学 校 代 码 10459
学号或申请号
密 级
论 文 编 号 036115

郑州大学

专业硕士学位论文

医学影像云存储系统的设计与实现

作 者 姓 名:
导 师 姓 名:
专 业 名 称: 工程硕士
培 养 院 系: 产业技术研究院
完 成 时 间: 2019 年 4 月

A thesis submitted to

Zhengzhou University

for the degree of Master

Design And Implementation Of Medical Image Cloud Storage System

By

Supervisor:

Master of Engineering

Industrial Technology Research Institute

April 2019

学位论文原创性声明

本人郑重声明：所呈交的学位论文，是本人在导师的指导下，独立进行研究所取得的成果。除文中已经注明引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写过的科研成果。对本文的研究作出重要贡献的个人和集体，均已在文中以明确方式标明。本声明的法律责任由本人承担。

学位论文作者：

日期：　　年　　月　　日

学位论文使用授权声明

本人在导师指导下完成的论文及相关的职务作品，知识产权归属郑州大学。根据郑州大学有关保留、使用学位论文的规定，同意学校保留或向国家有关部门或机构送交论文的复印件和电子版，允许论文被查阅和借阅；本人授权郑州大学可以将本学位论文的全部或部分编入有关数据库进行检索，可以采用影印、缩印或者其他复制手段保存论文和汇编本学位论文。本人离校后发表、使用学位论文或与该学位论文直接相关的学术论文或成果时，第一署名单位仍然为郑州大学。保密论文在解密后应遵守此规定。

学位论文作者：

日期：　　年　　月　　日

摘要

在国家政策积极引导下，加快落实分级诊疗制度，本论文为解决远程诊断过程中,医学影像文件“传输难、存储难”的问题，设计并实现了医学影像云存储系统。“传输难”指医学影像文件数量多，文件小，在远程传输方面，系统会面临系统响应时间长，文件易丢失的问题。“存储难”是因为每天都会产生海量的医学影像文件，随着时间的推移，系统会面临扩容难，文件损坏甚至丢失的问题。系统采用“一个中心，两个点”的网络架构，实现基层医院把存放在前置机上的检查文件上传到数据中心，然后上级医院实时下载的基本功能。系统的数据中心是以 Docker 容器为基础的新型 PaaS 云平台。在此基础上，本论本完成的主要工作如下：

1、在文件存储方面，我们采用分布式文件系统 HDFS、分布式数据库 HBase 和分布式协调服务 Zookeeper 三个组件搭建一个高可用、易扩容、高容错的数据存储服务。数据存储服务采用“活动-备用”的模式，避免出现单点实效的问题。并且通过合理地配置，数据存储服务具有一定的容错能力。

2、针对医学影像文件传输难的问题，远程传输模块采用 C/S 架构，融入了 Spring 框架，Mybatis 框架和 Phoenix 框架。把基层医院的文件压缩为 ZIP 文件后上传到数据中心，然后通过消息中间件 ActiveMQ 通知上级医院实时下载文件。为实现这个基本功能，远程传输需要设计与异构系统对接的接口，实现单个或者批量检查文件上传。为了提高文件的传输效率，远程传输模块采用线程池技术，实现文件并发传输。

3、针对医学影像文件调阅困难的问题。远程传输模块遵循 DICOM3.0 标准，利用 dcm4che3 工具包，医学影像文件经过解析后，根据文件的层次结构，分类归档并把相关信息分类输出到配置文件中。

4、针对远程传输模块查询信息不便捷的问题，系统的信息查询模块采用 B/S 架构，融入了 SpringMVC 框架、Spring 框架、Phoenix 框架。用户可以通过浏览器方便快捷地查询传输记录，特殊病例收藏记录以及存储情况统计。

通过完成上述的主要工作，本论文实现了医学影像云存储系统对实现分级诊疗制度，有着积极的推动作用。

关键词：Docker 容器；新型 PaaS 平台；HDFS；HBase；DICOM3.0 标准

Abstract

This paper was designed and achieved medical image system in cloud to solve the problem of medical image files being difficult to be transferred and stored during remote diagnosis, accelerating the implement of the graded diagnosis under the active guidance of national policies. “Difficult Transmission” referring to the large number of medical image files and small files faces the problems of long time of system response and easy loss of file in terms of remote transmission. “Difficult Storage” is because a large number of medical image files are generated every day. Over time, the system will face difficulties in expanding the capacity, damage or even loss of files. The system adopts a "one center, two points" network architecture, which realizes the basic function that the upper-level hospitals download them in real time after the basic hospital upload the inspection files stored on the front-end machine to the data center. The system's data center is a new PaaS platform in cloud based on Docker containers. On these bases, the main work of this thesis are as follows:

1. For the problem of medical image files being difficult to be stored, we utilizes a distributed file system HDFS, distributed database HBase and distributed coordination service Zookeeper to build a high-availability, easy-to-expand, and high-fault-tolerant data storage service. Data storage services use an "active-standby" model to avoid single-point effects. And the data storage service has a certain ability of tolerating some fault through reasonable configuration.

2. For the problem of medical image files being difficult to be transferred, the remote transmission module adopts the C/S architecture and incorporates the Spring framework, the Mybatis framework, and the Phoenix framework. The primary hospital’s files are compressed into ZIP files and are uploaded to the data center, and then the message middleware ActiveMQ is used to notify the superior hospital to download the files in real time. To achieve this basic function, remote transport requires designing a interfaces with heterogeneous systems to upload single or batch file. In order to improve the transmission efficiency of the file, the remote transmission module uses thread pool technology to implement file concurrent transmission.

3. For the problem of tardiness in accessing medical image files, after the medical image file is parsed, according to the hierarchical structure of the file the remote

transmission module which follows the DICOM3.0 standard and uses the dcm4che3 toolkit archives its classification and outputs related information to the configuration file.

4. For the problem that the remote transmission module is not convenient to query information, the system information query module adopts the B/S architecture and integrates the SpringMVC framework, the Spring framework, and the Phoenix framework. Users can query transmission records, collection records of special case and storage statistics quickly and easily through the browser.

Through the completion of the above main work, this paper has realized that the medical image system in cloud has a positive role in promoting the implementation of the graded diagnosis and treatment system.

Keywords:Docker containers, new PaaS platform, HDFS, HBase, DICOM3.0 standard

目录

摘要.....I

Abstract.....II

1 绪论.....7

1.1 课题的背景与意义.....7

1.2 国内外发展现状.....7

1.3 研究内容与目标.....8

1.4 论文的组织结构.....10

1.5 本章小节.....10

2 系统的设计基础.....11

2.1 云计算.....11

2.1.1 Docker 技术.....11

2.1.2 基于 Docker 的新型 PaaS.....12

2.2 DICOM 标准.....12

2.2.1 医学影像文件数据格式.....12

2.2.2 DICOM 文件层次结构.....13

2.3 系统采用的框架.....13

2.3.1 B/S 架构和 C/S 架构的区别.....14

2.3.2 Spring 框架.....15

2.3.3 SpringMVC 框架.....15

2.3.4 Mybatis 框架.....16

2.3.5 Phoenix 框架.....16

2.4 系统采用的中间件.....16

2.4.1 反代理服务器 Nginx.....17

2.4.2 Web 应用服务器 Tomcat.....17

2.4.3 分布式文件存储系统 HDFS.....17

2.4.4 分布式数据库 HBase.....18

2.4.5 分布式协调服务 Zookeeper.....19
2.4.6 消息中间件 ActiveMQ.....20
2.5 本章小节.....20
3 系统的设计与实现.....21
3.1 需求分析.....21
3.2 系统的总体设计.....22
3.3 系统各模块的详细设计.....24
3.3.1 远程传输模块.....24
3.3.2 信息查询模块.....32
3.4 数据库设计.....37
3.5 数据中心的设计与搭建.....39
3.5.1 数据中心的整体设计.....39
3.5.2 数据中心搭建的具体方案.....40
3.6 本章小结.....46
4 系统的关键技术.....47
4.1 医学影像归档及快速调阅方法的改进.....47
4.2 HDFS 集群数据均衡的实现.....48
4.3 数据中心的容灾方案.....49
4.3.1 数据损坏与恢复.....49
4.3.2 数据存储服务的单点失效问题.....49
4.4 远程传输模块的高并发解决方案.....51
4.5 本章小结.....52
5 系统测试.....53
5.1 测试环境.....53
5.2 远程传输模块的测试.....53
5.2.1 功能性测试.....53
5.2.2 性能测试.....55
5.2.3 效果展示.....55
5.3 信息查询模块测试.....57
5.3.1 功能性测试.....57

5.3.2 性能测试.....58
5.3.3 效果图展示.....58
5.4 本章小结.....59
6 总结与展望.....60
6.1 总结.....60
6.2 展望.....61
6.3 本章小结.....62
参考文献.....63
个人简历.....65
致谢.....66

1 绪论

1.1 课题的背景与意义

目前，优质的医疗资源主要集中在各大城市的大型医院，城乡之间配置不均匀，基层医疗机构人才短缺。居住在不发达地区的患者面临看病难看病贵的问题[1]。2018 年 4 月，国务院办公厅发布《国务院办公厅关于促进“互联网+医疗健康”发展的意见》，明确指出大力发展“互联+”医疗服务，并鼓励医疗联合体有效利用互联网技术，开展远程医疗服务，加快落实“基层检查，上级诊断”的分级诊疗格局。开展分级诊疗服务的首要条件是，基层医院患者的检查资料需要与上级医院共享。然而患者做一次放射性检查（如核磁共振检查，CT 扫描等），产生的医学影像文件遵循 DICOM 标准，同时具有文件比较小、数量多的特点，所以造成文件远程传输比较困难。并且，目前国内三甲医院每天都会产生十几 TB 的文件，按此推算，随着分级诊疗快速普及，每天的产生数据量可能会超过这个数据，便带来了医学影像在长期有效地存储管理方面比较困难的问题。

基层医院由于经费紧张，对患者资料的保存状况很糟糕，很容易造成患者资料丢失甚至损坏，这样不利于上级医院的诊断医生获得更全面的资料进行更精准地诊断。为了进一步分级诊疗制度，解决患者看病难、看病贵的难题，本论文需要围绕“传输难，存储难”的问题，设计一款医学影像云存储系统，对加快推进区域医疗、分级诊疗，实现医疗信息共享，有非常重要的意义。

1.2 国内外发展现状

医学影像归档和通信系统（Picture Archiving and Communication System，PACS），主要负责医学影像的传输、存储及管理等。目前，各大医院的 PACS 系统只能在本地医院的局域网中运行，采用集中式的存储方式存储海量的医学影像文件。随着时间的推移，这样的存储方式会造成后期的维护和管理比较困难，不易扩容，甚至出现数据丢失的情况。同时，医院的数据库一般采用传统的关系数据库，如 SQL Server、MySQL 等，虽然使用方便易于维护，但是不适合大量数据进行高并发地读写。国内多家企业把目光转移到云 PACS 系统，以公有云为中心，开展区域医疗服务[2]。目前，国内的云 PACS 系统都把文件上传到公有云的数据中心，从近期效益来看，可以减少系统的开发周期，降低开发成本和运维成本，但是从远期效益来看，根据医学影像诊断中心标准规定，医学影像资料必须在线至少保存三年，近线至少保存十年，这样势必会带来患者信息资料泄露的危险，同时拥有公有云的企业能否一直运营下去，这是各大医院担心的问题。并且

云 PACS 系统的文件传输都是以单个医学影像文件为单位上传到数据中心，然而在带宽不理想的情况下，系统的响应时间是不可忽略的因素，文件过多，响应时间过长，不仅降低了文件传输效率，而且大大增加了文件丢失的概率。

云计算（Cloud Computing）的概念是 20 世纪 60 年代由美国斯坦福大学教授 John McCarthy 首次提出来的。2006 年 8 月，谷歌将云计算技术第一次引入人们视野。经过十年的发展，据统计,2016 年在美国，运用云计算的企业达到 54%，而使用了云计算服务的新成立的互联网公司已经达到 90%以上。根据中国互联网络信息中心的数据统计,截至 2016 年 12 月,被调研的企业中，采用云计算的企业仅仅 21.4%[3]。由此可见，中国云计算的发展水平和欧美国家还有一定的差距。

2008 年，Amazon 公司发布了 Google 发布了 SimpleDB，EC2 等服务，同时谷歌公司也发布了 Google App Engine。这代表公有 PaaS 时代的到来。但是，私有 PaaS 的发展还不成熟，直到 Docker 容器技术的快速发展，私有 PaaS 迎来了曙光。根据 Linux 基金会的调查，在云计算开源项目方面，在受欢迎程度方面，Docker 仅次于 OpenStack。谷歌、Microsoft、Amazon 等公司的数据中心也支持 Docker 技术[5]。以 Docker 为基础的新型 PaaS 不仅解决了应用对传统 PaaS 平台的依赖性，而且可以提高应用的可移植性，实现应用的快速迭代[4]。

面对海量的数据存储，国内越来越多的企业开展了对 Hadoop 的研究和应用。中国电信开发了基于 Hadoop 的天翼云，淘宝利用 Hadoop 平台处理和存储电子商务交易的相关数据。此外，还有百度云，亚马逊的 AWS[5]。

为加快推进分级诊疗制度，各大医院需要一个以 Docker 容器为基础的，可以提供海量数据存储服务的新型私有 PaaS 平台。本课题的研究对加快推进分级诊疗制度的落实，有非常积极的作用。

1.3 研究内容与目标

本课题围绕医学影像文件“传输难、存储难”的问题，设计一款医学影像云存储系统。系统采用“一个中心，两个点”的网络架构，即一个提供海量的数据存储服务，Web 服务，JMS 服务数据中心，两个前置机分别部署在基层医院和上级医院，用来存储近期需要调阅的医学影像文件，和与本地 PACS 系统进行数据交互。遵循 DICOM 3.0 标准（Digital Imaging and Communications in Medicine，医学数字成像和通信），医学影像云存储系统可以把单个或批量检查文件从基层医院实时传输到上级医院，并且根据医学影像文件的层次结构进行解析然后分类归档以及相关信息持久化。随后，医生可以用其他医学影像阅读软件高速精准地调阅每一个文件。与此同时，医生可以根据权限方便快捷地查询传输记录、存储信息，收藏特殊病例并能导出所有的收藏记录。

为实现这个研究目标，系统需要满足以下几个方面：

（1）可以和其他系统进行对接。首先需要设计一个可以和远程诊断系统对接的接口，完成批量检查文件上传的功能，解决思路：解析远程诊断系统传递的参数，获取上传列表，然后根据默认的文件路径规则，找到文件，逐一上传。如果出现文件找不到的情况，用户可以手动选择文件目录，然后上传。其次需要设计一个可以和医学影像高速传输系统对接的接口，可以把基层医院的放射医疗设备当天产生的医学影像文件及时上传到数据中心，解决思路是：周期性地查询本地 PACS 系统的数据库，获取文件上传的列表，然后把文件逐一上传到数据中心。

（2）远程传输方面。首先，针对浏览器处理大文件的能力比较弱的问题，系统的远程传输模块采用 C/S 架构。其次，由于医学影像文件数量多，文件小，大批量医学影像文件远程传输时，常常出现文件丢失的现象。针对降低文件丢失概率问题，系统的解决思路：检查文件经过遍历，筛选出所有的医学影像文件，然后把这些文件压缩为一个比较大的 ZIP 文件，这样可以大幅度降低系统开销的同时，也能降低数据丢失的几率。针对大量的检查文件需要不定时地传输到数据中心，同时也要保证文件的传输效率的问题，系统的解决思路：基层医院把医疗检查设备产生的所有医学影像文件上传到数据中心时，系统采用定时线程池技术，周期性地查询本地数据库获取文件上传列表，然后用把文件并发上传到数据中心。系统远程传输模块的下载端采用定长线程池技术，实时并发下载文件。最后，针对随着时间推移，远期文件堆积在前置机上，造成存储资源的浪费的问题，解决思路：系统采用一个定时周期任务，自动清除超过期限的文件。

（3）医学影像文件的解析与归档。在成千上百的检查文件中，高速精准调阅每一个医学影像文件是比较困难的，系统的解决思路：系统根据医学影像文件的层次结构，利用 dcm4che3 工具包，解析每一个医学影像文件，然后分类归档，并把相关信息持久化。

（4）信息查询方面。系统采用 B/S 架构开发信息查询模块，医生可以在浏览器上随时随地查询相关信息。并且针对每一条传输记录进行查询详情、信息补充、检查文件下载、病例收藏、历次检查信息等功能。医生可以对收藏的特殊病例进行多种操作并可以导出所有的收藏记录。系统管理员能随时查看数据中心的状态信息，如果遇到数据分布不均衡，可以远程控制数据中心进行数据均衡。

（5）数据中心的搭建。数据中心需要满足以下需求：1、为了解决 Web 服务由于访问量猛增或服务器宕机，造成服务崩溃的问题，系统采用反代理服务器 Nginx，根据权重轮询的策略，进行负载均衡。2、为了降低数据中心的维护成本，提高系统的可移植性以及可以快速迭代的问题，数据中心需要采用以 Docker 为基础的新型 PaaS 平台。为了长期存储海量的医学影像文件，数据中心采用 Hadoop

组件（分布式文件系统 HDFS、分布式数据库 HBase、分布式协调服务 Zookeeper）搭建数据存储服务，通过合理地配置，满足高可用、高容错、易扩容的要求。

1.4 论文的组织结构

本论文以医疗影像云存储系统的设计与实现为主线，然后对课题的各个方面逐一进行的阐述

首先介绍了本论文的研究背景与意义，以及国内外现状，以及研究的主要内容。然后论文介绍了医学影像云存储系统的设计基础，接着阐述了系统的需求分析和系统的总体框架，并详细介绍了系统各模块的详细设计。随后针对系统的几个关键技术进行了重点阐述，最后对系统的进行了功能测试和性能测试，总结系统的不足之处，并做出展望。本论文各章节内容如下：

第一章绪论，介绍了本论文在分级诊疗的背景下的研究背景和意义，然后介绍了国内外发展现状，以及本论文研究的主要内容，最后对本论文的组织结构进行说明。

第二章系统的设计基础，简单介绍了云计算的概念，然后介绍了数据中心的相关基础，然后介绍了系统涉及的 DICOM3.0 标准的相关知识，最后系统开发用到的开发框架，以及中间件。

