flag){
...
}else{
...
}
```

```
}

@RequestMapping(“/login”)
Public ModelAndView login(HttpsServletRequest request) throws Exception{
userName = request.getParameter(“userName ”)
passWord = request.getParameter(“password”)
boolean flag= oracleservice .login(userName,password);
if(flag){
...
}else{
...
}
...
}
@RequestMapping(“/hdfsHandler”)
Public ModelAndView hdfsHandler(HttpsServletRequest request) throws Exception{
handlerCode= request.getParameter(“handlerCode”)
switch handlerCode:
case addPath:
...
case deletePath:
...
case removePath:
...
}
....
}
```

上述代码的实现逻辑相对来说比较简单。客户对于注册登录、hdfs 的操作都是根据不同的 url，被 allRequestsController 类的不同方式所拦截。当请求是用户注册时，会

被 register(HttpsServletRequest request)方法所拦截，然后从 request 实现是获取 username。然后再将由 sqlservice 实例对象处理。从中可以看出。关于对 Oracle 数据库的所有操作都是在 OracleService 类中完成的。这个 OracleService 里封装了对 Mybite 框架对 Oracle 数据的所有操作。同时关于 allRequestsController 里的其他方法也是如此分析。

到此，关于数据处理模块的实现基本完成。

4.4.3 实时监控模块实现

我们在第三章中已经对实时监控模块进行的详细的分析，这部分的实现主要包括三个方面的内容。一是实现对整个大数据云存储平台后端的云存储集群的监控，二是整体上对所有用户的文件同步情况作监控，三是对 namenode 所在的服务节点的整体性能进行监控。下面我们就这个三方面进行详细阐述。

云存储集群的监控

在第三章的分析和在第四单的云存储集群搭建中，已经清楚知道，我们的云存储集群是基于 hadoop 技术而实现。而在对 hadoop 的 hdfs 分布式文件系统的监控中，hadoop 有一个天然的优势。那就是 hadoop 内部已经集成了对 hdfs 的实时监控模块，而且是以 web 的方式实现的，可实现以图形化方式对 hadoop 云存储集群的远程监控。下图是关于访问我们搭建的 hadoop 云存储集群 namenode 页面的截图。

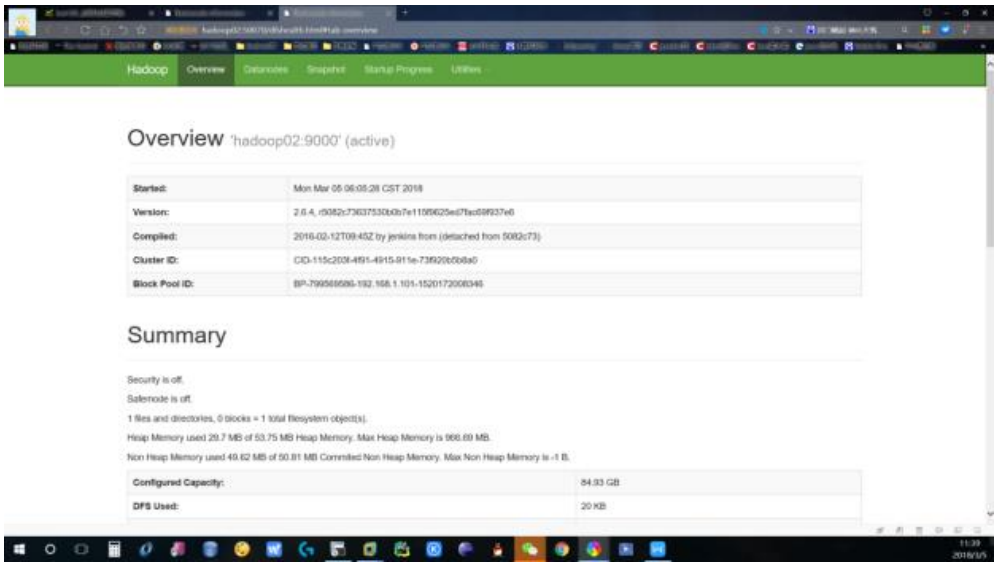


图 4-4 NameNode 的 Active 网页截图

Fig.4-4 The screenshot of the Active web page of NameNode

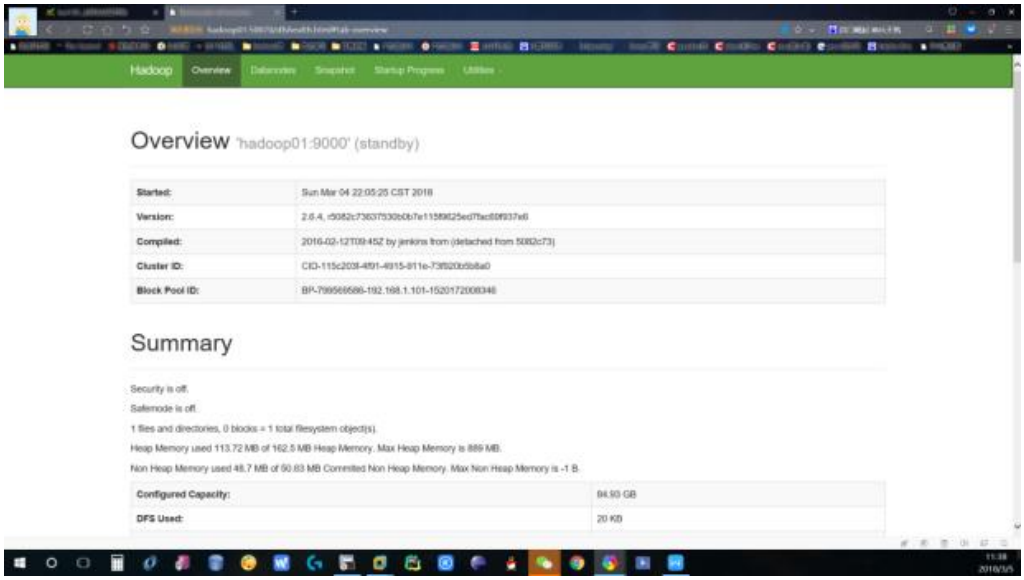