第三章医学影像云存储系统的总体设计，本章先进行了需求分析，然后介绍了总体架构，然后对各个模块的详细设计进行描述。

第四章系统的关键技术，本章主要介绍了系统实现的过程中比较重要的几个关键技术。

第五章系统测试，本章先介绍了系统的测试环境，然后对系统的各个模块分别进行了功能测试和性能测试

第六章总结与展望，本章先总结了一下全文，然后指出系统的不足，最后对系统的发展进行了展望。

1.5 本章小节

本章主要介绍了课题的研究背景和意义，然后分析了国内外的研究现状，然后介绍了论文的研究内容与目标，最后介绍了论文的组织结构。

2 系统的设计基础

本章主要介绍医学影像云存储的相关内容，包括系统采用的框架，医学影像解析需要遵循的 DICOM3.0 标准，以及数据中心的相关基础和采用的中间件。

2.1 云计算

云计算是一种，允许用户以通过网络访问一个可动态配置的共享计算资源池的模式，用户可以用最小的管理代价就可快速发布各种应用服务。云计算具有五大基本特征，分别是服务可计量、资源池化、弹性伸缩快捷、按需的自助服务以及网络接入广泛[6]。云计算有四种部署模式，分别为：公有云、行业云、私有云和混合云。如果按服务模式划分，云计算可以分为：

（1）基础设施即服务（Infrastructure as a Service, IaaS)，数据中心可以为用户提供付费的基础设施，部署用户自己的应用。用户不需要耗费很大的经济成本和人力成本去搭建基础设施平台。

（2）平台即服务（Platform as a Service，PaaS)，用户可以借用平台提供完备的生产环境（比如硬件设施、数据库软件，操作系统等），直接部署自己的应用，这样可以节省用户开发周期与人力成本。

（3）软件即服务（Software as a Service , SaaS)，供应商为用户提供相应的软件，用户可以通过网络操作软件。系统的维护与升级，全部交给供应商。

2.1.1 Docker 技术

Docker 是 dotCloud 公司于 2013 年开发的开源容器项目。Docker 采用虚拟化技术，使在其中运行的代码如同在虚拟机里。Docker 的设计思想是“一次封装，到处运行”[7]。Docker 通过对应用组件的封装（Packaging)、分发（Distribution)、部署（Deployment）、运行（Runtime）生命周期进行管理。应用组件可以是一个 Web 应用，一个开发环境或者是一个操作系统或者集群。Docker 的最大优势是，快速分发和部署。如果遇到服务器拆迁，开发人员需要耗费大量的精力进行重新部署和调试。而如果开发人员采用 Docker 容器技术，直接在新的服务器上启用需要的容器[8]。

在开发和运维的过程中，Docker 有几个比较突出的优势：1、交付和部署的周期短。开发人员使用 Docker 的镜像创建一个理想的开发环境，然后测试和运维人员可以快速搭建起完全一样的开发环境。这样，可以节省了大量的应用开发周期。2、资源利用率高。Docoker 是操作系统级别的虚拟化技术，不需要任何虚拟化管理工具的支持。同时，Docker 需要的资源很低。3、迁移和扩展便捷。Docker

容器可以在任何平台上运行，用户可以很容易地进行应用迁移。4、更新管理更便捷。如果使用 Dockerfile,开发人员可以只修改配置文件，就可以完成繁杂的更新工作。

Docker 的三大核心技术是：镜像（Image)、容器（Container）、仓库（Repositoty)。Docker 镜像是创建容器的基础。一个镜像可以容纳一个操作系统，在其操作系统上可以安装一个应用软件，比如 Redis,Tomcat 等。Docker 容器是通过 Docker 镜像创建的应用运行的实例。Docker 仓库集中存放着 Docker 镜像文件。Docker 仓库可以分为公开仓库和私有仓库。任何用户都可以从公开仓库下载需要的镜像，如 Docker Hub，国内腾讯云和阿里云提供的仓库。如果用户不想分享自己镜像文件，可以在本地网络内创建一个私有仓库。

2.1.2 基于 Docker 的新型 PaaS

新型 PaaS 在 Docker 容器的基础上搭建的一站式的应用开发运行平台，即 Docker 容器把应用和依赖的中间件等运行环境打包为一个镜像，应用的进程在 Docker 容器中运行，而不依赖宿主机提供的运行环境[9]。用户以应用打包的 Docker 镜像为单位提交到 PaaS 平台，实现了应用和 PaaS 平台的解耦[10]。这样的设计，不仅为用户开发应用带来了便捷，并且 PaaS 不需要提供各种各样的运行环境和应用的打包过程。

2.2 DICOM 标准

DICOM（Digital Imaging and Communication in Medicine，即医学数字成像与通信），是医学信息领域中医学影像和相关信息的国际标准。DICOM 标准中包含了医学数字影像从采集、通信到归档、显示等所有信息交互的协议。

2.2.1 医学影像文件数据格式

DICOM 标准定义了用于临床数据交互的文件格式。医学影像文件不仅包含了影像的像素信息，还有病人的基本信息和检查信息。所以，系统解析医学影像文件的功能必须遵循 DICOM3.0 标准。

医学影像文件的数据结构如图 2.1 所示，医学影像文件的数据由文件头和数据集合组成。每个医学影像文件都必须包含一个文件头，因为标识数据集合的相关信息在文件头中[11]。文件头由 128 个字节组成的，包含导言、前缀和文件信息元素组成。前缀的内容，用来识别医学影像文件，都为“DICM”。数据集合由按规则排列的数据元素组成，是医学影像文件重要的一部分。数据元素包含 4 个部分，标签（Tag)、数据描述（Value Representation，VR）、数据长度（Value Length，

VL）和数据域（Value Field，VF)。标签是一个 4 字节的无符号数整数，前 2 个字节表示组号，后 2 个字节表示元素号。每个数据元素都可以用标签唯一表示。数据描述表示该数据元数的数据的数据类型。数据长度指出还数据元素数据域的字节数。数据域指这个数据元素的值。

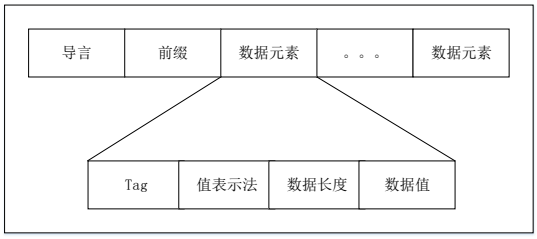

图 2.1　医学影像文件的数据结构

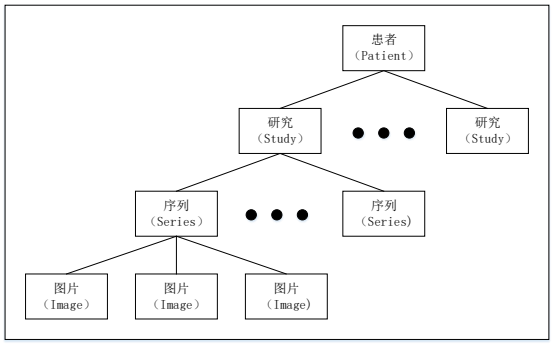

图 2.2　医学影像文件的层次结构

2.2.2　DICOM 文件层次结构

目前，“软读片”在各大医院慢慢普及推广。“软读片”，就是医生可以直接在 PC 机上阅读医学影像文件，而不借用胶片媒介。所以，在临床诊断过程中，患者做一次放射检查，产生的 DICOM 文件越来越多，少则几十幅，多则几万幅。所以，医学影像文件将影像信息按层级划分为 4 个级，分别为：患者（Patient)、研究（Study)、序列（Series)、图像（Image)[12]。如图 2.2 所示，一个患者做检查前会生成一个检查号，检查号作为该患者（Patient)的唯一标识 UID。一个患者可以有若干个研究（Study)，一个 study 可以有若干个序列（Series),一个 series 可以有若干个图像。不同的层次都对应一个唯一标识 UID。虽然，患者做一次检查的影像文件成千上万，但通过这 4 个层级的 UID,可以快速精确地找到想要的医学影像文件。

2.3　系统采用的框架

应用软件如果采用 C/S 架构，就会具有比较强的业务逻辑处理能力，响应速度块，但是不易升级和维护，用户只能在安装软件的电脑上操作；采用 B/S 开发

的 web 服务，虽然使用方便快捷，有利于系统的维护和升级，但是无法处理数量多或者比较大的文件。因此，为了提高用户体验，更高效地传输文件，更方便地查询信息，医学影像云存储系统采用 C/S 和 B/S 结合的混合架构。

医学影像云存储系统的远程传输模块采用 C/S 架构，因为患者检查文件包含成百上千的医学影像文件,甚至有的达到几万张，检查文件的大小从几十兆到几个 GB。如果用户通过浏览器传输文件会出现响应延迟比较长，甚至造成浏览器崩溃的情况，严重影响用户的体验。同时，为了实现软件开发“高内聚，低耦合”的目标，远程传输模块融入了 Spring 框架、Mybatis 框架和 Phoenix 框架。Spring 可以负责创建所有的实例，维护实例之间的依赖关系，这样就可以降低所有实例之间的耦合性。而 Mybatis 实现 SQL 操作与逻辑代码的分离，访问数据库获取的结果集可以自动与实例进行映射，并且在维护上简单方便。同时，Spring 对 Mybatis 提供了很好的支持作用。Phoenix 框架可以实现通过 JDBC 访问 HBase 数据库。

医学影像云存储系统的信息查询模块采用了 B/S 架构，融合了 SpringMVC 框架、Spring 框架和 Phoenix 框架。SpringMVC 框架可以与 Spring 框架进行完美对接，实现 Web 层的解耦。这样的设计，不仅实现了“高内聚，低耦合”的原则，而且用户可以在任何电脑上通过浏览器查询系统的相关信息。

2.3.1 B/S 架构和 C/S 架构的区别

C/S 架构，即 Client/Server（客户端/服务器端）架构，把需要执行的任务有效地分配到客户端和服务器端，减少了系统的通信开销，同时可以充分有效地利用客户端和服务器端的硬件环境的资源。C/S 架构把系统分为客户端和服务器端两层：客户端负责界面显示和业务逻辑 ，服务器层是通过网络访问数据库服务器。也就是说，C/S 架构的系统分为用户表示层和数据库层。由于客户端与服务器端之间直接相连，所以，它的响应速度非常快。

B/S 架构，即 Browser/Server（浏览器端/服务器端）架构，互联网技术中新兴的网络结构模式。这种模式，客户端采用 Web 浏览器，如谷歌、猎豹、IE 等浏览器，系统的业务逻辑与数据库交互等核心部分集中在服务器上。服务器上可以安装数据库，如传统关系型数据库（SQL Server,MySQL，Orancle 等），和非关系型数据库（HBase,Redis,MongoDB 等）。用户可以通过浏览器，访问远程服务器，与数据库进行数据交互。用户可以用任何浏览器，通过网络访问服务器。B/S 架构便于系统升级，维护，只需在服务器端进行升级和维护，所有用户的客户机不用任何操作。但是，它的响应速度不及 C/S 模式。同时，由于业务逻辑集中在服务器，客户端是 Web 浏览器，所以 B/S 模式处理本地文件的能力较弱。

这两种架构的区别是，B/C 架构是 C/S 架构改进后的网络结构模式，B/S 架构属于 C/S 架构，因为 B/S 架构的客户端是 Web 浏览器[13]。此外，C/S 可以融入任何通信协议，而 B/S 架构规定只支持 HTTP 协议。

2.3.2 Spring 框架

Spring 框架是一个轻量级的 Java 开发框架，它解决了企业应用开发中的复杂性问题。Spring 框架的主要优点是分层架构，允许开发者可以自由选择开发组件。Spring 可以利用 JavaBean 来代替 EJB 实现相应功能[13]。Spring 有两个核心，控制反转（IOC）和面向切面（AOP）。Spring 框架主要由 6 大模块集合组成，如 2. 3 所示，分别是核心容器（Core Container）、AOP（Aspect Oriented Programming）和设备支持（Instrmentation）、数据访问与集成（Data Access/Integeration）、Web、报文发送（Messaging）、测试[14]。这些模块集合或者模块可以单独使用, 也可以多个模块联合使用。

Spring 框架的优点是：1、方便解耦，简化开发。开发人员可以把对象之间的依赖关系由 Spring 的核心 IOC 负责。这样开发人员不再为生成实例，配置文件的解析而编写代码[15]。2、支持 AOP 编程。用户可以把业务逻辑中的通用功能分离出来，单独编写代码。在执行业务逻辑时，会根据业务逻辑的要求，自动把该代码切入到合适的位置。3、支持声明事务。开发人员可以通过声明的方式管理事务，提高开发的质量和效率。4、程序测试便捷。Spring 提供了对 Junit4 的支持，测试程序非常便捷。5、支持集成各种优秀框架。Spring 支持其他优秀的框架（如 Mybatis、Hibernate、 Quartz 等），并且可以减少框架的使用难度[16]。

图 2.3 Spring 框架的组成

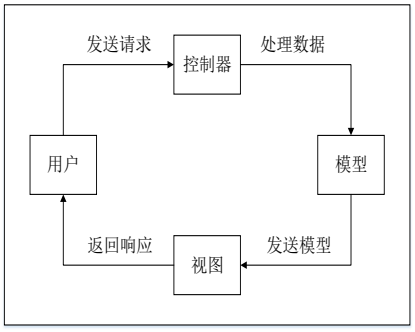

图 2.4 MVC 架构模式的工作原理

2.3.3 SpringMVC 框架

MVC(Model View Controller ，模型-视图-控制器)架构模式，把系统分为 3 个部分，即模型、视图和控制器。模型（Model）负责系统的数据持久化，视图（View）负责系统与用户的交互，控制器（Controller）可以负责请求的分发和处

理，并选择相应的视图显示模型的数据。MVC 模式的工作原理如图 2.4 所示，用户向控制器发送请求，控制器接受用户请求并委托模型进行数据处理，模型把接受到的数据进行业务逻辑处理后数据持久化，然后推送模型至视图，视图显示模型的数据，并响应用户的请求[17]。

SpringMVC 是一种基于 Java 的，使用请求-响应模型模型的轻量级框架，实现了 MVC 架构模式的设计思想。SpringMVC 是简化 Web 服务开发的。

2.3.4 Mybatis 框架

我们在 JavaEE 项目开发中，需要使用 Java 中的 JDBC(Java Data Base Connectivity)实现与数据库的数据交互以及数据持久化。MyBatis 通过 XML 进行配置和原始映射，把数据访问接口和 POJO（Plain Old Java Objects，普通的 Java 对象）通过映射文件与数据库进行交互，省略了通过 JDBC 进行数据交互的代码逻辑[18]。

Mybatis 框架可以分三层：1、Java API 接口层，开发人员可以通过调用接口直接对数据库进行访问。2、数据处理层，负责具体的 SQL 操作和结果的映射等，完成与数据库进行数据交互的任务。3、基础支撑层，为数据处理层提供连接管理、配置加载、缓存处理和事务管理[19]。

2.3.5 Phoenix 框架

分布式数据库 HBase 不是关系型数据库，所以它不支持 SQL 查询。开发人员可以通过 HBase 提供的 Java API 进行数据的读写。所以，开发人员需要开发一个 HBase 的客户端。但是，大多数的开发人员习惯利用 Java 提供的 JDBC（Java DataBase Connectivity,java 数据库连接）连接数据库。为实现这个目的，开发人员可以利用 Apache Phoenix，实现通过 JDBC 方式与 HBase 进行数据交互[20]。Phoenix 是基于 Java 语言，并作为 HBase 的 JDBC 驱动。Phoenix 支持 ACID 事务功能，支持标准的 SQL 和 JDBC API。它还可以通过 SQL 查询来完成一个或者多个 HBase 扫描，并编排执行以生成 JDBC 的结果集。

2.4 系统采用的中间件

医学影像云存储系统的数据中心需要提供 Web 服务、海量的数据存储服务、JMS（Java Message Service，Java 消息服务）服务。数据中心都是以 Docker 容器为基础的新型 PaaS 平台，Web 服务需要负责负载均衡的反代理服务 Nginx，Web 应用服务器 Tomcat；海量的数据存储服务需要分布式文件系统 HDFS，分布式数

据库 HBase 以及分布式协调服务 Zookeeper；JMS 服务采用消息中间件 ActiveMQ。本小节将逐一阐述采用这些中间件的原因。

2.4.1 反代理服务器 Nginx

Nginx 是俄国开发的一款高性能轻量级的反向代理服务器和 web 服务器。由于运行的 Nginx 占用内存比较小、并发能力比较强，中国很多网站使用 Nginx，比如百度、网易、京东等[21]。Nginx 可以运行在 Windows、Linux、Mac Os 等操作系统上，同时可以支持百万级别的 TCP 连接。Nginx 可以作为反向代理实现负载均衡，它有四种负载均衡策略：1、权重轮询，即把接收的请求根据权重按顺序逐个分配到不同的服务器上，如果某个服务器宕机，Nginx 会自动把该服务器剔除队列，请求的处理情况不受影像。2、ip_hash 策略，即每个请求按照 IP 的 hash 结果分配。3、fair 策略。根据服务器处理请求的响应时间进行均衡分配，响应时间短请求处理效率高的服务器请求分配的概率比较高。4、url_hash 策略，即根据访问的 URL 的 hash 结果进行分配请求，每一个 URL 请求指向特定的服务器[22]。

2.4.2 Web 应用服务器 Tomcat

Tomcat 是 Apache、SUN 和其他公司基于 Java 共同开发的一款轻量级开源的 Web 服务器。同时，它也是一款比较优秀的 Servlet 容器[23]。在中小型系统和并发访问的用户不多的情况下，Tomcat 是开发和调试 JSP 的首选。Tomcat 凭借先进的技术、稳定的性能，是目前比较流行的 Web 应用服务器。本系统的 Web 服务是通过 Tomcat 实现的。

2.4.3 分布式文件存储系统 HDFS

HDFS（Hadoop Distributed File System，Hadoop 分布式文件系统）具有高度容错性，所以适合部署在廉价的服务器上。同时,它能提供高吞吐量的数据访问，非常适合拥有数据集比较大的应用程序。

磁盘进行读/写数据时的最小单位是该磁盘默认的数据块大小[24]。HDFS 集群存储的文件也被划分多个数据块。HDFS 集群默认的数据块为 128M,开发人员可以根据需要自己设定。如果一个文件小于文件系统默认的数据块大小，存放到文件系统不会占用整个数据块的存储空间。一般地，普通磁盘默认的数据块大小为 512 字节，而 HDFS 文件系统设定的数据块比前者大的多。后者的目的是为了最小化寻址开销。由于数据块很大，文件的每一个数据块从本地磁盘传输的时间会远远多于定位数据块开始位置的时间。因此，磁盘传输速度决定了一个拥有诸多数据块的文件传输时间。这样设计的好处是：1、需要存储的文件的大小可以

大于文件系统中任意一个磁盘的大小。因为文件的数据块不需要存放在同一磁盘上。2、存储单元是抽象的数据块，数据块是文件系统的存储对象，这样的设计简化了文件的存储管理。3、数据块有利于进行数据备份，这样可以进一步提高数据的容错能力和可用性。每个数据块复制到几个物理上相互独立的服务器上，默认为 3 个，这样就确保在数据块、磁盘或服务器发生故障后不会丢失数据。如果发现数据块出现损坏，文件系统会读取另一个副本从其他磁盘[25]。

HDFS 的设计具有以下几个特点：

1、超大文件。文件的大小有几十 MB、几十 GB、甚至几十 TB。目前，存储 PB 级数据的 HDFS 集群已出现了。

2、流式数据访问。最高效的访问模式是“一次写入、多次读取”，这也是 HDFS 的构建特点。

3、普通的商用硬件。Hadoop 适合运行在普通商用硬件的集群上。当集群的规模很庞大时，节点有很高的故障，所以集群需要设计成高可用的。

4、数据访问的时间延迟。HDFS 适用于高数据吞吐量的应用，不支持低时间延迟数据访问的应用。

5、不适用海量小文件存储。由于文件系统的元数据存放在 Namenode 的内存中，所以 Namenode 的内存容量决定文件系统存放的文件总数。一般地，一个文件的相关存储信息大约为 150 字节。所以，文件系统的文件总数达到几百万个是可行的，但是超过几十亿个，就超出目前硬件的能力。

6、不支持任意修改文件。HDFS 只支持单个用户写入文件，并且只支持在文件末尾添加数据。

2.4.4 分布式数据库 HBase

Apache HBase 是一个满足实时地访问规模比较大的数据面向列的分布式数据库，可以满足用户实时访问海量的数据。同时，HBase 是可以运行在 HDFS 上的 NoSQL 数据库。HBase 有较强的伸缩性，它可以通过增加节点的方法来实现线性拓展。目前，很多互联网公司正在使用 Hbase,如 Facebook、Twitter、Yahoo 等[26]。HBase 把数据存储在带时间戳的表中。表中的每个单元格由行和列交织的构成，HBase 把相应的数据存放到对应的单元格中，会默认加入时间戳。用户通过表的主键进行访问，同时，表根据主键的键值的字节序进行排序。HBase 中的表由很多行构成，表中的行由多个列族构成，每个列族由多个列构成。所有的成员都存放在文件系统上[27]。

HBase 表在初始创建时，只有一个区域（region)，随着存储的数据的增大，达到设定的阈值时（默认为 10GB）,表会自动分裂成两个区域（region）。区域

（region)是 HBase 集群数据分布的最小单位。当一个表存放的数据量超过宿主机的容量，表可以分成若干个区域分别存放在不同的服务器上。

HBase 集群的构成与前文提到的 HDFS 集群相似，即一个 Master 节点协调管理一个或者多个 Regionserver 从属机，如图 2.5 所示。Master 的主要工作有三个，分别是：1、负责启动安装；2、合理地把区域分配给 Regionserver；3、恢复出现故障的 Regionserver。Regionserver 的主要工作有两个，分别是：1、管理零个或者多个区域；2、相应客户端的读/写请求。

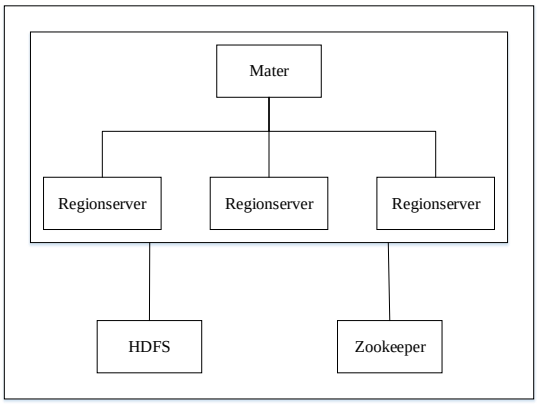

图 2.5　HBase 集群的成员

HBase 与传统的关系型数据库管理系统(Relational Database Management System，RDBMS)的区别从几个方面逐一分析。在硬件设施方面，HBase 集群可以运行普通的服务器上，而 RDBMS 需要配置比较高的服务器上。在容错机制上，HBase 集群同时运行多个节点，个别节点宕机不影响用户正常访问；RDBMS 需要一个有效的高可用(HA)机制进行维护。在数据容量上面，HBase 可以存储 PB 级的数据，而 RDBMS 只能存放 TB 级数量。在支持的数据类型方面，HBase 只支持字节（Bytes）类型，而 RDBMS 支持的数据类型很丰富，比如字符（Char)类型、Interger(整数类型)、日期（Date）类型等。在吞吐量方面，HBase 的吞吐量达到每秒几百万次查询，RDBMS 的吞吐量是每秒数千次查询。

2.4.5　分布式协调服务 Zookeeper

Zookeeper 是 Hadoop 家族的分布式协调服务。它具有六大特性分别是：1、轻量级。Zookeeper 提供一些比较简单的操作，它的核心是一个比较精简的文件系统。2、具有高可用性。应用程序可以依赖 Zookeeper 的高可用性，Zookeeper 可以运行一组节点，辅助系统避免出现单点故障的问题。3、高性能。针对写操作，Zookeeper 的吞吐量已经超过每秒 1000 次操作，而对于读操作，吞吐量超出了高几倍。4、拥有一个资源库。它提供了一个通用协调模式实现方法的开源共享库，开发人员可以直接对资源库进行添加和改进。5、松耦合交互方式。进程

之间需要通信时，可以在不同时存在的情况下，通过 Zookeeper,进行信息交互。6、富有表现力。Zookeeper 可用于进行多种协调数据结构和协议的实现，比如分布式队列和分布式锁[28]。

2.4.6 消息中间件 ActiveMQ

Apache ActiveMQ 是 Apache 基于 Java 语言开发的开源消息中间件。JMS 定义了有关消息传递的标准和规范，是一组实现消息传递的接口。而 ActiveMQ 是 JMS 的具体实现。产生消息的一方叫生产者，接收消息的一方叫消费者。消息传递有两种方式，一种是点对点通信。生产者把产生的消息放入专门的队列 Queue,然后只有一个消费者接收消息。另一种方式是发布/订阅模式，生产者把消息放入主题 Topic,然后订阅了这个主题 Topic 的消费者都可以接收到这个消息。本系统的 JMS 服务采用点对点通信模式，实现文件实时传输的功能[29]。

2.5 本章小节

本章主要介绍了医学影像文件云存储平台搭建涉及的相关知识。本章简单介绍了数据中心采用以 Docker 容器为基础的新型 PaaS，以及数据中心实现各种服务采用的中间件。解析医学影像文件需要了解的数据集的数据元素构成，快速调阅单幅医学影像文件需要了解的医学影像文件层次结构。本章也说明了采用 C/S 和 B/S 架构的原因，介绍了系统采用的多个框架。

3 系统的设计与实现

本章首先进行需求分析，然后介绍了系统的总体设计，之后详细阐述了基于C/S 架构的远程传输模块的详细设计，基于 B/S 架构的信息查询模块的详细设计，数据库表设计，以及数据中心的搭建方案。

3.1 需求分析

在云计算技术、移动互联网技术的高速发展背景下，随着医学数字成像技术的发展与快速推广，每天都会产生的医学影像文件也随着爆炸式增长。例如，患者做一次 CT 检查，一般会产生几百个医学影像文件，大概有几十兆，如果患者需要做一次 CT 增强，文件数量会达到几千个，文件大小大约几个 GB。

为满足分级诊疗服务需求，医学影像云存储系统需要把医学影像文件高效地从基层医院传输到上级医院。同时，基层医院的医生可以很方便快捷地查询本医院的传输记录，系统管理员可以在线查询数据中心的节点运行情况及存储情况。综上所述，医学影像云存储系统需要满足以下需求：

（1）系统的远程传输模块可以与其他异构系统进行数据交互。比如与远程诊断系统进行对接，可以实现单个或批量检查文件同时上传；与医学影像高速传输系统对接，实现把基层医院放射科的医疗检查设备产生的文件全部上传。

（2）系统可以从基层医院的本地数据库读取需要远程诊断的患者信息，然后存储到数据中心。

（3）系统可以与远程诊断系统的数据库进行数据交互。系统把文件传输完毕，可以修改远程诊断系统数据库的该患者检查文件的状态信息，以便上级医院可以查询记录及时进行诊断。

（4）提高文件传输效率，降低数据丢失的概率。系统需要把待上传的检查文件，遍历筛选出标准的医学影像文件，然后压缩为 ZIP 文件。

（5）上级医院可以实时下载检查文件。系统采用 ActiveMQ 消息中间件实现 JMS 服务，基层医院文件上传完毕后，上级医院可以实时下载文件。

（6）医学影像文件的解析与归档。系统需要对每一个医学影像文件进行解析，根据医学影像文件的层次结构进行分类归档，然会把这些文件的层次信息分类记录到配置文件中，供医学影像阅读软件高速精准地调阅每一个文件。

（7）数据中心支持海量的数据存储服务。数据中心的数据库采用分布式数据库 HBase、采用分布式文件系统 HDFS 和分布式协调服务 Zookeeper,三个组件搭建一个支持满足高可用、高容错、易扩容的文件系统。

（8）用户查询相关信息便捷。系统的信息查询模块是采用 B/S 架构的 Web 服务，用户可以在任何电脑上通过浏览器查询相关信息。

（9）数据中心支持带有负载均衡的 Web 服务。系统采用反代理服务 Nginx，根据权重轮询原则，进行负载均衡, 避免访问请求过多的时候，Web 服务过度忙碌，访问请求无法及时处理。

（10）医生可以根据多重条件查询传输记录，并针对每条记录可以进行查看信息详情、信息补充、文件下载、病例收藏、查看历次检查记录、查看文件的调阅记录等操作。

（11）医生可以根据权限查看医院的存储情况，基层医院的医生可以查看本医院的存储情况，上级医院的医生和系统管理员可以查看所有医院的存储情况。

（12）系统管理员可以查看数据中心几个关键的节点运行情况。如果遇到单点失效的情况，可以及时处理。

（13）远程控制数据中心进行数据均衡。当系统管理员查看各节点运行情况，发现文件系统存储数据的节点 Datanode，数据分布不均匀时，可以远程启动 HDFS 的均衡器，进行数据均衡。

（14）设计一个合理、高效的数据库表。系统需要一个设计合理、高效的数据库表，进行数据存储。

3.2 系统的总体设计

医院的 PACS 系统都是在本地局域网的环境下运行，为了保证医院的信息安全，基层医院和上级医院都需要设置一台前置机，既可以与本地 PACS 系统进行数据交互，又可以通过广域网与数据中心连接。因此，本系统采用“一个中心，两个点”的网络架构，如图 3.1 所示，“一个中心”指一个医学影像云存储系统的数据中心，“两个点”指基层医院和上级医院分别设置的前置机，前置机上存放近期需要调阅的医学影像文件，避免医学影像文件被无关人员获取，并且通过前置机，系统可以与本地医院 PACS 系统进行数据交互。

综上所述，医学影像云存储系统按功能分为两个模块如图 3.2 所示，分别是远程传输模块和信息查询模块。远程传输模块主要负责单个或者批量检查文件上传、文件实时下载、医学影像文件的解析与归档，并且与异构系统的数据库进行数据交互。信息查询模块主要功能是查询文件传输记录并进行多种操作，特殊病例收藏及导出收藏记录，还有数据中心的存储情况统计、各节点的运行情况、以及出现存储数据分布不均衡时，可以远程控制文件系统进行数据均衡。

因为患者检查文件的文件数量比较多，单个文件比较小,总占用空间比较大。如果用户通过浏览器传输文件会出现响应延迟比较严重的情况，为了提高文件的传输效率，远程传输模块采用 C/S 架构。同时，为了实现“高内聚，低耦合”的开发原则，远程传输模块融入了 Spring 框架、Mybatis 框架和 Phoenix 框架，如图 3.3 所示。View 层用 JavaFX 开发，负责与用户信息交互。JavaFX 是 Java 的图形用户界面工具包，可以用来创建和部署富客户端应用程序。Controller 层负责业务逻辑处理，融入了 Spring 框架。Model 层负责系统数据的持久化，加入 Mybatis 框架和 Phoenix 框架。该模块利用 Mybatis 访问异构系统的数据库，利用 Phoenix 访问数据中心的数据库。

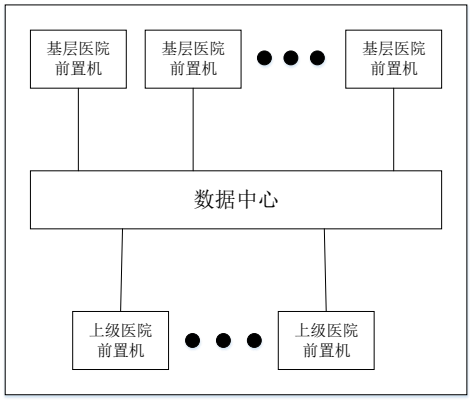

图 3.1 医学影像云存储系统的网络架构

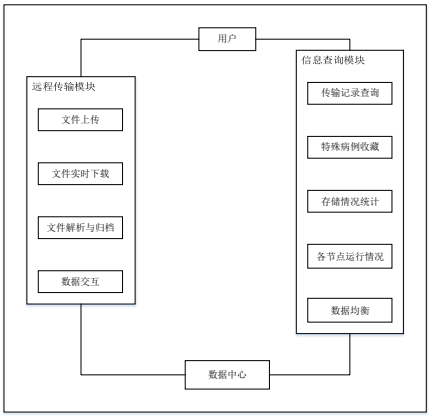

图 3.2 医学影像云存储系统的主要功能架构

为了让用户可以在任何电脑上查询相关的信息，信息查询模块是基于 B/S 架构，融合了 SpringMVC 框架、Spring 框架和 Phoenix 框架，如图 3.4 所示。视图层负责与用户进行信息交互，由 JSP/HTML 组成。表现层通过调用业务层负责处理具体的业务模块流程的控制，加入 SpringMVC 框架。业务层负责业务模块的逻辑应用设计，加入 Spring 框架。持久层负责系统数据的持久化，加入 Phoenix 框架。

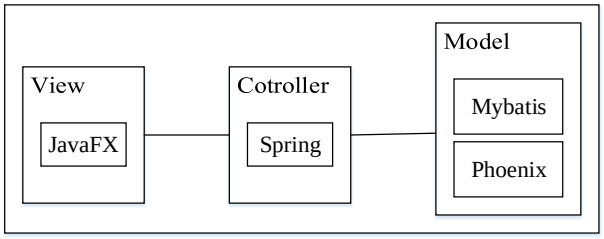

图 3.3　远程传输模块框架结构

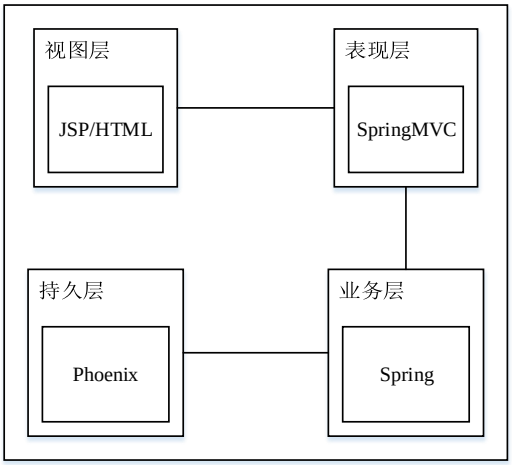

图 3.4　信息查询模块框架结构

3.3　系统各模块的详细设计

3.3.1　远程传输模块

在基层医院，远程传输模块需要和远程诊断系统对接，可以实现批量检查文件同时上传，也可以自定义选择上传文件的目录，同时也需要与医学影像高速传输系统对接，把基层医院放射科的医疗检查设备当天产生的文件及时传输到数据中心。本人曾在一家公司实习，医学影像高速传输系统是该公司开发的一款成熟的与医疗检查设备对接的系统。在上级医院，远程传输模块需要实时下载传输的文件，并且每个医学影像文件利用 dcm4che3 工具包，经过解析后，根据文件的层次结构分类归档，然后把所有文件的层次信息分类记录到配置文件中。同时，为了提高前置机的资源利用率，前置机不被大量远期的检查文件堆积，只存放近期需要调阅的医学影像文件，所以远程传输系统需要自动定期清除超过保存期限的文件。

3.3.1.1　与远程诊断系统对接的接口

根据分级诊疗原则，基层医院的医生遇到疑难杂症时，可以把该患者的病例

资料传输到上级医院，然后上级医院的专家对该患者进行远程诊断。如果基层医院的文件存放的路径是“path/yyyy/MM-dd/checkNum”的格式，path 是文件默认存放的第一级目录，“yyyy/MM-dd/”是该检查文件的检查日期，checkNum 是该患者该次检查的检查号，也是该检查文件的唯一标志符。远程传输模块的配置文件会设置 path 的路径，那么远程传输模块就可以根据检查日期和检查号直接找到待上传的文件从而实现文件的批量上传。

当医生需要批量上传时，远程诊断系统会调用远程传输模块的可执行文件，如图 3.5 所示，同时传递相关参数。参数的内容包括状态值、本医院 ID、医生 ID、检查号、检查日期。远程传输模块的状态机，根据状态值判断文件上传的方式。如果状态值为“1”，则表示需要批量上传，然后把参数中的检查号和检查日期分别放入 Map 集合中。然后再遍历 Map 集合，根据检查号和检查日期，逐一找到文件，然后把文件上传到数据中心。如果出现文件找不到的情况，该模块会弹出对话框显示“文件不存在”。远程诊断系统也可以手动选择上传的检查文件，启动远程传输模块的可执行文件并传递参数，内容包括：状态值为“2”，医院 ID 和医生 ID, 点击“选择文件”，自定义选择检查文件目录，然后点击“文件上传”，远程传输模块就会把该文件上传到数据中心。

医学影像文件上传的过程如图 3.6 所示，首先把检查目录下的文件经过遍历后，筛选出文件名后缀是“.dcm”的医学影像文件，然后利用 dcm4che 工具包，解析出该检查文件的患者基本信息，然后把文件路径放入一个 List 队列中，之后遍历 List 队列，把这些文件压缩为 zip 文件，然后利用 HDFS 提供的 Java API 把文件上传到数据中心。如果文件上传成功，则更新远程诊断系统的数据库中该患者的基本信息存储到数据中心的数据库。然后，通过 JMS 通知远程传输模块的下载端进行实时下载。最后，把已经上传成功的该检查文件的患者信息返回给可执行文件的控件 textArea 中。