图 4-5 NameNode 的 Standy 网页截图

Fig.4-5 The screenshot of the Standy web page of NameNode

从上图中可以得知关于 Hadoop 的 hdfs 的总体运行信息，包括有 HDFS 的可用空间和空间利用率、当前访问的 namenode 的状态、活跃的 datanode 节点、死亡的 datanode 节点和 namenode 的元数据存储位置等等信息。另外，当我们点击网页上的 Browse the file system 按钮时，还可以查看当前存储在 HDFS 系统上的所有的用户文件，包括文件名、文件类型、文件的使用权限以及文件的副本数等等更为细致的信息。

元数据服务器性能监控

结合第三章的功能与性能分析以及上一小节的分析，我们已经深刻理解到元数据是获取存储在 hdfs 中的文件的索引，而 namenode 节点相当于 hdfs 的大脑。一旦 namenode 服务节点出现任何故障，都会严重影响整个大数据云存储平台的可运行性与性能。因此，在第三章功能与性能需求中我们已经提出，需要给元数据服务节点增加监控机制，以实时掌握元数据所在服务节点的实时性能状况信息，包括内存使用率、线程数据、CPU 利用率等等。

本人之前一直从事 java 相关的开发工作，对 JDK 相对比较了解。同时为实现平台的快速搭建，所以，这里我们 JDK 自带的 JConsole[52]工具来对管理元数据的 namenode 进程所在服务节点进行实时监控。

Jconsole 一个是 JDK 里面的工具，可对进程进行实惠监控。我们先安装 JDK，然后在 JDK/bin 目录下即可找到这个工具。Jconsole 可以对本地 JVM 和远程 JVM 进行实

时监控，也正是由于他提供了对远程 JVM 进行监控的支持，使得 Jconsole 的使用十分夜广泛。开发人员不用在需要被监控的机器上安装任何程序，只需要有一个安装在 JDK 的 PC 并与被监控机器连接在同一内网，即可实现对指定机器上的进行实现实时监控，使用起来非常方便。此外，Jconsole 更为强大的一点是，它还提供可视化的 GUI 界面，对被监控的进程画出大里的图表，为开发人员提供更为直观的监控。

下面是关于 Jconsole 工具对 namenoce 进程进行监控的阐述。

首先我们安装好 JDK 后，在命令行中输入 jconsole 命令即可运行连接界面（在 windows 和 linux 系统均可）。连接界面如下图所示。

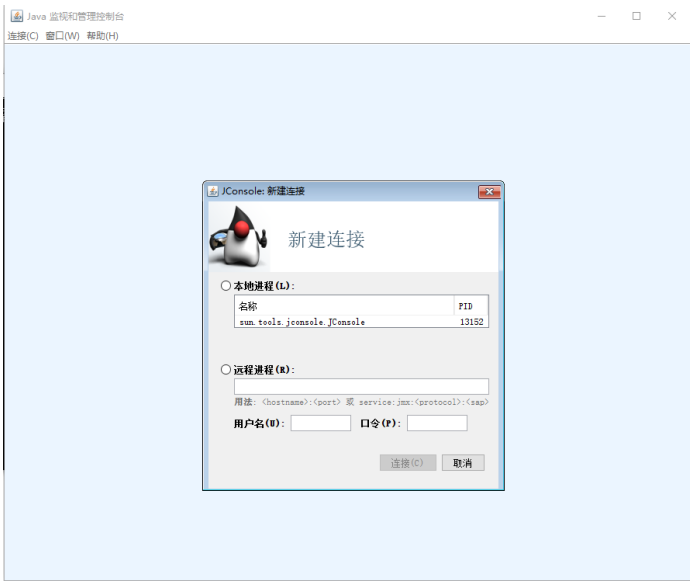


图 4-6 Jconsole 连接截图

Fig.4-6 The screenshot of Jconsole connection

然后在远程进程栏下输入 namenode 的 IP 地址和端口号，再输入用户名和登录密码，即可实现对元数据所在节点进行监控。下图为 namenode 所在服务节点的 namenode 进行的具体监控状况。

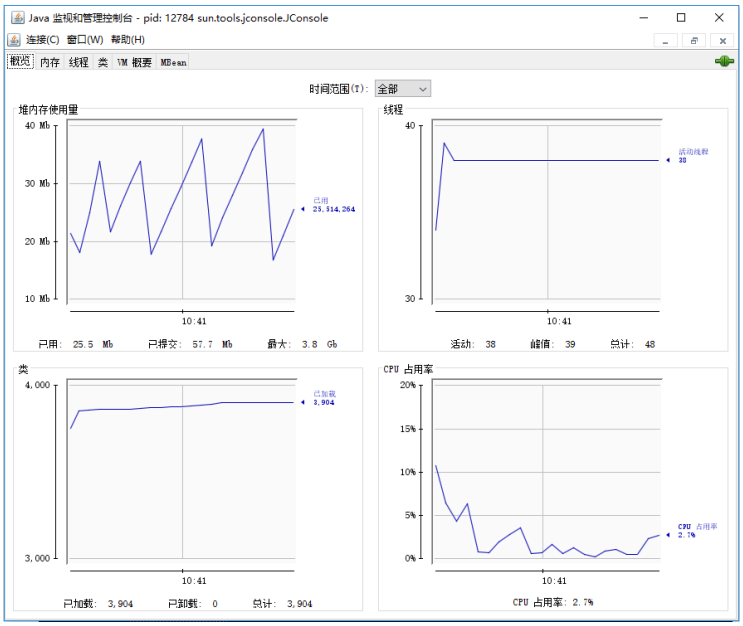


图 4-7 Jconsole 整体监控截图

Fig.4-7 The screenshot of Jconsole voerall monitoring

从上图可以看出，可以进行实时监控的信息有堆内存使用量、线程数、已经加载 java 类的数据、CPU 占用率等多方面，同时还可以的具体的指定方面作进一步的查看。下图为对内存中的非堆内存使用量的详细状况。

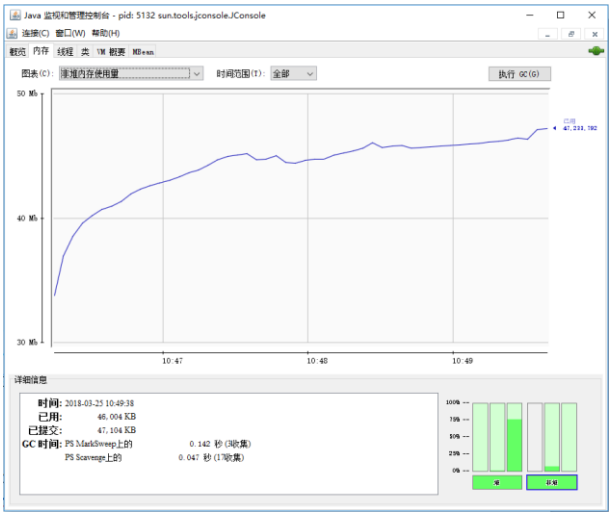


图 4-8 Jconsole 非堆内存使用量详细监控截图

Fig.4-8 The screenshot of Jconsole’s non-heap memory usage details

文件同步情况监控

在实时监控模块的实现中，云存储集群的整体监控与 namenode 进程的监控都是通过 hadoop 自带的或 JDK 自动的工具进行实现的。所以上两步的开发量与开发难度都比较小。而在对文件的同步情况进行监控中，我们需要自己编写程序来实现这一部分的

监控。鉴于文件同步是在应用层而不是在内核层，同时对文件的监控也就是在多线程传输的基础上对同步信息进行抓取，所以这一监控部分的实现我们将使用 Java Swing 来实现。

Java Swing 是一套专门为 java 开发的 GUI 图形界面工具包，包括文本框、分隔空格、按钮和表等等控件[43]-[45]。我们使用 Swing 对当前正在进行同步操作的用户信息和对应用户的同步文件信息通过图形界面呈现出来即可。在 cloud.download.manager.syncSwing 包下的 syncSwingManager 类来实现的。该类通过不断的在多线程传输模块中抓取用户同步数据，把整体同步状况呈现出来。下图为使用 Swing 开发的图形界面。

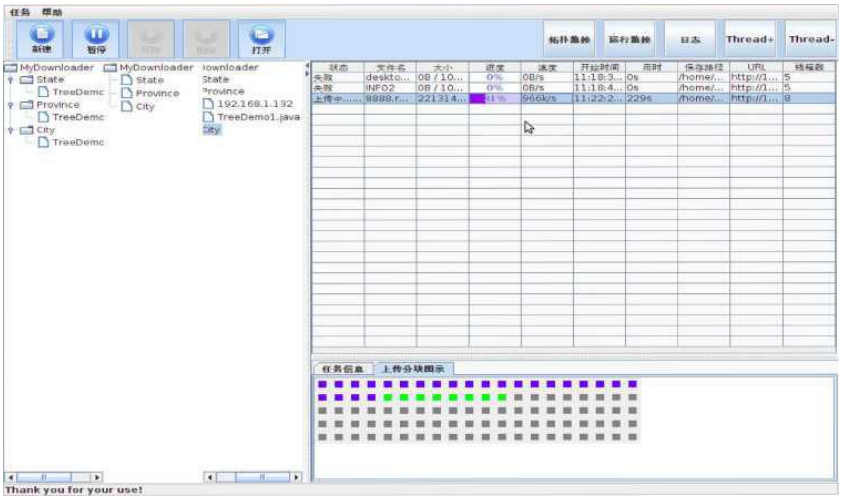


图 4-9 文件同步详情截图

Fig.4-9 The screenshot of the file synchronixation details

4.5 本章小结

在本章中我们根据第三章中对大数据云存储平台的功能与性能需求分析的基础上，对大数据云存储平台的实现从三个主要方面分别进行，包括客户端模块、安全传输模块以及服务器端。同时在这三大主要模块的实现过程中，又对其作细分和实现。在客户端模块中主要包括注册登录、文件过滤驱动、虚拟硬盘以及文件同步的实现。在安全可靠传输模块方面重点阐述了断点续传、https、虚拟硬盘技术的使用与实现。在服务器端则作了更为详细的详述，重点使用了当前互联网行业主流的两大技术，分别是基于 SpringMVC+Spring+Mybite 的 SSM 后台框架搭建的数据处理模块，和基于 Hadoop 大数据技术搭建的分布式云存储集群，最后还对服务器端的监控作了简要概括

和分析。

第五章 大数据云存储平台的性能测试

经过第三章的功能、性能分析与平台整体设计和第四章的平台实现，我们成功搭建了大数据云存储平台。在这章中我们将对此大数据云存储平台进行相关测试与分析工作。

5.1 文件透明加密的测试

在第二章和第三章中我们详细研究文件透明加密技术，即我们使用的文件过滤驱动模块。客户端的第二层加密，其极大的提升了客户端的安全性与可靠性。即使其他要没经过我们同意的情况下，获得了我们电脑的使用权，但由于他没有我们的账号密码和密钥，所以他人是没办法查看到我们存储在客户端本地的文件的。这里我们测试的内容是，分别通过使用客户端直接打开查看我们的文件，与直接在虚拟硬盘里打开查看我们的文件，测试文件的内容。

我们分别以两种不同方式打开存储在本地的文件，测试结果如下面的截图所示。

图 5-1 客户端文件明文展示截图