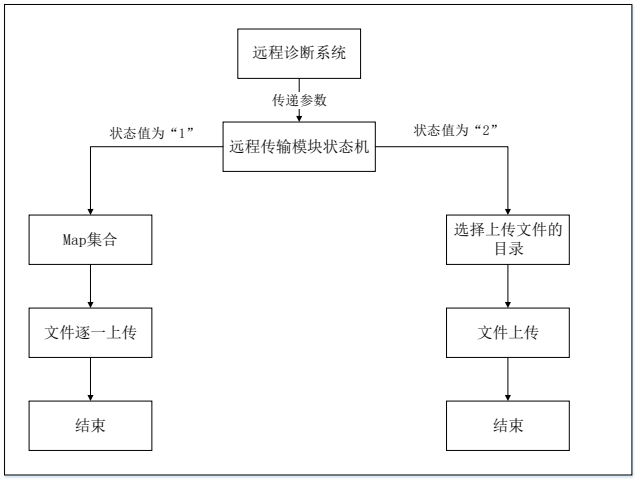

图 3.5　远程传输模块与远程诊断系统对接的接口

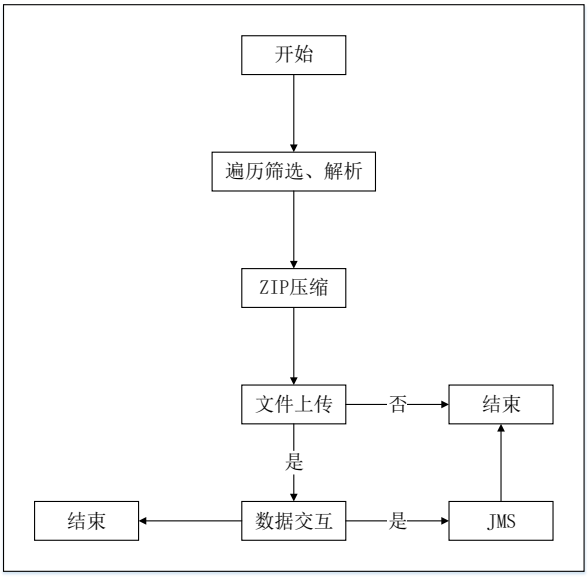

图 3.6　文件上传流程

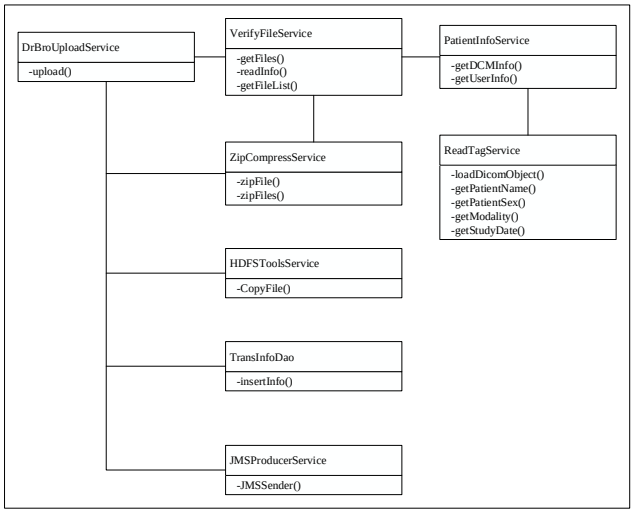

图 3.7　与远程诊断系统对接具体实现类图

上图 3.7 是文件上传实现的类图，实现的具体流程是:

1、远程传输模块根据文件存放路径的规则找到待上传的文件，调用 DrBroUploadService 对象的 upload()方法，开始文件上传的进程。

2、遍历筛选文件，并解析医学影像文件。调用 VerifyFileSerice 对象的 readInfo()方法，首先调用 getFiles()方法遍历文件目录，把检查文件声明为 File 对象，通过 File.listFiles()获取该目录下所有文件的文件列表，然后通过 for 循环遍历该文件列表，通过 File.getName()获取文件名，然后调用 String.endWith()判断文件名是否包含“.dcm”。如果判断结果为是，把该文件添加到一个 FileList 对列里，然后调用 PatientInfoService 类的 getUserInfo()方法解析该医学影像文件。解

析文件，首先调用 ReadTagServie 对象的 loadDicomObject()，把文件转化为 DicomInputStream 对象，然后调用 readDataset()方法读取 DicomInputStream 数据流，然后把结果封装为 Attributes 对象，接着就可以通过 Attributes.getBytes()获得患者的基本信息。如检查号（patientID）、患者姓名（patientName）、患者性别（patientSex）、检查类型（modality）还有检查日期（StudyDate）。然后把这些基本信息，添加到一个数组里，返还给 VerifyFileSerice 对象的 dcmInfo[]数组，同时，把 FileList 队列返还给 VerifyFileSerice 对象的 list 队列。

3、把该文件把文件压缩为 ZIP 文件。接着调用 ZipCompressService 对象的 zipFiles()方法，遍历 list 队列，获取队列中的每一个元素，然后调用 zipFile()方法进行压缩。首先把文件元素转化为 FileInputStream 文件输入流，然后把该文件生成一个 ZipEntry 对象，然后通过 ZipOutputStream.putNextEntry()方法输出。

4、调用 HDFSToolsService 队形的 CopyFile()方法，把文件上传到数据中心。首先把待上传的 ZIP 文件转换为 FileInputStream 文件输出流，然后生成一个 Configuration 实例，通过 FileSystem.get()方法连接 HDFS，在 HDFS 中 create()创建一个空文件，然后通过 IOUtils.copyBytes()把文件复制到数据中心。

5、数据交互。调用 TransInfoDao 接口的 insertInfo()方法，把该检查文件的相关信息记录到数据中心的数据库，

6、给远程传输模块的下载端发送消息，调用 JMSProducerService 对象的 JMSSender()方法，向远程传输模块的下载实时传递消息。实现上级医院实时下载文件。

7、文件上传完毕，通过 DrBroUploadService 对象的 getInfo()方法，把刚上传的检查文件的患者信息返还给前端,并在前端 TextArea 组件中显示。

3.3.1.2 与医学影像高速传输系统对接的接口

基层医院放射科的医疗设备把产生的医学影像文件通过医学影像高速传输系统实时传送到基层医院的前置机上，并按照一定的规则存放。如前文所说，文件路径存放规则是“path/yyyy/MM-dd/checkNum”，“/yyyy/MM-dd/”是检查日期，checkNum 是检查号。如图 3.8 所示，远程传输模块周期性查询基层医院本地数据库 SQL Server，获取当日检查的文件上传列表，然后文件上传列表根据当日已经上传成功的日志列表的记录进行校对。如果日志列表内没有该文件的上传记录，则校对成功，接着查询数据中心的数据库是否存在与该文件的记录；如果存在，则返回上一步，开始下一个文件校对。如果数据中心的数据库不存在该文件的上传记录，则把文件上传到数据中心。如果数据中心的数据库存在该文件的上传记录，则返回，进行文件上传列表的下一个文件的记录校对。文件如果上传失败，

则在错误日志列中记录该文件信息，并在客户端上由显示上传失败的列表，维护人员可以检查错误原因，解决问题后，之后可以手动上传，或者等待下一周期上传。如果文件上传成功，则按照规则生成一个患者 ID,然后根据检查号查询本地数据库该患者的基本信息，然后把这些信息存储到数据中心的数据库。最后通过 JMS 服务通知远程传输系统的下载模块实时下载文件。该模块采取双重校验的方法的原因是：读取 JSON 文件的时间远远小于访问数据库的系统响应时间，尤其在下午，当天做检查的患者总数已经比较多，经过日志列表初次筛查，可以大幅度减少系统开销。同时客户端的传输记录列表显示可以直接从日志文件中获取。

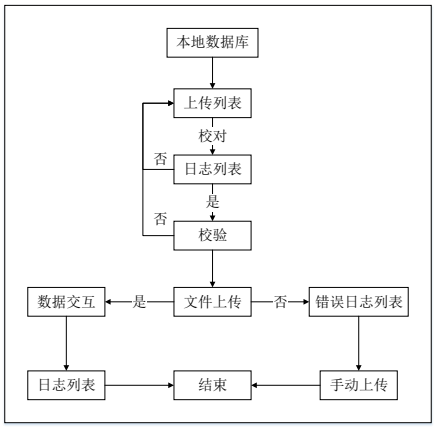

图 3.8 与医学影像高速传输系统的对接方案

图 3.9 是与医学影像高速传输系统对接具体实现类图，实现的具体的流程：

1、开始文件传输。调用 DrSerUpload 对象的 upload()方法，开始文件上传的进程。

2、获取文件上传列表。通过 LocalFileService 对象的 getUploadList()方法，然后调用 FileListDao 接口的 getFileList 方法，通过 FileListMapper 映射文件访问本地数据库，查询当天患者检查的检查号，然后把结果集存放在 List 队列中，并把 List 队列作为返回值返回。

3、与日志列表校对：日志列表是一个 JSON 文件，每天的日志文件以日期命名，每天会生成一个日志文件，同时也会第二天会把前一天的日志文件删掉，所以读取日志文件时，判断文件是否存在，不存在就创建一个文件。首先读取日志文件，调用 uploadRecordService 对象的 readJson()读取日志文件，把结果添加到 Set 集合。然后逐次取出上传列表的元素，判断 Set 集合中是否包含该元素。如果包含该元素，则进行上传文件下一个元素校对，如果不包含，则进行下一步。

4、与数据库进行校验。调用 TransInfoDao 接口的 isExit(),查询数据中心的数据库中的 transDCMInfo 表是否存在该检查号，如果存在就返回上一步，不存在，就可以进行下一步。

5、把文件上传到数据中心。调用 ZipCompressService 对象的 zipFiles()方法，把检查文件中所有的医学影像文件，调用 zipFile()方法进行压缩。然后调用 HDFSToolsService 队形的 CopyFile()方法，把文件上传到数据中心。

6、如果文件上传成功，则进行数据交互。首先调用 LocalInfoSerive 对象的 queryPatientInfo()方法，之后调用 LocalInfoDao 接口，通过 LocalInfoMapper 映射文件查询本地数据库该患者的基本信息，然后调用 PatientInfoDao 接口 savePatientInfo()方法把患者基本信息存储到数据中心的数据库中。

7、如果文件上传失败，通过调用 uploadRecordService 对象的 writeError()方法会把该检查号、检查日期写入错误日志文件中。错误日志文件也是一个 JSON 文件。

8、第 6 步成功后，调用 JMSProducerService 对象的 JMSSender()方法，向远程传输模块的下载实时传递消息。然后返回第 3 步，进行文件上传列表下一个元素的校对。

9、文件上传列表的所有元素都校验完成后，把文件上传列表的所有元素通过调用 uploadRecordService 对象的 writeRecord()方法,记录到日志文件中。然后等待下一个周期获取新的文件上传列表。

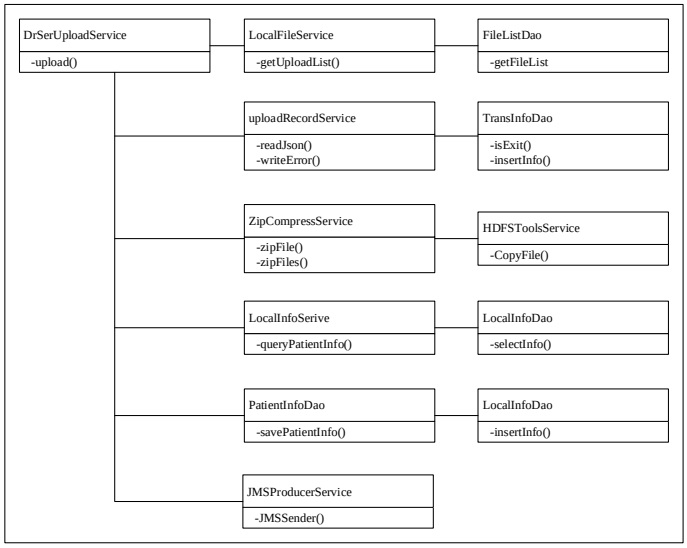

图 3.9　与医学影像高速传输系统对接具体实现类图

3.3.1.3　JMS 服务的实现

本系统的 JMS 服务是通过 Apache ActiveMQ 实现的。远程传输模块采用 ActiveMQ 消息队列的 PTP（Point to Point，点对点）消息模型，如图 3.10 所示。该模型采用异步通信的方式，一个生产者发送消息，然后把消息保存在 Message Queue 队列中，直到有一个消费者接收消息，消息从队列中取出。基层医院是消

息的生产者，上级医院是消息的消费者，基层医院文件上传完毕，通过 ActoiveMQ，向远程传输模块的下载端传递消息，实现上级医院文件实时下载的功能。消息的内容包括，基层医院的患者 ID,文件保存的地址。消息传递的具体流程如图 3.11 所示。

生产者发送消息的流程是：首先用 ConnectionFactory 工厂类创建实例化连接工厂，然后通过 createConnection()方法建立连接，然后启动连接。之后通过 Connection 创建一个会话 session,利用 session 创建一个消息队列 queue 指定消息发送的目的地 destination，通过 session 和 destination 创建一个消息生产者 MessageSender，最后使用 MessageSender 发送消息。

消费者接收消息的流程是：首先用 ConnectionFactory 工厂类创建实例化连接工厂，然后通过 createConnection()方法建立连接，然后启动连接。之后通过 Connection 创建一个会话 session,利用 session 创建一个消息队列 queue 指定消息发送的目的地 destination，通过 session 和 destination 创建一个消息接收者 MessageReceiver， 通过 receive()方法接收消息。

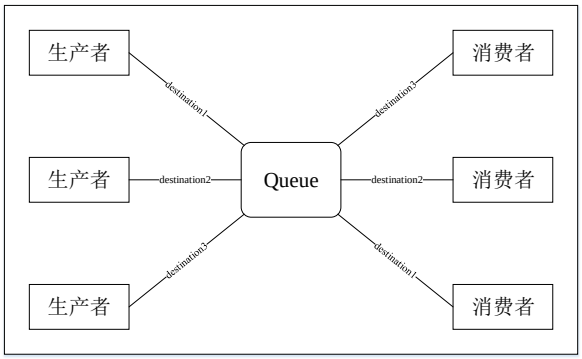

图 3.10　JMS 点对点消息服务模型

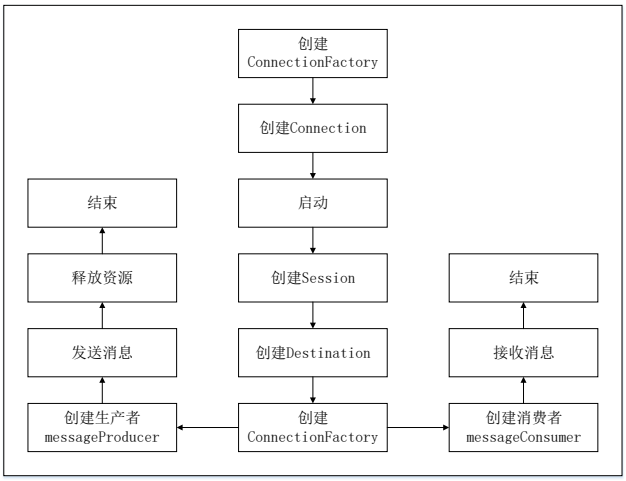

图 3.11　JMS 消息传递流程

3.3.1.4 文件实时下载方案

基层医院文件上传完毕，通过 ActoiveMQ 的消息队列，实时向上级医院传递文件下载的消息。上级医院接收到消息后如图 3.12 所示，首先解析消息内容，根据消息中的文件路径，从数据中心把文件下载到上级医院的前置机上。由于患者的检查文件是以 ZIP 文件的格式存储的，所以文件需要解压缩为普通格式。文件解压缩后，医学影像文件是以无序的方式排列的。患者检查文件中包含成百上千的医学影像文件，这样的凡是不便于医生快速调阅，所以需要对患者文件进行进一步处理。患者检查文件的解析与归档，以及快速调阅医学影像文件的方法见 4.1 小节。

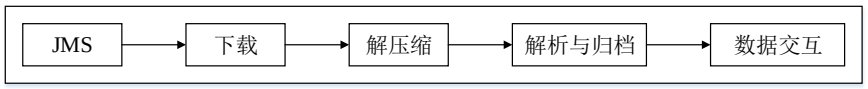

图 3.12　文件下载流程

图 3.13 是文件实时下载的具体实现的类图，具体实现的流程是：

1、实时接收下载消息，时刻监听 ActiveMQ 的消息队列，通过调用 JMSConsumerService 对象的 runJMS(),时刻监听 ActiveMQ 的消息队列，如果有消息队列中有新的消息，就把消息取出并解析。

2、文件下载。远程传输模块需要从每个消息中解析出患者检查文件在数据中心的存放路径，然后调用 FileDownloadService 对象的 download()方法，开始文件下载、解析归档。首先调用 HDFSToolsService 对象 getFile()，连接数据中心的文件系统 HDFS，把文件下载到上级医院前置机的指定路径中。

3、文件解压缩。刚下载的文件是检查文件经过压缩的 ZIP 文件，所以下载的文件需要解压缩。调用 ZipUncompressService 对象的 zipToFile()方法，把下载的文件还原为普通的文件目录。

4、文件解析与归档。调用 CreatFileInfoService 对象的 createInfo()方法，解析每一个医学影像文件然后再根据医学影像文件的层次结构分类归档，最后，把这些关于层次结构的信息分类记录到配置文件中。

5、与远程诊断系统的数据库数据交互，把该检查文件的状态信息改为“已上传图像”。调用 RemoteDiagnoseService 对象的 updateFileStatus()方法，然后调用 RemoteDiagnoseDao 接口的 updateStatus()方法与 RemoteDiagnoseMapper 映射文件，与远程诊断系统的数据库交互。

6、最后，把该传输记录记录到数据中心的数据库。先调用 TransInfoDao 接口的 insertDcmInfo()方法把传输记录记录到数据库。再调用 OperateRecordDao 接口的 instertOperateInfo()方法，把此次操作记录存入数据库。

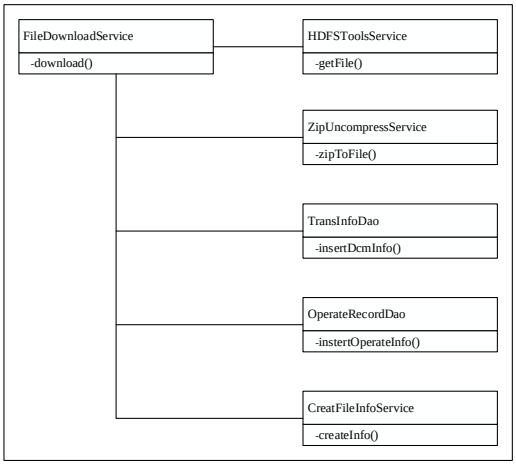

图 3.13　文件实时下载的具体实现的类图

3.3.1.5　前置机远期检查文件定期清除的实现

基层医院和上级医院的前置机上存放着近期可能调阅的医学影像文件，所有的医学影像文件都会在数据中心长期保存。所以为了保证前置机可以长久高效的运行，前置机上存储的图像需要定期删除。基层医院的前置机上文件存放路径的格式是“Path/yyyy/MM-dd/CheckNum”。上级医院的前置机上文件存放路径的格式是“Path/HospitalID/yyyy/MM-dd/CheckNum”。文件清除的原则，首先判断文件目录中检查日期是否超过保存期限，如果超过期限，再判断检查文件目录的创建时间是否超过期限，如果超过期限则删除这个检查文件。同时，当判断过程中如果发现某一级目录下为空文件，则删除这一级文件。本系统默认每天 20 点开始文件清除任务，用户可以设定文件清除的周期，系统默认为两周。远程传输模块会开启一个定时周期线程池，定时开始文件清除。

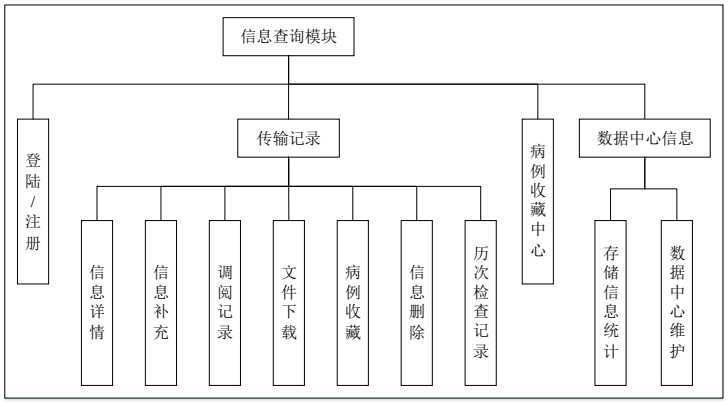

图 3.14　信息查询模块功能结构

3.3.2　信息查询模块

医学影像云存储系统的信息查询模块是基于 B/S 架构开发的，用户可以通过

浏览器根据自己的权限查询相关的信息。信息查询模块的主要功能如图 3.14 所示，信息查询模块的功能分四个模块：登陆/注册模块、传输记录模块、病例收藏中心模块、数据中心信息模块。其中传输记录模块又分七个部分：信息详情、信息补充、调阅记录、文件下载、病例收藏、信息删除、检查记录。

3.3.2.1 登陆/注册模块

系统的用户权限分为三级：系统管理员的权限为一级，可以对用户的注册信息进行审核，可以查看所有基层医院存储信息，比如各医院存储容量，文件上传数量，远程诊断的患者数量，还可以查看数据存储服务各节点的运行情况。上级医院的医生的权限为二级，可以查看与本医院合作的基层医院相关信息，比如传输记录、存储信息，还可以收藏特殊病例，并能在收藏中心查看收藏记录并能导出收藏记录。基层医院的医生只能查看与本医院相关的信息。

用户在登陆页面输入自己的账号和密码后，点击“登陆”，登陆页面会向 UserloginController 对象发出登陆请求，然后调用 exitInfo()方法向 UserInfoService 对象发出请求判断用户账号和密码是否匹配， UserInfoService 对象通过 UserInfoDao 接口与数据库交互，如果用户信息存在，则调用 HttpSession 对像的 setAttribute()方法把用户信息存放在 Session 中，然后重定向至信息查询模块主页面。如果用户信息不存在，则显示账户或密码错误。当重定向至主页面后，调用 UserloginController 的 getUser()方法从 session 中得到用户信息，然后调用 UserInfoService 对象的 getAccessLevel()获得权限等级，然后封装为 JSON 对象返回给 web 页面,然后根据这些信息渲染页面。

3.3.2.2 传输记录模块

用户可以无条件或者多个条件进行多重条件查询传输记录，检索条件有 6 种：医院名称、检查日期范围、身份证号、患者 ID、检查号、患者姓名。其中根据登录权限，基层医院的“医院名称”检索条件默认为本医院名称，上层医院可以选择任意基层医院，没有选择任何基层医院，系统默认为全选。检查日期范围系统默认为当天，也可以选择三天内，一周内，也可以自定义时间范围。用户可以通过身份证号查询该患者的历次检查记录。针对每一条传输记录，用户可以进行 7 种操作：信息详情、信息补充、调阅记录、文件下载、病例收藏、信息删除、检查记录。信息详情、调阅记录、病例收藏四种操作，都会弹出模态框，模态框内会显示相关信息。信息补充主要补充患者的基本信息，该操作包含在信息详情中。

信息详情操作，会在模态框中显示的内容有：患者的基本资料（身份证号，患者 ID、检查号、姓名、性别、年龄、家庭住址、医保号、手机号还有患者信息的第一次登记时间和最近一次信息更新时间）、患者的检查信息（诊断状态信息、

归属医院，患者类型、检查登记时间、检查类型、检查时间）、检查文件的存放信息（文件是否存在，文件是否损坏）。

信息补充操作：在信息详情模态框中，对患者的基本信息进行补充，如身份证号，姓名、家庭住址、医保号、手机号。患者的其他信息则不能修改。

调阅记录操作：该操作会显示文件所有的传输记录，包括上传或下载此检查文件的用户 ID,归属医院 ID,操作类型，操作时间、存放路径。

文件下载操作：这个操作是通过 JMS 服务向远程传输模块的发送消息，远程传输模块把检查文件从数据中心下载到本医院的前置机。

病例收藏操作：如果医生遇到特殊病例，可以收藏该病例资料，便于开展日后科研工作。用户点击“病例收藏”，可以对该病例添加备注信息，包括疾病类型，检查部位，医嘱信息，诊断信息。其中，疾病类型和检查部位，可以在下拉框菜单中选择。

信息删除操作：删除此条传输记录，这个操作慎重使用。

历次检查记录操作：患者的历次检查资料越多，诊断医生的误诊率就越低。所以，此操作可以刷新传输记录的列表，只显示同一个身份证号的患者检查记录，并对每条记录进行各种操作。

为实现传输记录模块的功能，图 3.15 是传输记录模块的类图，TransferRecordController 类定义了负责传输记录模块各种操作的方法，tranDCMInfoServiceimp 类继承了 tranDCMInfoService 接口的所有方法，tranDCMInfoService 接口是处理业务逻辑的接口，主要负责查询数据库中的记录信息，更新患者基本信息、删除记录。tranDCMInfoDaoimpl 类继承了 tranDCMInfoDao 接口的所有方法，tranDCMInfoDao 接口主要负责与数据库 transDCMInfo 表进行数据交互。CollectCaseServiceimpl 类继承了 CollectCaseService 接口的所有方法，CollectCaseService 接口是处理业务逻辑的接口，负责存储病例收藏的相关信息。CollectCaseDaoimpl 类继承了 CollectCaseDao 接口的所有方法，CollectCaseDao 接口负责与数据库 CollectFileRecord 表进行数据交互。OperateRecordServiceimpl 类继承了 OperateRecordService 接口的所有方法，OperateRecordService 接口是处理业务逻辑的接口，负责获取该检查文件调阅记录的相关信息。OperateRecordDaoimpl 类继承了 OperateRecordDao 接口的所有方法，OperateRecordDao 接口负责与数据库 OperateRecord 表进行数据交互。JMSProduceServiceimpl 类继承了 JMSProduceService 接口的所有方法，JMSProduceService 接口主要作用是作为消息生产者，通过 ActiveMq 的消息队列，向远程传输模块发送文件下载的消息。

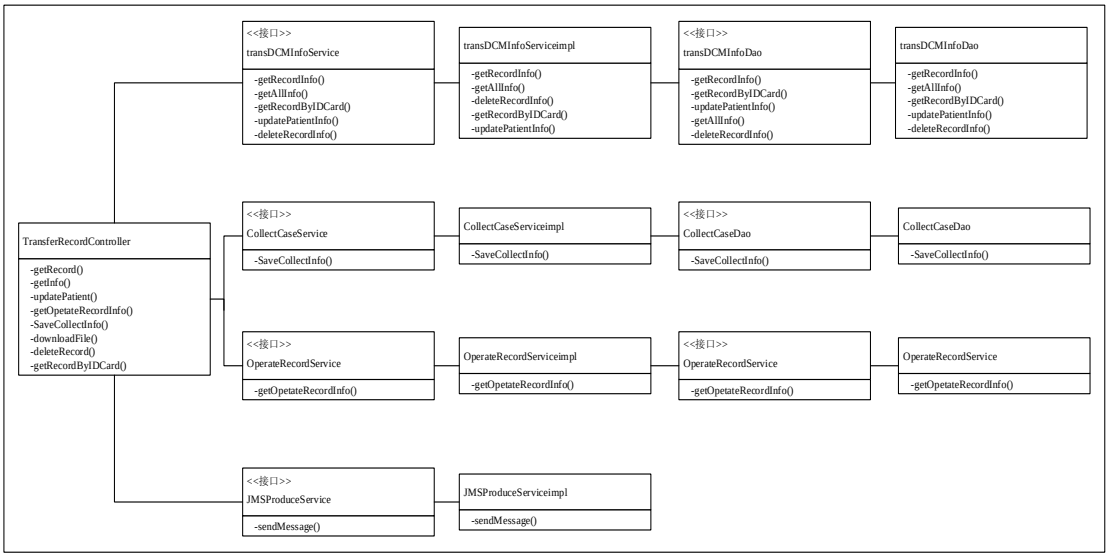

图 3.15　传输记录模块的类图

3.3.2.3　病例收藏中心功能

用户可以查询个人收藏的病例信息。针对每一条收藏记录，有两种操作，文件下载，删除信息。用户还可以把全部收藏记录导出并生成 Excel 文件。文件下载通过 JMS 服务，向远程传输模块的下载端发送消息，下载端会把文件下载到本地前置机上。为实现病例收藏中心的功能，需要定义 CollectCaseController 类，CollectCaseService 接口, CollectCaseServiceimpl 类，JMSProduceService 接口，JMSProduceServiceimpl 类，CollectCaseDao 接口，CollectCaseDaoimpl 类，transDCMInfoDao 接口，transDCMInfoDaoimpl 类。下图是病例收藏中心模块的类图：

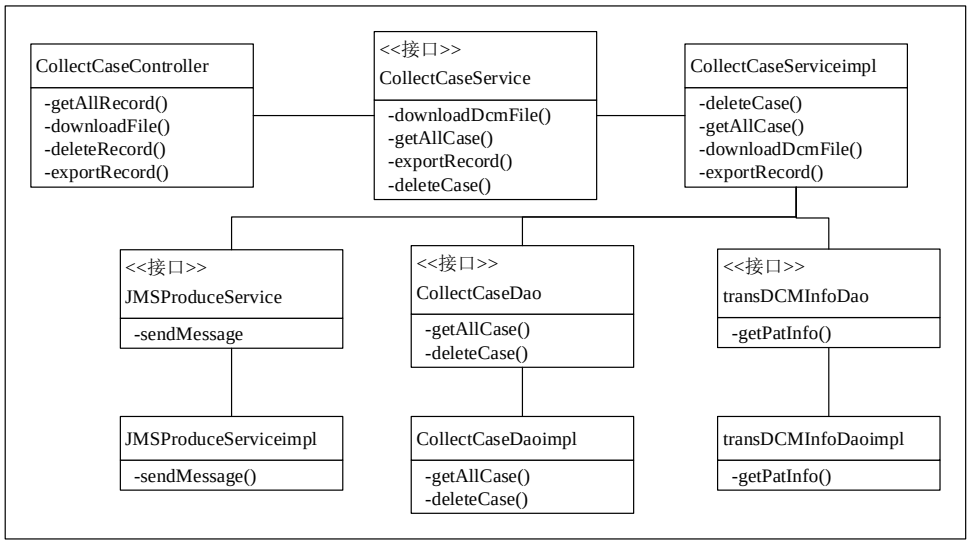

图 3.16　病例收藏中心模块类图

CollectCaseController 类中有查询所有的收藏记录的方法 getAllRecord()，文件下载的方法 DownloadFile,删除记录的方法 deleteRecord(),还有导出收藏记录的方法 exportRecord()。CollectCaseServiceimpl 类继承了 CollectCaseService 接口的所有方法，CollectCaseService 是病例收藏模块的业务逻辑接口。JMSProduceServiceimpl 类继承了 JMSProduceService 接口的方法，JMSProduceService 接口负责通过 ActiveMQ 的消息队列，向远程传输模块发送消息，实现文件下载的业务逻辑接口。CollectCaseDaoimpl 类继承了 CollectCaseDao 接口的所有方法，CollectCaseDao 接口负责与数据库 CollectFileRecord 表进行数据交互，获取收藏记录或者删除收藏记录。transDCMInfoDaoimpl 类继承了 transDCMInfoDao 接口的所有方法，transDCMInfoDao 接口负责与数据库 transDCMInfo 表进行数据交互，获取患者信息。

3.3.2.4 数据中心信息模块

数据中心模块主要功能是查询基层医院存储情况的统计信息，数据中心数据存储服务各节点的运行情况，还可以控制数据存储服务进行节点间的数据均衡。基层医院的存储情况的统计信息包括，在一定时间范围内基层医院在数据中心的存储总容量，文件个数和患者数量。用户登陆后，可以根据用户的权限等级查询相关信息。为实现数据中心信息模块的功能，需要定义 DataCenterController 类，DataCenterService 接口，DataCenterServiceimpl 类，HadoopToolService 接口，HadoopToolServiceimpl 类，transDCMInfoDao 接口，transDCMInfoDaoimpl 类。图 3.16 是数据中心信息模块的类图：

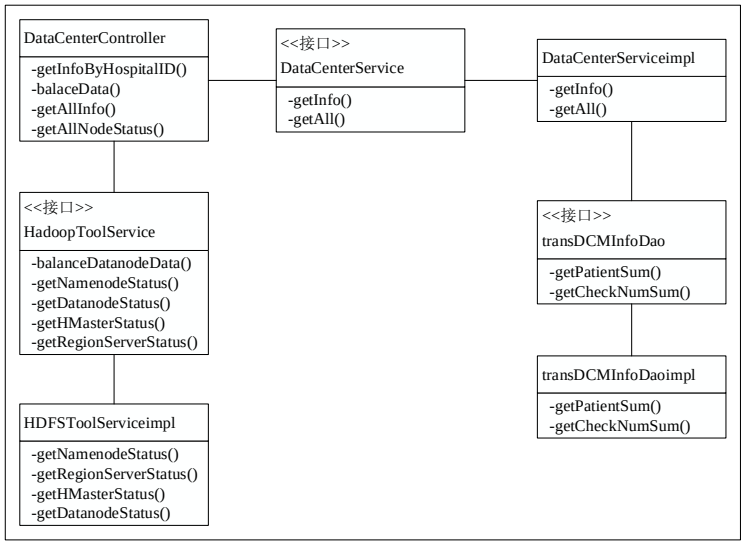

图 3.16 数据中心信息模块的类图

DataCenterController 类中有通过医院 ID 查询该医院的统计信息的方法 getInfoByHospitalID()，数据均衡的方法 balanceData(),查询所有医院存储情况的

统计信息的方法 getAllInfo()，获取数据存储服务节点的运行状态的方法 getAllNodeStatus()。DataCenterServiceimpl 类继承了 DataCenterService 接口的所有方法，DataCenterService 是数据中心信息模块的业务逻辑接口，处理从数据库获取相关信息的业务逻辑。HadoopToolServiceimpl 类继承了 HadoopToolService 接口的方法，HadoopToolService 接口负责与数据存储服务交互的接口，获取关键节点的运行状态。transDCMInfoDaoimpl 类继承了 transDCMInfoDao 接口的方法 transDCMInfoDao 接口负责与数据库进行数据交互，根据基层医院的医院 ID 获取身份证号数量和检查号的数量，即患者数量和文件数量。

3.4 数据库设计

系统不仅需要把检查文件从基层医院传输到上级医院，而且需要相关信息记录到数据库中，便于查询传输记录以及其他操作。图 3.17 是系统的 E-R 图 (Entity Relationship Diagram，实体-联系图)。

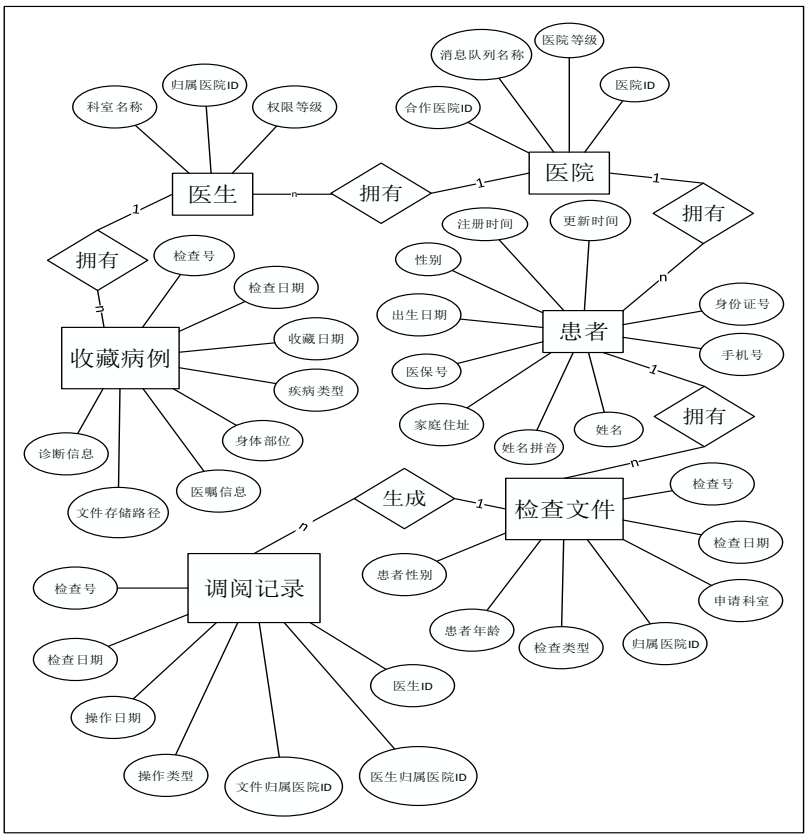

图 3.17　实体 E-R 图

在面向列的数据库 HBase 中，每条记录信息都是根据行的键值 RowKey 排序的，一个合理有效行行键将可以大大提高数据库的查询效率。并且，HBase 中的数据存储类型为字节数组，即 byte[]。系统比较重要的表如下方所示。