Fig.5-1 The screenshot of document of client display clearly

图 5-2 客户端文件密文展示截图

Fig.5-2 The screenshot of document of client display in cipher

从上面的测试结果如果，我们直接打开虚拟硬盘中的文件出现乱码，无法查看到文件内容。所以，文件透明加密功能模块测试成功。

5.2 用户文件上传和下载的测试与分析

本论文所设计的大数据云存储平台是基于 Hadoop 而实现的，我们的服务器端包括数据处理后端与云存储集群，且它们是处于同一个内网中的，共同构成了大数据云存储平台的服务端[50]。本论文中设计和实现的物理平台采用的局域网带宽为 50Mb/s，从传输的理论计算，实际理论值可以达到 6.25Mb/s。本论文为了测试对比实际的上传与下载速度，所以我我们分别用文件小为 1 到 10G 的这 10 个文件来对平台进行测试，结果如下表所示。

表 5-1 文件上传与下载速度比较结果

Tab.5-1 The results of file upload and download speed comparison

	1G	2G	3G	4G	5G	6G	7G	8G	9G	10G
上传时间（单位:分）	10	19	30	46	58	88	90	103	110	140
下载时间（单位:分）	13	25	29	58	61	69	82	106	121	130

从上面的列表中的数据可以看出，此大数据云存储平台上传和下载的速度基本相同，大概每 GB 的数据量需要 10 分钟的上传或下载时间。当随着文件数据量大小逐渐

增大，每 GB 数据量所需要的时间则越来越大。分析出现上述现象的原因可知，当文件数据里越大，其理论上需要的上传或下载时间则越长，所以文件在网络上传输的过程中出现网络故障的可能性也会随之呈线性增加，如现在网络阻塞、网络信号不稳定、数据帧丢失等等网络原因。基于第三章我们已经详细讨论过，我们在文件数据传输模块使用了多线程及断点续传技术。当出现网络故障使当前由断点续传技术所划分出来的数据块在网络中中断传输，则文件传输模块会使用断点续传技术重传该数据块。这就引起了上面出现的现象，文件越大，其需要的传输时间会比理论值越大。另一方面，我们同时在文件数据传输模块使用了多线程，所以大大提升了文件数据的传输速度以及数据块重传速率。

综合第三章以及上面的分析，我们得出结论，在用户的文件上传与下载的测试，符合第三章中提到的性能需求和可靠性需求。

5.3 用户登录的响应性测试

响应测试包含用户文件同步的响应与云存储系统登录的等性能测试。只有达到一定的性能指标，才能说明我们的大数据云存储平台可以在实际中应用。我们先对平台的系统登录响应时间进行测试。在测试中我们用 10 个不同的用户进行登录测试。具体测试结果如下表所示。

表 5-2 客户端登录时间与成功率结果

Tab.5-2 The results of clients' logining time and successful rate

用户登录账号	登录时间（秒）	第二次登录时间（秒）	是否达标
BigData_user_1	2	--	是
BigData_user_2	2.1	--	是
BigData_user_3	3.4	--	是
BigData_user_4	登录超时	2.2	--
BigData_user_5	2.5	--	是
BigData_user_6	4	--	是