User 表为用户表，表的主键为 hospitalID+doctorID,即医院 ID 与医生 ID 的组合,具体情况如下表。

表 3.1　数据库 User 表

行键	Info 列族	
hospitalID+doctorID	info:hosName	本医院名称
	info:hosLevel	本医院级别
	info:docLevel	医生级别
	info:cosHos	合作医院
	info：jmsDestination	归属医院 JMS 目的地

TransDCMInfo 表为传输记录表,表的主键是 idcard+checkNum，即身份证号与检查号的组合，具体情况如下表。

表 3.2　数据库 transDCMInfo 表

行键	patientInfo 列族		dcmInfo 列族	
idcard+checkNum	patientInfo:patientID	患者 ID	dcmInfo:modality	检查类型
	patientInfo:patName	患者姓名	dcmInfo:studyDate	检查时间
	patientInfo:patNamePy	姓名拼音	dcmInfo:departmentID	申请科室 ID
	patientInfo:age	患者年龄	dcmInfo:hospitalID	归属医院 ID
	patientInfo:gender	患者性别	dcmInfo:localPath	本地存储地址
	patientInfo:patBirthDate	出生日期	dcmInfo:dataPath	存储路径
	patientInfo:address	住址	dcmInfo:transDate	上传日期
	patientInfo:telephone	手机号码	dcmInfo:transTime	上传时间
	patientInfo:regtime	注册时间	dcmInfo:uploadflag	上传状态
	patientInfo:updatetime	更新时间	dcmInfo:downloadflag	下载状态

CollectFileRecord 表为文件收藏记录表，表的主键为 hospitalID + dockerID + checkNum,即医院 ID、医生 ID 和检查号的组合。

表 3.3　数据库 collectFileRecord 表

行键	record 列族	
hospitalID+	record:studydate	检查日期
dockerID+	record:collectDate	收藏日期
checkNum	record:datapath	数据中心存储地址
	record:diseaseType	疾病类型
	record:bodyPart	身体部位
	record:doctorAdvice	医嘱信息
	record:diagnosticInfo	诊断信息

OperateRecord 表为文件调阅记录表，表的主键为 dockerID +checkNum,即检查号与医生 ID 的组合。

表 3.4　数据库 OperateRecord 调阅记录表

行键	operate 列族	
dockerID+c	operate:hosIDOfDoc	医生所属医院 ID
heckNum	operate:hosIDOfFile	文件所属医院 ID
	operate:studyDate	检查时间
	operate:operationType	操作类型
	operate:operationTime	操作时间

3.5　数据中心的设计与搭建

3.5.1　数据中心的整体设计

医学影像云存储系统的数据中心是在 Docker 容器基础上搭建的新型 PaaS。数据中心目前提供三种服务，JMS 服务、Web 服务和数据存储服务，如图 3.18 所示。JMS 服务通过 ActiveMQ 消息中间件实现的。Web 服务，即信息查询模块，是通过轻量级 Web 服务器 Tomcat 进行网页发布，同时，为了提高服务的并发性能，Web 服务加入了负载均衡。负载均衡是通过高性能的反代理服务器 Nginx 实现的。数据存储服务采用了分布式文件系统 HDFS、分布式数据库 HBase 和分布式协调服务 Zookeeper。为了保证数据存储服务的远程访问安全，远程访问采用 SSH 安全网络协议。

普通的服务器出现宕机的情况是很常见的，如果分布式文件系统 HDFS 的

Namenode 节点所在的服务器宕机，那么数据存储服务就无法访问了。分布式数据库 HBase 也面临这样的问题。为了避免这样的情况，数据中心的数据存储服务采用高可用的搭建方式如图 3.19，分布式文件系统 HDFS 采用“active-standby”的模式，即“活动-备用”模式，使用两个 Namenode,活动 Namenode 接收请求，处理请求，备用状态的 Namenode 随时接替活动状态的 Namenode 的工作，并且通过共享日志的方式，与活动状态的 Namenode 时刻保持同步。

而 HBase 集群的高可用依赖 Zookeeper 集群实现的，同时开启两个 HBase 的主节点 HMaster,HBase 出现故障，通过 Zookeeper 实现负载均衡，主节点切换[30]。集群的容灾方案详见 4.3 节。

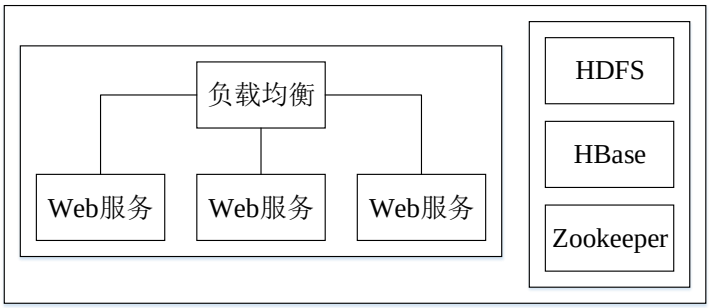

图 3.18　数据中心的整体架构

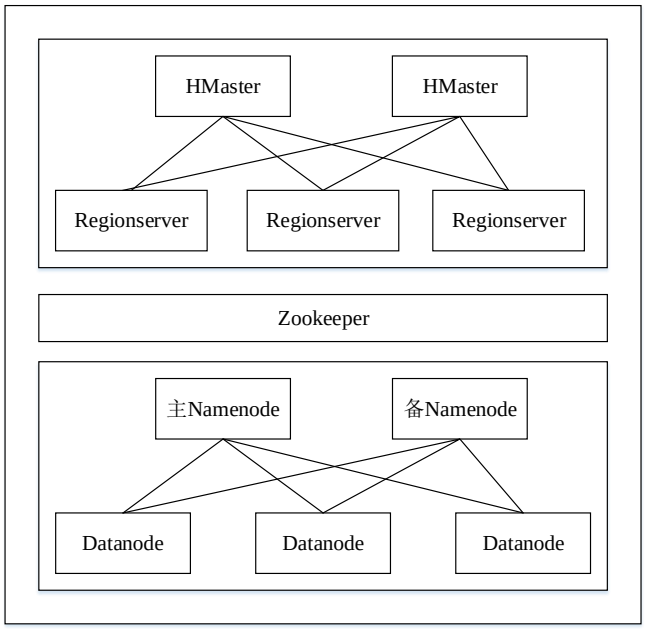

图 3.19　数据中心的高可用方案

3.5.2　数据中心搭建的具体方案

数据中心是基于 Docker 搭建的，所以首先需要制作一个包含 JDK 的基础镜像。本系统的基础镜像采用 Ubuntu16 系统和 Java8。然后我们可以从各官网上

下载各组件的安装包。

3.5.2.1 JMS 服务的搭建

JMS 服务是通过 ActiveMQ 中间件实现的，所以我们需要在 Docker 的基础上，安装 ActiveMQ。具体搭建方法如下：

（1）我们在 slave120 电脑上启动一个基础镜像容器，从官网上下载 apache-activemq-5.10.5-bin.tar.gz。

（2）然后把该安装包解压到/usr 中，保存该镜像副本名为 activemq。

（3）在刚生成的镜像副本的基础上，启动 activemq 容器，注意把 61616 和 8161 两个端口映射出来。

（4）启动 ActiveMQ。

3.5.1.2 Web 服务的搭建

我们用三台电脑（master118,master119,slave120）模拟带有负载均衡的 Web 服务，负载均衡采用权重轮询的负载策略。

（1）首先在 master118 电脑上，从 docker 官网上拉取最新的镜像，如 docker pull tomcat , docker pull nginx。

（2）启动一个 Tomcat 容器，把容器中 Tomcat 的端口映射到宿主机上，我们可以使用命令

```
docker run -d -p 8081:8080 --name tomcat1    tomcat
```

（3）我们把信息查询模块的项目文件复制到 Tomcat 的子文件 webapps 下。

（4）把这个容器，生成一个镜像副本,起名为 webServer，发送到本地 docker 仓库上。

（5）在 master118 上部署 nginx 负载均衡，首先启动一个 nginx 容器。

（6）然后配置 nginx 相关信息，本系统采用权重轮询的方式，weight 默认为 1，数值越大，负载的权重就越大;backup 表示其它所有的非 backup 服务器宕机或者忙碌的时候，请求 backup 服务器。修改 nginx.conf 文件，在 http{}中加入一下内容：

```
server {
        listen 80;
        location   / {
                proxy_pass http://blance;
        }
 }
 upstream blance{
```

server 192.168.2.118:8081 weight=3;
server 192.168.2.119:8082 weight=1;
server 192.168.2.120:8083 backup;
}

（7）然后分别在另两台电脑上，拉取 webServer 镜像，然后启动 webServer 容器，根据配置把相应的端口号映射出去。

3.4.1.3 数据存储服务的搭建

数据存储服务由三台电脑（master118，master119，slave120）组成，两台电脑 master118，master119 作为 master 节点，同时，三台电脑也作为 slave 节点，利用 docker 容器技术，可以模拟搭建两个 master 节点，三个 slave 节点的集群模式。

首先安装配置 Zookeeper，具体流程是：

（1）首先从官网上下载一个 zookeeper-3.4.13 安装包。

（2）然后用基础镜像在 master118 电脑上，启动一个基础镜像的容器。

（3）把安装包复制到/home/hadoop/目录下，然后创建文件夹/home/hadoop/zookeeper/data/zkData。

（4)开始配置 zookeeper 相关信息,conf 目录下修改 zoo.cfg 文件。加入 dataDir 的路径，即刚创建的/home/hadoop/zookeeper/data/zkData；然后添加 zookeeper 集群主机及端口号，注意，节点数必须是奇数，例如：

slave.1 = master118:2888:3888
slave.2 = master119:2888:3888
slave.3 = slave120:2888:3888

（5）在 data/zkData 目录下，创建文件 myid,然后添加内容 1，表示在配置文件 zoo.cfg 中指定的 slave.1。

（6）然后保存为一个镜像副本取名为 zookeeper,然后提交在私人仓库，然后再其他机器上拉取 zookeeper,启动 zookeeper 容器。

（7）修改其他电脑上的 myid,根据第（4），（5）步。

（8）然后启动 zookeeper 集群，启动文件是 bin 目录下的 zkServer.sh。

然后我们搭建 HDFS 集群。

（1）把从官网上下载的 hadoop-2.8.5.tar.gz 解压在基础镜像的容器中。

（2）SSH 配置。在容器中安装 SSH 安装成功后，然后创建一个公钥对，然后把这个公钥复制到其他 hadoop 容器 ~/.ssh/authorized_keys 文件中，可以实现无密码登陆。

（3）在容器../hadoop-2.8.5 目录下，创建三个文件夹 tmp, namenode,datanode.

（4）配置 Hadoop 相关信息。我们主要修改../hadoop-2.8.5/etc/hadoop 目录下的文件.

配置运行环境，修改 hadoop-env.sh 文件中 JAVA_HOME 指定容器中 jdk 的安装地址。

修改 core-site.xml 文件，hadoop 高可用模式中连接到 nameService 的属性。

```
<property>
  <name>fs.defaultFS</name>
  <value>hdfs://ns1</value>
</property>
```

然后临时文件夹 hadoop.tmp.dir 的目录路径，如

```
<property>
        <name>hadoop.tmp.dir</name>
        <value>/home/hadoop/hadoop-2.8.5/tmp</value>
</property>
```

然后指定 zookeeper 集群的地址和端口,如：

```
<property>
  <name>ha.zookeeper.quorum</name>
  <value>master118,master119:2181,slave120:2181</value>
</property>
```

修改 hdfs-site.xml 文件，指定 dfs.replication（数据块的副本个数，个数不可以大于 Datanode 节点数，一般设置 3 个）;为 Namenode 集群设定一个 service name，属性名 dfs.nameservice ,值为 ns1; nameservice 包含 namenode 名字，属性名 dfs.ha.namenodes.ns1,值为 master118,mater119；master118 的 namenode 的 rpc 地址和端口号，rpc 用来和 datanode 通讯 ，属性名 dfs.namenode.rpc-address.ns1.master118，值为 master118:9000；master119 的 namenode 的 rpc 地址和端口号，属性名 dfs.namenode.rpc-address.ns1.master119，值为 master119:9000；master118 的 namenode 的 http 地址和端口号，用来和 web 客户端通讯，属性名 dfs.namenode.http-address.ns1.master118，值为 master118:50070；master119 的 namenode 的 http 地址和端口号，属性名 dfs.namenode.http-address.ns1.master119，值为 master119:50070；namenode 之间用于共享编辑日志的 journal 节点列表，属性名为 dfs.namenode.shared.edits.dir ，值为 qjournal://master118:8485;master119:8485;slave120:8485/ns1;指定该集群出现故障时，是否自动切换到另一台 namenode,属性名 dfs.ha.automatic-failover.enabled.ns1，值为 true;journalnode 上

用于存放 edits 日志的目录,属性名 dfs.journalnode.edits.dir，值为/home/hadoop/hadoop-2.8.5/tmp/ data/dfs /journalnode；客户端连接可用状态的 NameNode 所用的代理类，属性名 dfs.client.failover.proxy.provider.ns1，值为 org.apache.hadoop.hdfs.server.namenode.ha.ConfiguredFailoverProxyProvider；一旦需要 Namenode 切换，使用 ssh 方式进行操作,属性名 dfs.ha.fencing.methods，值为 sshfence；如果使用 ssh 进行故障切换，使用 ssh 通信时用的密钥存储的位置，属性名 dfs.ha.fencing.ssh.private-key-files，值为/home/hadoop/.ssh/id_rsa；connect-timeout 超时时间，属性名 dfs.ha.fencing.ssh.connect-timeout，值为 30000。

修改 mapred-site.xml 文件，采用 yarn 作为 mapreduce 的资源调度框架。属性名 mapreduce.framework.name，值为 yarn。

修改 yarn-site.xml 文件，启用 HA 高可用性，属性名 yarn.resourcemanager.ha.enabled，值为 true；resourcemanager 的名字，属性名 yarn.resourcemanager.cluster-id，值 yrc；使用了 2 个 resourcemanager,分别指定 Resourcemanager 的地址，属性名 yarn.resourcemanager.ha.rm-ids，值为 rm1,rm2；指定 rm1 的地址，属性名 yarn.resourcemanager.hostname.rm1，值为 master118；指定 rm2 的地址，属性名 yarn.resourcemanager.hostname.rm2，值为 master119；指定当前机器 master188 作为 rm1，属性名为 yarn.resourcemanager.ha.id，值为 rm1；指定 zookeeper 集群机器，属性名 yarn.resourcemanager.zk-address，值为 master118:2181,master119:2181,slave120:2181；NodeManager 上运行的附属服务，默认是 mapreduce_shuffle，属性名 yarn.nodemanager.aux-services，值为 mapreduce_shuffle；

（5）修改 slaves 文件，指定节点位置。

master118
master119
slave120

（6）我们把配置好的 hadoop，像通过运行指令 docker commit 生成一个副本，把该副本上传到自己的 Docker 私有仓库上,

（7）在其他两台电脑上从私有仓库拉取下来。Master118 电脑开启一个名为 master118 的容器，一个名为 slave118 的容器；Master119 电脑开启一个名为 master119 的容器，一个名为 slave119 的容器；slave120 机器开启一个名为 slave120 的容器。启动的时候，需要把用到的端口号都通过-p port 指令与宿主机的端口进行映射。

（8）配置各节点容器 IP 地址。修改每个容器中/etc/hosts 文件，指定各节点的 IP 地址。

（9）在 master119 电脑上修改 master119 容器中的 yarn-site.xml 文件，把 master119 作为 rm2,属性名 yarn.resourcemanager.ha.id，值为 rm2。同时删除 slave120 容器中的该属性对，因为 slave120 不是 ResourceManager。

（10）在 master118 容器中启动 Journalnode,启动脚本 hadoop_2.8.5/sbin/ hadoop-daemon.sh。

（11）格式化 Namenode 节点与 ZKFC。前者运行的指令是：hdfs namenode -format，后者运行的指令是： hdfs zkfc -formatZK。注意，第一次启动前需要格式化，但是之后格式化前，需要把之前创建的 namenode , datanode , tmp 先删除后重新创建。

（12）master120 电脑作为备用主节点，需要备用主节点同步主节点元数据。

（13）在 master118 电脑上 master118 容器中运行 sbin 目录下的脚本：/start-dfs.sh、/start-yarn.sh、hadoop-daemon.sh start zkfc。

在 master119 电脑上的 master119 容器中启动 ResourceManager,sbin 目录下，运行指令 yarn-daemon.sh start resourcemanager。

(14)在 master118 电脑上的 master118 容器上，查看个节点的运行状态。

在 HDFS 集群的基础上，搭建 HBase 集群。

（1）我们利用基础镜像启动一个容器，然后把 hbase-1.3.3.tar.gz 解压

（2）配置 HBase.修改/conf 目录下 hbase-env.sh、hbase-site.xml、regionservers 三个文件。

（3）hbase-env.sh 文件配置环境信息，JAVA_HOME 指向 JDK 目录路径；HBASE_MANAGES_ZK 设置 false，表示禁用 HBase 自带的 zookeeper，false 指使用独立的 zoopkeeper；保存 pid 文件，制定给 HBASE_PID_DIR 属性为 /home/hadoop/data/hbase/pids。

hbase-site.xml 文件配置，设置 HRegionServers 共享目录，属性名 hbase.rootdir，值为 hdfs://master118:9000/hbase，指定 Hmaster 主机，属性名 hbase.master ， 值 为 hdfs://master188:60000 ； 启 用 分 布 式 模 式 hbase. cluster.distributed ，值为 true；指定 Zookeeper 集群位置，属性名 hbase. zookeeper.quorum，值为 master118:2181,master119:2181，slave190:2181；
指定独立 Zookeeper 安装路径，属性名 hbase.zookeeper.property.dataDir，值为 /home/hadoop/zookeeper-3.4.11 ； 指 定 ZooKeeper 集 群 端 口 ， 属 性 名 为 hbase.zookeeper.property.clientPort，值为 2181。

（4）修改 regionservers 文件，指定 RegionServers 所在机器：

master118

master119

slave120

（5）创建 pid 文件保存目录，在/home/hadoop/目录下，运行指令：mkdir data/hbase/pids -p

（6）生成一个新的镜像副本，然后上传到私有仓库，并拉取到 master119 的电脑上。

（7）在 master118 电脑,启动容器，在容器中运行脚本 start-hbase.sh。master119 作为备用主节点，master119 启动容器，然后运行命令 hbase-daemon.sh start master。