BigData_user_7	3.3	--	是
BigData_user_8	3.6	--	是
BigData_user_9	4.2	--	否
BigData_user_10	3.0	--	是

从上面的列表可以看出，第四次测试用户出现了一次登录超时，系统并弹出了登录超时提示，而其他九次测试都可以实现一次登录成功。而在这九次测试中，只有一次测试的登录时间是超过 4s 的。所以从上面的测试来说，成功率>=80%，我们的登录响应性能是符合预期需求的。

5.4 文件的同步性能测试

我们的大数据云平台在客户端和服务端系统都有同步监控功能模块，当用户在客户端对虚拟硬盘进行更改后，能马上被客户端监控到，接着采取多线程以及断点续传技术实现与云存储集群的同步。这次测试中，我们进行 20 次测试，其中 10 次测试中在客户端修改不同大小的文件；另外 10 次是在新终端上登录我们的账号，然后在客户端上查看 10 个不同大小的文件，测试其同步情况。

以下为我们在客户端修改用户文件后，客户端监控模块的测试情况。

表 5-3 客户端监控模块测试结果

Tab.5-3 The results of client monitoring module test

	1G	2G	3G	4G	5G	6G	7G	8G	9G	10G
客户端监控模块响应时间（单位：毫秒）	0.5	0.7	0.1	0.3	0.5	0.8	0.9	0.1	1	0.4
同步时间（单位：分）	5	8.7	12.9	16.4	24	33	47	60	90	100
同步速率（单位：Mb/s）	3.4	3.9	3.9	4.1	3.5	3.1	2.5	2.3	1.7	1.7

从上表可以得出，客户端的文件监控模块成功监控到虚拟硬盘文件的修改，同时其响应时间都在 1 秒钟以内，其响应性能优于我们在第三章时的响应性能需求。

以下是我们在客户端登录一个新账号后，读取服务端里的 10 个文件，测试得到的结果。

表 5-4 客户端读取文件测试结果

Tab.5-3 The results of client read file test

	1G	2G	3G	4G	5G	6G	7G	8G	9G	10G
服务端系统监控模块响应时间（单位：毫秒）	2	3	2.2	3.1	1.5	4.2	3.8	1.8	3.7	4
同步时间（单位：分）	8	14.7	18.9	25.4	30	46	66	65	103	121
同步速率（单位：Mb/s）	2.1	2.3	2.7	2.7	2.8	2.2	1.8	2.1	1.5	1.4

总结以上两个列表的 20 个测试结果，可得客户端与服务器端的文件监控模块成功启动并运行，并且其整体的上下行速率都在 2Mb/s，性能达利我们的需求。同时我们也看到，不仅是在监控同步方面，同时在之前的文件上传、下载方面，其性能都与当前网络状况有关，这部分因素不是本论文要研究的范围之内，所以从整个来看，所有的测试都基本达到我们的需求。

第六章 总结与展望

6.1 总结

本论文重点研究以Hadoop集群相关系统以及MVC框架的特性,然后基于Hadoop为后端去存储框架设计并实现的一个自动化、可监控的大规模云存储平台。我们所设计并实现的大数据去存储平台在初步经过测试后，整体功能基本完成，整体性能也表现出良好的稳定性和可用性。本论文主要研究与开发包括下几所列的几个方面：

研究了Https网络安全传输协的原理及其与Http网络传输议的区别，还有多线程与断点续传技术，作为这个项目的安全可靠传输的理论基础。

研究了虚拟硬盘技术和透明加解密技术中的文件过滤驱动技术，以为实现一个安全可靠的客户端软件。

重点分析了大规模Hadoop集群的划分方案以及其详细的搭建过程，并在第三单中以较大篇幅详细叙述了Hadoop集群的搭建过程

各个功能模块都实现后，在真实的环境中进行多次测试，并测试结果能满足实际需求，表现出良好的可用性。

6.2 进一步工作展望

自二十一世纪以后，计算机技术超高速发展，同时也直接带动了大数据领域相关技术的蓬勃发展。现在以Hadoop生态体系为大数据技术中心，Hadoop开源社区也是益活跃，Hadoop体系也在大数据的发展长河中演进。与此同时，越来越多的大数据相关的成果不断地涌现出来。因为大数据云存储平台的进一步工作主要有：

在高并发方面，我们还没有对平台的高并发作出应对措施。因为在接下来的平台版本升级中，可能加入nginx以实现平台的负载均衡，以削减使用高峰时的并发量。

在整个平台的容灾与日常维护方面，还需要对这方面作进一步加强，以增加大数据云存储平的健壮性，如使用kafka中间件对数据作缓存、使用redis内存数据库提升用户的注册、登录速度和用户体验、使用Flume把系统的所有日志都沉到Hadoop中等等。这样改进，其目的是为了设计对一个能应对多个环境的大数据云存储平台。

参考文献

[1] 翟永东. Hadoop分布式文件系统(HDFS)可靠性的研究与优化[D]. 华中科技大学, 2011.

[2] 王跃. 基于Hadoop分布式文件系统的分析与研究[J]. 计算机光盘软件与应用, 2011(9):161-162.

[3] 周丹. 基于paxos算法的Hadoop分布式文件系统高可用性探究[J]. 电子测试, 2014(s1):24-26.

[4] 郎为民, 陈凯, 赵旭,等. 大数据中心云存储安全研究现状[J]. 电信快报, 2015(12):7-12.

[5] 缪璐瑶. Hadoop安全机制研究[D]. 南京邮电大学, 2015.

[6] 王峰, 雷葆华. Hadoop分布式文件系统的模型分析[J]. 电信科学, 2010, 26(12):95-99.

[7] Shvachko K, Kuang H, Radia S, et al. The Hadoop Distributed File System[C]// IEEE, Symposium on MASS Storage Systems and Technologies. IEEE Computer Society, 2010:1-10.

[8] White T, Cutting D. Hadoop : the definitive guide[J]. O'reilly Media Inc Gravenstein Highway North, 2012, 215(11):1 - 4.

[9] Xiao L I, Ren W. Research on SpringMVC-based Multi-Platform J2EE Development[J]. Journal of Jilin University, 2017.

[10] 胡启敏, 薛锦云, 钟林辉. 基于Spring框架的轻量级J2EE架构与应用[J]. 计算机工程与应用, 2008, 44(5):115-118.

[11] Ho C. Using MyBatis in Spring[M]// Pro Spring 3. 2012:397-435.