3.6 本章小结

本章从实际的需求进行分析，然后根据需求分析，进行了系统的总体设计，接着详细介绍了系统各模块具体设计与实现方案，包括远程传输模块的设计与实现、信息查询模块的设计与实现，数据库的设计，以及数据中心的设计与搭建。本论文围绕“一个中心，两个点”的网络架构，实现了医学影像文件从基层医院传输到数据中心，然后再实时下载到上级医院，同时用户可以通过浏览器查询相关信息。

4 系统的关键技术

第三章详细阐述了医学影像云存储系统的总体设计与各模块的具体实现，本章节在第三章基础上，就系统开发的关键问题，给出详细的解决方案。本章主要解决的问题有:1、如果高速精准调阅文件数量比较多的医学影像文件。2、数据存储服务会随时间的推移，或因为过度忙碌，或者因为新增 Datanode 节点，出现数据块分布不均匀的情况，如何远程控制数据存储服务进行数据均衡。3、因为服务器是廉价的服务器搭建的，那么如果数据中心的主节点宕机，系统如何能保证正常运行。4、大量的文件需要传输时，那么远程传输模块如何高效安全地把文件从基层上传到上级医院。

4.1 医学影像归档及快速调阅方法的改进

一个患者做一次检查会产生成百上千的医学影像文件。医生可以获取更详细的信息进行精准诊断，但是同时也存在一些问题，如何快速调阅医学影像文件。由第二章第二小节可知，每个医学影像文件都有一个标准的 tag 信息，其中包括患者的基本信息，图像的像素信息，还有序列信息，研究信息。本系统利用开源的 dcm4che 工具包解析医学影像文件。Dcm4che3 是遵循 DICOM3.0 标准，基于 Java 的工具开发包。根据医学影像文件的文件层次结构，如图 2.6 所示，一个患者的检查文件，有一个或者几个研究 studyUID，每个研究对应多个序列 seriesUID，每个序列对应若干个医学影像文件。其中每个序列的医学影像文件是有序的。所以医学影像归档的方法是，解析医学影像文件的 tag 信息，根据每个医学影像文件的 studyUID，seriesUID 分类归档，如图 4.1 所示。

文件归档后就是医学影像文件快速调阅的问题，解决办法：把医学影像文件的 studyUID,seriesUID 等信息持久化。如图 4.1 所示，这是解析归档后的文件目录 1.2.840.113704.1.111.2656. 1498891623.1 目录下，存放着同一个 studyUID 的医学影像文件。Baseinfo 文件存放着患者的检查号和 accessionNum。seiesinfo 目录下存放着医学影像文件 tag 信息的配置文件如图 4.2 所示。图 4.3 中文件名带有“study”字符的文件是一个 studyUID 的 series 信息配置文件。图 4.4 中该 studyUID 下共有 144 个医学影像文件，3 个序列 seriesUID。图 4.1 中，文件名不带“study”字符的文件是 series 的配置信息。图 4.4 中，该 series 包含 75 个医学影像文件，以及每个文件的文件名。文件名后面有一个分号加一个数字，数字代表该医学影像文件在该序列中的排序位置。从这几个图中可以看出，一个患者检查文件的医学影像文件 Tag 信息有一点复杂，存储到数据库，很容易造成数据冗余，并且与其他专业医学影像文件阅读软件对接时很不方便。本节介绍的调阅方法，

不仅可以最精准最快速有序地获取每个医学影像文件，而且与其他医学影像文件阅读软件对接时，更加方便快捷。

医学影像文件解析与归档的方法具体实现：远程传输模块通过 dcm4che3 工具包，把每一个标准的医学影像文件转化为 DicomInputStream 流,然后调用 DicomInputStream.readDataset()方法读取 Tag 信息，并把 tag 的元数据转化为 Attributes 对象，然后就可以 getBytes ()方法逐一获取各种信息。其中，解析 Tag 信息时，会出现乱码的情况，遇到此情况，应该用“gb18030”编码格式。然后根据层次结构，把文件先转化为 FileInputStream 文件输入流，然后通过文件输出流 FileOutputStream，分类归档相应目录中。同时，文件的 studyUID、seriesUID、文件名等信息放入对应的 Map 集合与 List 队列结合的数据结构中，当所有的文件解析完后，把这些信息记录到 seriesinfo 目录下的对应配置文件中。

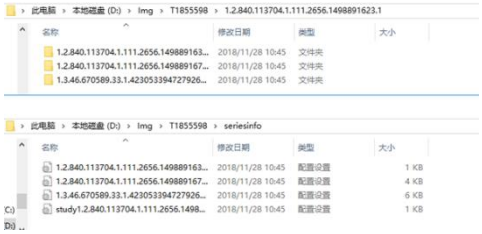

图 4.1　医学影像文件归档实例

图 4.2　医学影像文件分级归档第一级目录

图 4.3　医学影像文件 Study 配置信息

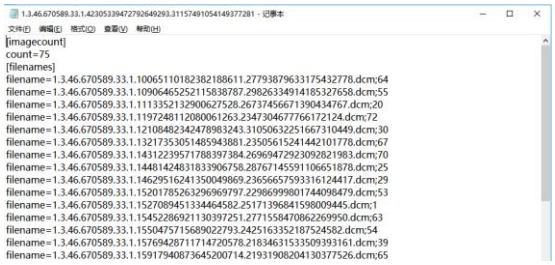

图 4.4　医学影像文件 Series 配置信息

4.2　HDFS 集群数据均衡的实现

分布式文件系统 HDFS 凭借高容错性，易扩容等的特性，在市场上得到了广泛应用。但是随着时间的推移，一个集群经常会遇到添加新节点、剔除旧节点，或者某个节点存储容量已经饱和，会造成 Datanode 节点间数据块的分布越来越不均衡，导致部分 Datanode 节点特别忙碌。为了保证数据存储服务正常高效的运行，本系统有两种方式实现 Hadoop 集群的数据均衡。

第一种方式为，系统管理员可以在活动状态的 Namenode 节点上，在终端启动 Hadoop 守护进程——均衡器，启动指令: % start-balancer.sh。可以添加-threshold 参数指定阈值（可选，默认为 10%），以判定集群是否均衡。无论在什么时候，一个集群中只能有一个均衡器在运行。均衡器工作原理是：均衡器从 Namenode

获取各个 Datanode 磁盘使用情况，然后确定需要移动的数据块，然后根据数据块分配原则，把数据块分散存放到不同节点，然后不停地移动数据块，直到集群均衡为止。均衡器会把每次数据块移动的记录写入到日志文件中。为了不干扰集群正常运行，均衡器在后台运行，且带宽受限制，默认为 1MB/s。

第二种实现方式，本系统在信息查询模块设置了数据均衡功能，系统默认每天 20 点进行数据均衡。数据均衡的具体实现方式是，开启一个定时周期线程 ScheduledExecutorService，第一次执行任务的时间是当天的 20 点，之后每逢 24 小时执行任务。数据均衡任务首先通过 HdfsConfiguration()连接 HDFS 文件系统，然后调用 ToolRunner.run()启动一个守护线程，然后调用 Balancer 实例的 Cli()方法，开始数据均衡。

4.3 数据中心的容灾方案

4.3.1 数据损坏与恢复

分布式文件系统 HDFS 把文件以数据块的方式存储到各个 Datanode 节点上，那么数据块出现损坏的情况也是很常见的。为了保证医学影像文件的完整性，我们需要对损坏的数据块进行自动恢复。我们可以用 Hadoop 提供的 fsck 工具检查系统的文件的保存状况。fsck 工具可以检索所有 Datanode 节点中缺失的数据块和副本个数过多过少的数据库。fsck 工具的工作原理：循环遍历 Namenode 节点的命名空间，然后检查所有找到的文件，然后获取文件数据块的元数据并指出问题所在。也可以通过该工具，查找一个文件的数据块。

在信息查询模块，用户可以在传输记录模块的信息详情中，显示检查文件是否保存完好。该功能的具体实现：首先通过 HdfsConfiguration()方法连接 HDFS 文件系统，然后调用 FsckServlet 实例的 doGet()方法。

如果出现数据块损坏的情况，数据存储服务可以通过 Datanode 数据块扫描器检测和修复数据块。数据块扫描器可以通过 dfs.datanode.scan.period.hours 属性设置定期检查的时间。每个 Datanode 节点均运行一个数据块扫描器，定期检测该节点上存放的数据块，损坏的数据块被报告给 Namenode 并及时修复数据块。

4.3.2 数据存储服务的单点失效问题

本论文设计的数据存储服务由分布式文件系统 HDFS、分布式数据库 HBase、分布式协调服务 Zookeeper 搭建的。为了避免当服务器宕机的时候，用户无法远程访问文件系统，系统采用高可用的方式避免单点失效的问题。HDFS 集群中如果 Namenode 出现单点失效问题，由前文可知，活动状态的 Namenode 故障，会

自动转移到备用状态 Namenode。这种自动切换的功能，是通过二者共享编辑日志实现的。Namenode 会把每次文件的操作记录到编辑日志，所以备用 Namenode 只需要把编辑日志读取一遍，就可以替代活动状态的 Namenode，用户的远程访问不会受到太大的影响[30]。

活动状态的 Namenode 和备用状态的 Namenode 的数据保持同步，需要一组独立进程的 journal 节点时刻记录编辑日志文件的改动,如图 4.5 所示。活动状态的 Namenode 的节点把所有的操作记录同步到 journal 节点，然后备用状态的 Namenode 时刻监控并读取 journal 节点的修改信息，并同步到自己的命名空间中。同时，Datanode 必须把文件的数据块信息同时发送给两个 Namenode,这样备用状态的 Namenode 可以时刻待命，代替活动状态的 Namenode。为了实现这样的过程，需要保证两个 Namenode 的硬件配置是一样的，journal 节点可以在不同的服务器上运行，但是节点数必须超过 3 个，而且要数量是奇数。

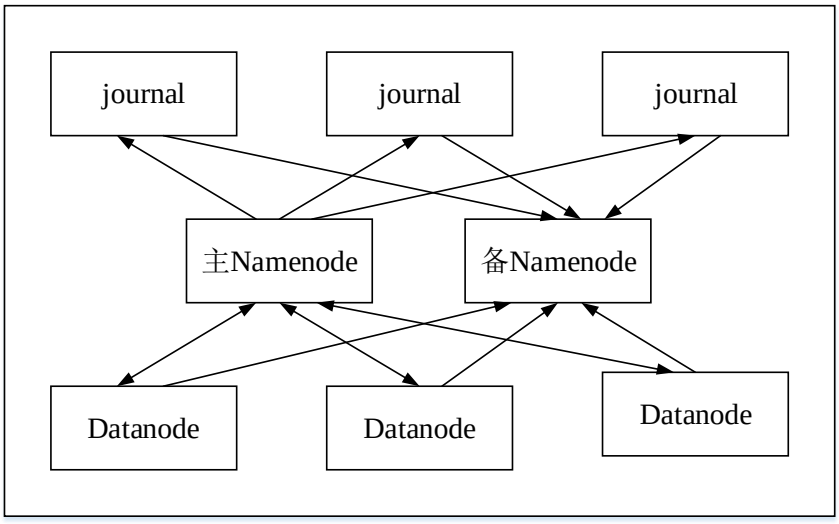

图 4.5　HDFS 的高可用机制

HBase 集群的高可用机制是依赖 Zookeeper 和 HDFS 实现的，首先 HBase 的数据会把数据存储到分布式文件系统 HDFS 中，之前我们提到 HDFS 把文件以数据块的形式存储在数据存储服务中，并且有 3 个副本，所以 Regionserver 节点会由于 HDFS 的容错机制，不会造成数据丢失。

HBase 的主节点 Hmater 的主要负责管理数据表 table 和区域 Region 的数据信息，远程访问数据库都不需要主节点 Hmaster 参与，所以它的负载很低。但是 Hmaster 是不可或缺的，为了保证 HBase 集群能正常运行，可以运行两个 Hmaster，Zookeeper 保证同一时刻只有一个 Hmaster 运行，并且遇到 Hmaster 故障，立马切换另一个。每次数据访问，Regionserver 都会把操作记录下入 Hlog 日志里，所以，HMaster 只需读取 Hlog 文件，备用状态的 Hmaster 就可以接替活跃的 Hmaster.

4.4 远程传输模块的高并发解决方案

针对基层医院，一个患者做一次放射性检查（如 CT、核磁共振），一般需要二十多分钟，一家基层医院的放射科的医疗检查设备的数量仅有几台。在此情况下，系统不仅要保证文件能及时上传到数据中心情况，还要尽可能占用最少的前置机的资源。因此，与医学影像高速传输对接的接口需要周期性查询本地数据库，然后采用一个定长的线程池，把文件上传到数据中心。

该接口采用 ScheduledThreadPoolExecutor 周期性执行任务。如图 4.4 所示，在主线程中，生成一个 ScheduledThreadPoolExecutor 线程池实例，然后定时查询数据库获得文件上传列表，然后经过信息校验，把所有需要上传的文件信息封装为 ScheduledFutureTask 实例，然后放入 DelayQueue 中，线程从 DelayQueue 中获取已到期的 ScheduledFutureTask，然后逐次把文件遍历后筛选出医学影像文件，然后再压缩成 ZIP 文件，然后把 zip 文件上传到数据中心。根据实际情况 ，三条线程并发执行文件上传的任务已经满足需要，所以三条线程把文件长传完毕后，修改 ScheduledFutureTask 的 time 变量，确定下次定时获取列表的时间。

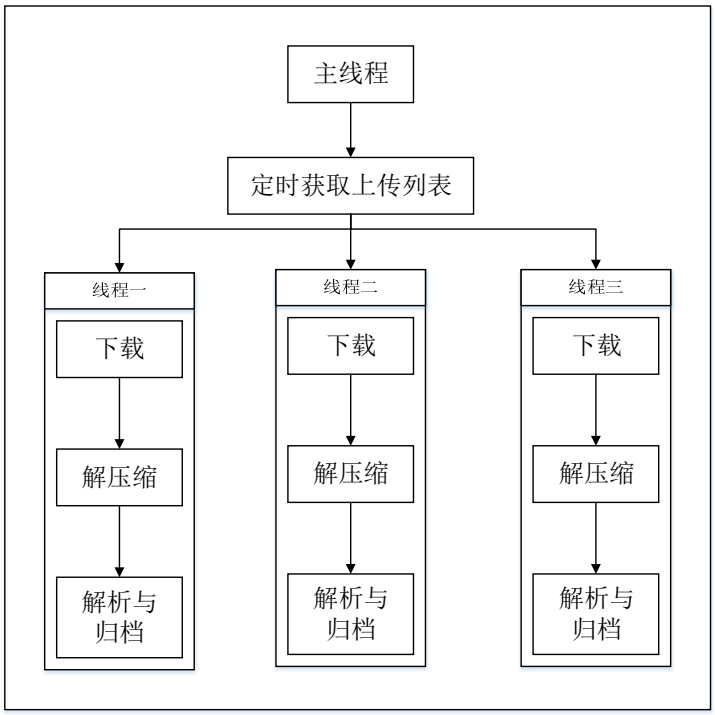

图 4.4 远程传输模块的线程池文件上传

远程传输模块下载端需要实时地把文件从数据中心下载到本地医院的前置机上，需要实时接收通过 ActiveMQ 传递的消息。为了减少系统创建线程，销毁线程的开销，提高系统的传输效率，所以在下载端加入了定长的线程池。这样的设计的原因是，前置机的配置不需要很高，当任务过多，创建的线程过多，会占用前置机大量的内存，会造成卡顿，甚至是崩溃。

远程传输模块的下载端采用 FixedThreadPool 定长线程池，如图 4.5 所示，限制线程最大为 3 条，当有新的 JMS 消息时，会创建一个线程，进行文件的下载、解压缩和解析与归档。如果正在执行任务的线程达到 3 个时，JMS 消息会放入 workQueuqe 阻塞队列中等待，直到有线程任务执行完毕，空闲时，则该 JMS 消息会从阻塞队列中被取出，然后执行该任务。

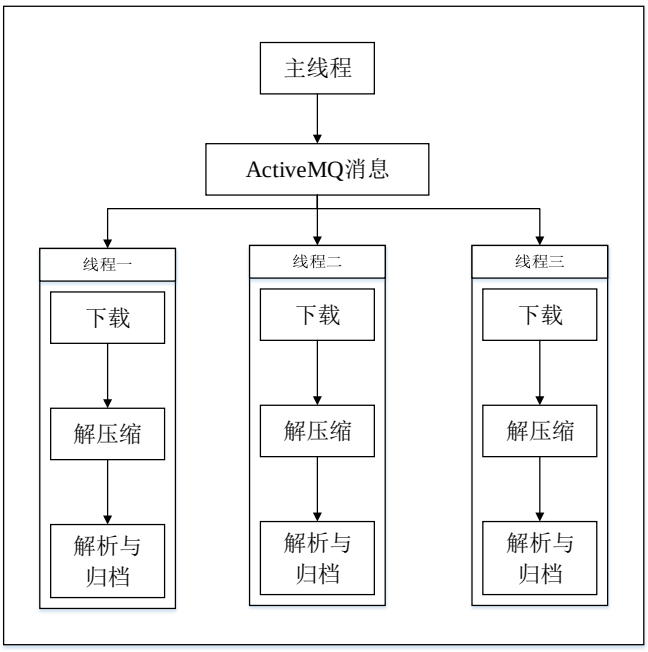

图 4.5　远程传输模块的下载端线程池文件下载

4.5　本章小结

本章重点介绍了医学影像云存储系统几个比较关键的问题，包括医学影像文件的解析归档和调阅的问题，HDFS 集群的数据均衡问题，以及数据中心的容灾方案，还有远程传输模块的高并发解决方案。

5 系统测试

5.1 测试环境

本章将对医学影像云存储系统的各个模块进行功能性测试和性能测试。系统的测试环境是本人用实验室的五台电脑搭建的，数据中心由电脑 master118、电脑 master119、电脑 salve120 搭建的，电脑 1 作为基层医院的前置机，电脑 2 作为上级医院的前置机。表 5.1 是系统测试环境的硬件配置，表 5.2 是系统测试所需要的软件配置。

表 5.1 系统测试环境硬件配置

	master118、master119、salve120	电脑 1、电脑 2
操作系统	Ubuntu 系统	Win10
内存	4GB	4GB
硬盘容量	1TB	1TB
显卡	Inter 集成显卡	Inter 集成显卡

表 5.2 系统测试环境软件配置

软件名	版本
Docker	1.11.0
Hadoop	2.8.5
Zookeeper	3.4.13
HBase	1.3.3
Nginx	1.3.16
Tomcat	8.5
jdk	1.8
SQLServer	2008
医学影像高速传输系统	
模拟医疗设备软件	