[12] 吉朝明, 汪松, 权全,等. 基于Spring MVC、Apache Shiro、MyBatis框架整合的代码生成器:, CN 105824619 A[P]. 2016.

[13] 李帅力. 基于SpringMVC及MyBatis框架的智慧园区访客管理系统的设计与实现[D]. 浙江工业大学, 2016.

[14] Xiao L I, Ren W. Research on SpringMVC-based Multi-Platform J2EE Development[J]. Journal of Jilin University, 2017.

[15] 荣艳冬. 关于Mybatis持久层框架的应用研究[J]. 信息安全与技术, 2015, 6(12):86-88.

[16] Ho C. Using MyBatis in Spring[J]. 2012.

[17] 张艳军, 王剑, 叶晓平,等. 基于Netty框架的高性能RPC通信系统的设计与实现[J]. 工业控制计算机, 2016, 29(5):11-12.

[18] 范华峰. 一种基于Netty框架的网络应用服务器设计方法[J]. 福建电脑, 2015(10):33-34.

[19] 肖凯. 基于Reactor模式的Muduo网络框架技术研究[D]. 武汉邮电科学研究院, 2016.

[20] Yang J, Zhang H, Han L, et al. Design and implementation of software consistency detection system based on Netty framework[J]. 2016.

[21] 黄经赢. 基于Shiro框架的细粒度权限控制系统的设计与实现[J]. 广东技术师范学院学报, 2013, 34(7):20-23.

[22] 加尔肯&#;胡孜哈依尔. 基于Shiro框架的授权机制设计与实现[J]. 科学导报, 2015(8).

[23] 徐孝成. 基于Shiro的Web应用安全框架的设计与实现[J]. 电脑知识与技术, 2015(16):93-95.

[24] 魏兴国. HTTP和HTTPS协议安全性分析[J]. 程序员, 2007(7):53-55.

[25] 徐识溥, 陈建林, 赵京音,等. HTTPS协议在单点登录系统的应用[J]. 微型电脑应用, 2012, 28(1):40-42.

[26] 唐屹, 王志双. 网银HTTPS协议的配置状况研究[J]. 信息网络安全, 2017(1):16-22.

[27] 茹惠素. 基于HTTPS协议的统一登录系统设计与实现[J]. 浙江工业大学学报, 2008, 36(5):527-530.

[28] Xu S, Chen J, Zhao J, et al. The Application of HTTPS Protocol on Single Sign-on System[J]. Microcomputer Applications, 2012.

[29] 李凡, 刘学照, 卢安,等. WindowsNT内核下文件系统过滤驱动程序开发[J]. 华中科技大学学报:自然科学版, 2003, 31(1):19-21.

[30] 邵昱, 萧蕴诗. 基于文件系统过滤驱动器的加密软件设计[J]. 计算机应用,

2005, 25(5):1151-1152.

[31] 郑磊, 马兆丰, 顾明. 基于文件系统过滤驱动的安全增强型加密系统技术研究[J]. 小型微型计算机系统, 2007, 28(7):1181-1184.

[32] 沈玮, 王雷, 陈佳捷. 基于文件系统过滤驱动的加密系统设计与实现[J]. 计算机工程, 2009, 35(20):157-159.

[33] 董亮卫, 黄鹂声, 张文婧,等. Windows NT文件系统过滤驱动程序在信息安全中的应用[J]. 信息技术, 2005(10):135-138.

[34] 李民. 基于Windows文件系统过滤驱动的文件加/解密技术研究与实现[D]. 四川大学, 2006.

[35] Zheng L, Zhao-Feng M A, Ming G U. Techniques of File System Filter Driver-based and Security-enhanced Encryption System[J]. Journal of Chinese Computer Systems, 2007, 28(7):1181-1184.

[36] Shao Y, Xiao Y S. Design of encryption software based on file system filter driver[J]. Computer Applications, 2005, 25(05):1151-1152.

[37] 兰旭辉, 熊家军, 邓刚. 基于MySQL的应用程序设计[J]. 计算机工程与设计, 2004, 25(3):442-443.

[38] Widenius M. Mysql Reference Manual[M]. O'Reilly & Associates, Inc. 2002.

[39] 管莹, 敬茂华. DES算法原理及实现[J]. 电脑编程技巧与维护, 2009(4):5-7.

[40] 李少芳. DES算法加密过程的探讨[J]. 计算机与现代化, 2006(8):102-104.

[41] 胡美燕, 刘然慧. DES算法安全性的分析与研究[J]. 内蒙古大学学报(自然版), 2005, 36(6):693-697.

[42] Saputra H, Vijaykrishnan N, Kandemir M, et al. Masking the Energy Behavior of DES Encryption[C]// Design, Automation and Test in Europe Conference and Exhibition. IEEE, 2003:84-89.

[43] 王晓哲. Java Swing组件技术[J]. 天津职业院校联合学报, 2008, 10(3):138-142.

[44] 裴龙, 何大可. Java2 Swing组件设计模式分析[J]. 计算机应用, 2001, 21(z1):274-275.

[45] [Etal M L . . Java Swing[M]. 清华大学出版社, 2004.

[46] Junqueira F P, Reed B C. The life and times of a zookeeper[C]// ACM Symposium on Principles of Distributed Computing. ACM, 2009:4-4.

[47] 李文韵, 崔毅东. 基于ZooKeeper的集群节点管理方案的设计与实现[J]. 2013.

[48] 袁子淇. 基于ZooKeeper的集群应用配置管理的设计与实现[D]. 内蒙古大学, 2015.

[49] Junqueira F, Reed B. ZooKeeper: Distributed Process Coordination[M]. O'Reilly Media, Inc. 2013.

[50] 崔文斌, 牟少敏, 王云诚,等. Hadoop大数据平台的搭建与测试[J]. 山东农业大学学报(自然科学版), 2013, 44(4):550-555.

[51] 淡武强. 搭建Hadoop实验平台[J]. 网络安全和信息化, 2016(5):42-46.

[52] Tardif S. JConsole and JDK version[J].

攻读学位期间的科研成果

论文

[1] 袁斯烺. 浅谈 Hadoop 与 SSM 的大数据云存储平台的研究与实现[J]. 电子世界, 2018(7).

学位论文独创性声明

本人郑重声明：所呈交的学位论文是我个人在导师的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。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。本人依法享有和承担由此论文所产生的权利和责任。

论文作者签名：袁斯煜　　日期：2018.5.29

学位论文版权使用授权声明

本学位论文作者完全了解学校有关保存、使用学位论文的规定，同意授权广东工业大学保留并向国家有关部门或机构送交该论文的印刷本和电子版本，允许该论文被查阅和借阅。同意授权广东工业大学可以将本学位论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印、扫描或数字化等其他复制手段保存和汇编本学位论文。保密论文在解密后遵守此规定。

论文作者签名：袁斯煜　　日期：2018.5.29

教师签名：　　日期：2018.6.5

致谢

转眼间，我已经在大学里度过了三年了研究生美好时光，在即将离开之际，回忆起过往三年的点点滴滴。很兴幸当初能来到这个实验室，这是一个很有人情味的实验室。在学姐学长的帮助和指点下，我开始了自己的求学生涯，同时能增长的我的人生阅历。虽然在这个过程中，也遇到了很多困难，自己也迷惘过，但也正是这些经历丰富了我的人生。在此我真的很感谢在这三年来，一直对我不离不弃的、给我支持、关心的同学和老师。

在这里，我非常感谢一直给支持和帮助的谢胜利教授和吴宗泽教授，正是因为有他们的教育，才成就了今天的我。谢教授和吴教授一直以来为真务实的态度与崇高的职业观和不辞劳苦的工作作风深深地影响着我。在谢团队这如此优良的学习环境和自由开放的氛围下，培养了我们先进和思维方式和敢于担当的勇气，这些都必让我终身受益。

在 408 实验室这个大家庭时，研一时的我们从初识到相逢，并成为一群实习、生活上互助友好的伙伴，曾经一起上课、一起讨论试题、一起出去小聚。随着必修课与选修课的完结，我们有了更多在实验室动手实践项目的时间，一起研究项目、开会总结，有过成功的喜悦，也有过项目难点的苦楚。转眼研三的我们，一起求职面试、讨论日后工作，伴随着论文的结束，象征着我们硕士生涯的完结，即将进入人生的第二个十字路口，迈进社会、寻找适合自己的发展方向。在此，我要向陪我一起度过研究生三年的各位导师、同学、朋友和亲人们表达最真诚的感谢。

感谢我曾在的研究小组所有成员，有已经毕业的师兄师姐，有新加入的师弟师妹，更有同级的兄弟伙伴，是大家的付出与努力、责任心与信心同筑，共同将项目组的任务不断的向前推进。同时，感谢 408 实验室的师兄弟姐妹们，我们都是 408 大家庭的成员，感谢一路有你们，期待我们能一起前行。

最后，还要感谢我至爱的亲人，一路有你们在背后的默默支持，给予了我勇往直前、永不放弃的动力。