5.2 远程传输模块的测试

5.2.1 功能性测试

医学影像云存储系统的远程传输模块分为三部分，与远程诊断系统对接的接口、与医学影像高速传输系统对接的接口、以及文件实时下载端。

与远程诊断系统对接的接口的操作流程：用户在网页上启动可执行文件并传递参数，接口解析参数，并跳转相应的页面。如果参数传递的状态值为“1”,则表示在默认路径下的多个检查文件一起上传，如果状态值为“2”，则表示医生需要手动选择上传的检查文件，点击“选择文件夹”按钮，然后选择待上传的文件夹，然后点击“开始上传”按钮，文件开始上传，进度条开始滚动变化，上传的动态信息显示在下方的显示框中。远程传输系统的功能性测试结果，如表 5.3 所示。

表 5.3　与远程诊断系统对接接口的功能性测试

测试项目	预期效果	实际效果
启动可执行文件	可以正常运行	正常
文件批量自动上传	接口可以根据传递的参数自动把多个检查文件上传	正常
选择上传的文件夹	可以选择文件夹，文件夹目录选中后回在输入框中显示	正常
开始上传文件	文件开始上传，并在下方的方框中显示动态信息	正常
文件上传进度显示	进度条在随着文件开始上传开始变化，上传结束后，停止变化	正常
与远程诊断系统数据交互	从远程诊断系统的数据库读取数据，并存储到数据中心的数据库	正常

与医学影像高速传输系统对接的接口，负载把基层医院放射科的医疗检查设备产生的所有医学影像文件周期性传输到数据中心。表 5.4 为与医学影像高速传输系统对接的接口的功能性测试情况。

表 5.4　与医学影像高速传输系统对接的接口的功能性测试

测试项目	预期效果	实际结果
修改前置机基本信息，本地相应的配置文件并保存一致	修改完信息，点击“保存”本地相应的配置文件也随之改变	正常
文件目录地址设定	可以选则文件保存的目录	正常
医院存储情况统计	存储情况扇形统计，正常显示	正常
上传记录更新	上传记录列表可以自动更新	正常
下载记录更新	下载列表可以自动更新上传记录	正常
信息查询可以多条件查询	点击“查询”按纽，根据查询条件，把传输记录在表格中显示	正常
信息查询可以重置查询条件	点击重置，全部恢复初始条件	正常

远程传输模块的下载端负载把基层医院传输到数据中心的检查文件，实时下载到上级医院的前置机上，然后遵循 DICOM3.0 标准，把每一个医学影像文件进行解析归档。

表 5.5　远程传输系统下载模块功能性测试

测试项目	预期效果	实际结果
修改前置机基本信息，本地相应的配置文件并保存一致	修改完信息，点击“保存”按钮，本地相应的配置文件的信息能保持一致	正常
文件目录地址设定	可以选则文件保存的目录	正常
医院存储情况统计	存储情况扇形统计，正常显示	正常
下载列表更新	下载列表可以自动更新上传记录	正常
信息查询可以多条件查询	点击“查询”按纽，根据查询条件，把传输记录在列表中显示	正常
信息查询可以重置查询条件	点击重置，全部恢复初始条件	正常

5.2.2　性能测试

模拟设备端的软件，可以模拟医疗检查设备，设置文件传输的间隔时间，并且重复修改医学影像文件患者的基本信息，发送一个患者的检查文件；高速传输系统接收设备产生的图像。远程传输系统把文件传输到上级医院的前置机上。

测试的方法：不同大小的检查文件分别被模拟设备端的软件，发送给医学影像传输系统，并且每次都修改医学影像文件的 Tag 信息，时间间隔为 10s，然后上传 10000 次。在局域网下，传输带宽为 8MB/S 的条件下，测试结果的平均时间如上表所示，测试中没有出现内存泄露的问题，并且 CPU 占用率保持在 40%。虽然在局域网中，远程传输模块的传输速度不如医学影像高速传输系统，但是在广域网中，带宽不理想的条件下，由于远程传输模块以 ZIP 文件为传输单位，节省了大量的系统响应时间，所以远程传输模块的传输效率将优于医学影像传输系统。本系统也进行了性能测试，测试结果如下：

表 5.6　远程传输系统性能测试结果

检查文件	高速医学影像传输系统传输时间	远程传输模块上传时间	远程传输模块下载时间
42.2MB，215 个文件	37s	42s	30s
28.3MB，149 个文件	29s	33s	25s
340MB，694 个文件	5min20s	5min35s	5min10s

5.2.3　效果展示

图 5.1 是与远程诊断系统的对接接口，负责单个和批量的检查文件上传。

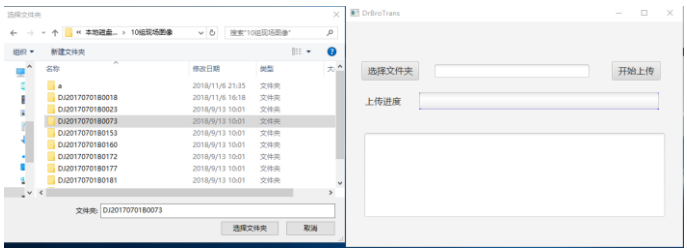

图 5.1 与远程诊断系统的对接接口效果图

图 5.2 是与医学影像高数传输系统的对接接口，负责把基层医疗设备产生的医学影像文件上传到数据中心。

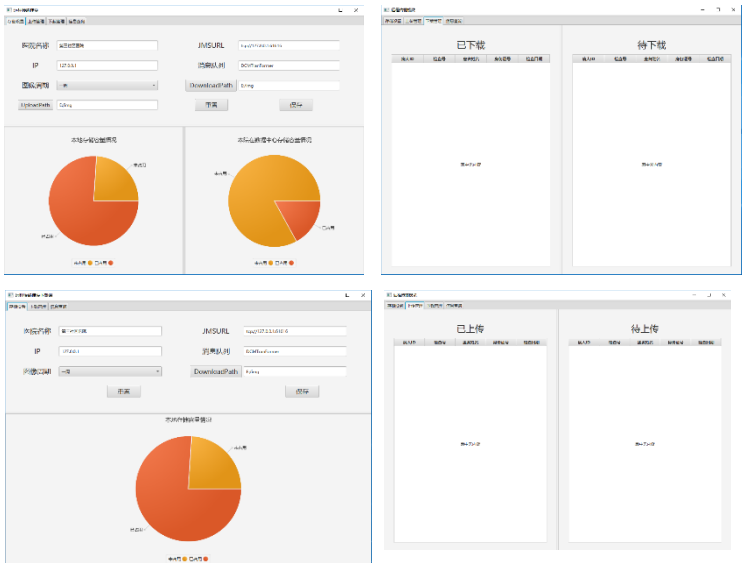

图 5.2 与医学影像高数传输系统的对接接口

图 5.3 是远程传输模块下载端的效果图，负责实时接收检查文件。

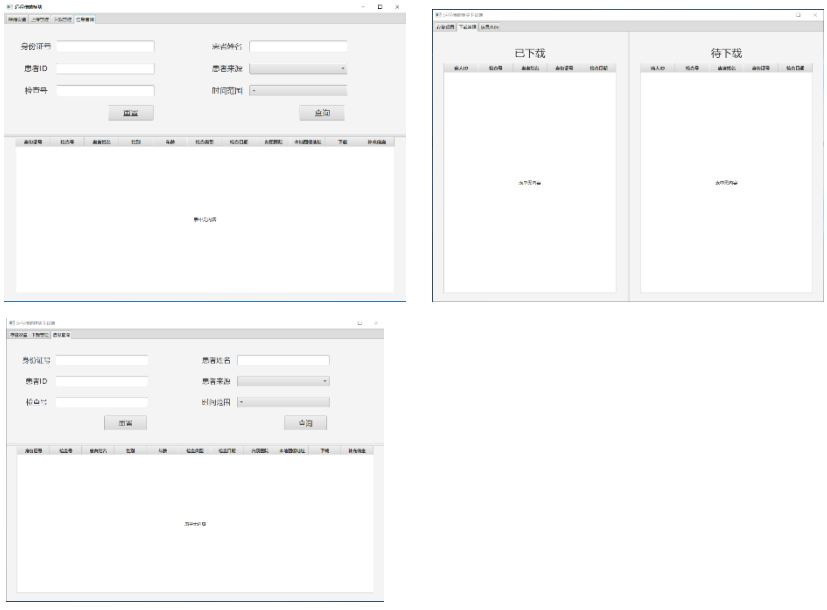

图 5.3 远程传输模块的下载端

5.3 信息查询模块测试

5.3.1 功能性测试

本小节将对医学影像云存储系统信息查询模块的各个模块进行功能性和性能测试。下面的几个表分别是信息查询模块进行功能性测试的结果。

表 5.6 信息系统传输记录查询功能性测试

测试项目	预期结果	实际结果
多条件查询传输记录	查询结果在表格中显示	正常
用户可以自定义选择时间范围	用户可以选择查询时间的开始和截止日期	正常
用户可以选择系统设置的几个时间范围进行查询	可以按照选择“当天”、“三天内”、“一周内”的时间范围进行查询	正常
查询条件重置	点击“重置”按钮，查询条件会恢复到初始状态。	正常
查看信息详情	点击每条记录的“更多”下拉框选择“信息详情”，弹出模态框，显示详细信息。	正常
修改患者信息	在“信息详情”的模态框中，修改患者基本信息后，点击“保存”，数据库对应信息会保持一致。	正常
显示调阅记录	弹出模态框，并显示该文件的调阅记录	正常
文件下载到本地前置机	文件可以从数据中心下载到本地前置机上。	正常
收藏特殊病例	可以对该病例添加“疾病类型”、“身体部位”、“医嘱信息”、“诊断信息”等备注信息	正常
信息删除	删除此条记录	正常
查询历次检查记录	可以显示该患者多次检查的文件相关信息	正常

表 5.7 信息查询系统传输记录查询功能性测试

测试项目	预期结果	实际结果
医生收藏病例记录的显示	可以显示医生所有病例的收藏记录	正常
删除单个收藏记录	可以删除该条收藏记录	正常
下载收藏病例的医学影像文件	可以把该患者的医学影像文件下载到本地前置机	正常
可以导出收藏记录	可以把所有的收藏记录导入到 Excel 文件中	正常
查询患者多次检查记录	弹出模态框，显示该患者的所有的检查记录	正常

表 5.8　数据中心信息的功能性测试

测试项目	预期结果	实际结果
查询本医院的存储情况统计	可以显示本医院的存储情况	正常
查询数据中心运行情况	可以显示数据中心各节点的运行状态	正常
远程数据均衡	可以远程控制数据存储服务的数据存储节点进行数据均衡	正常

5.3.2　性能测试

针对本系统的 web 服务——信息查询模块的性能测试，系统采用 Apache 开发的 JMeter 压力测试工具，进行并发模拟测试。测试采用不同数量的线程并发，对相同的操作进行重复运行。测试结果如下：

表 5.9　数据中心信息的性能测试结果

并发数量	操作次数	完成时间	占用内存	平均响应时间	失败率
100	1000	36s	16%	1.3s	0
400	1000	110s	26%	1.6s	0
700	1000	283s	35%	2.4	0
1000	1000	396s	38%	2.9	0.0034

5.3.3　效果图展示

图 5.4 是信息查询系统的查询查询记录的页面。

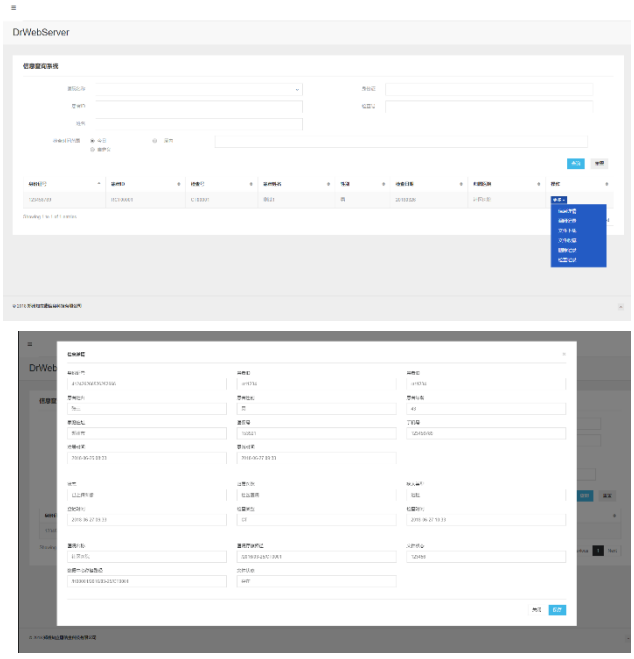

图 5.4　信息查询系统的传输记录的效果图

图 5.5 是病例收藏中心的页面。

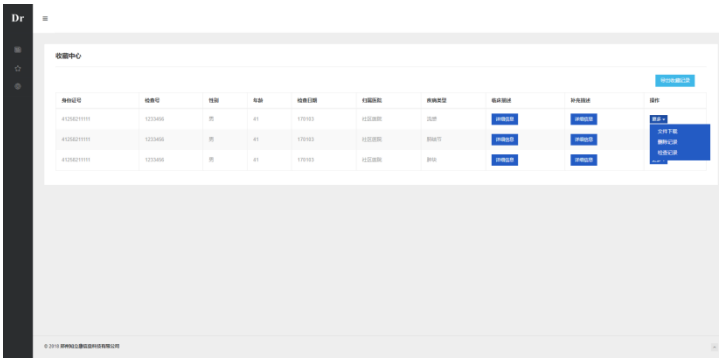

图 5.5　病例收藏中心的效果图

图 5.6 是数据中心统计情况的页面。

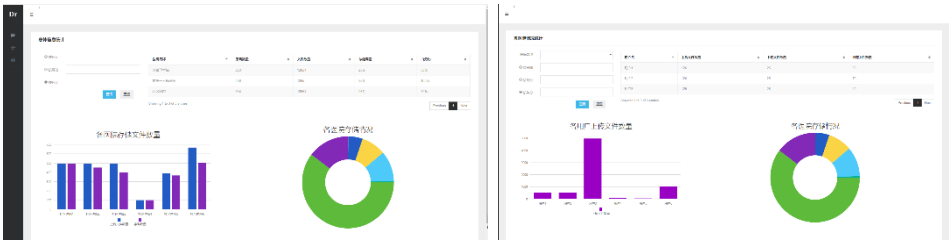

图 5.6　数据中心统计情况效果图

5.4　本章小结

本章对医学影像云存储系统的功能性和性能性进行了测试，首先介绍了一下测试的环境，然后详细介绍了各功能测试的项目，预期结果和实际结果。系统的测试结果满足设计的预期效果，可以通过验收。

6 总结与展望

本章将对系统的工作进行总结，根据分级诊疗的实际需求，对于系统需要解决的问题进行分析，并提出对应的解决方法。并对系统的不足以及后期的工作进行展望。

6.1 总结

目前，我国医疗资源分布还不均匀，不发达地区的患者无法享受优质的医疗资源，而三甲医院的专家每天都被常见疾病所围绕，有限的医疗资源没有得到充分利用。国家连续出台多个政策鼓励大力发展分级诊疗制度，合理有效利用有限的优质医疗资源。而落实分级诊疗制度，最基本、最关键的问题是，基层医院需要把患者的检查文件传输到上级医院。

随着数字成像技术的发展，“软读片”在各大城市的各大医院得到普及，虽然降低了患者做检查的成本，但是同时带来了患者检查文件的文件数量越来越多的问题。这就造成普通的文件传输方式无法满足现实需要，并且医院每天都会产生海量的医学影像文件，造成传统的集中存储的方式无法满足现在医学影像文件爆发式增长的需求。所以本论文围绕“传输难，存储难”的问题，设计了医学影像云存储系统。

本人在实习公司参与医学影像云存储系统的研发，从现实需求出发，到系统架构设计，数据库设计，以及各功能模块的实现，完成了系统的开发工作。

首先本论文采用“一个中心，两个点”的网络架构，即一个数据中心，上级医院和基层医院分别设置的前置机。前置机用来存储近期需要调阅的医学影像文件，并且可以与基层医院的内部 PACS 系统进行数据交互。数据中心采用以 Docker 为基础的新型 PaaS（Platform as a Service,平台即服务）云平台。这样就完成了应用与 PaaS 平台的解耦问题。开发人员只需要把应用和依赖的框架中间件用 Docker 打包成镜像，这样平台不需要准备对任何语言的支持，也不需要将应用打包的复杂过程。为解决海量的医学影像文件存储难的问题，数据中心采用分布式文件系统 HDFS、分布式数据库 HBase、分布式协调服务 Zookeeper，搭建了高可用、易扩容、高容错的数据存储服务。数据存储服务的 HDFS 采用“活动-备用”模式，两个 Namonode 同时运行，保证活动的 Namenode 节点出现单点失效问题，备用状态 Namenode 可以随时代替活动的 Namenode;HBase 依赖 HDFS 和 Zookeeper 实现高可用，HMaster 出现单点失效的问题，Zookeeper 可以自动切换主节点，如果存储数据损坏，可以从 HDFS 中读取其他副本。数据中心还提供了

Web 服务、负载均衡、以及 JMS 消息服务。Web 服务通过 Web 服务器 Tomcat 发布的，负载均衡通过反代理服务器 Nginx 实现。而 JMS 服务通过消息中间件 ActiveMQ 实现的。

其次，系统需要解决医学影像文件传输难的问题。医学影像文件遵循 DIOCOM 标准，文件比较小，数量比较多的特点，市场上的流行浏览器无法满足文件远程传输的需求，所以系统的远程传输模块采用 C/S 架构。为了实现“高内聚、低耦合”的开发原则，远程传输模块加入了 Spring 框架，Mybatis 框架和 Phoenix 框架。

为了避免把他人的检查文件上传的错误，远程传输模块需要对检查文件进行遍历和解析，利用 dcm4che3 工具包解析医学影像文件，并把患者的基本信息显示到前端文本框。

为提高传输效率，降低系统开销，远程传输模块把检查文件中的医学影像文件压缩为 ZIP 文件，然后文件上传完毕，通过 ActiveMQ 消息中间件，实时通知上级医院下载文件；并且与医学影像高速传输系统对接接口，采用了 ScheduledThreadPoolExecutor 定时周期线程池技术把设备产生的医学影像文件并发上传；下载模块采用了定长线程池技术，并发处理文件。

为了其他医学影像阅读软件高速精准调阅每一个医学影像文件，远程传输模块利用 dcm4che3 工具包，解析每一个医学影像文件，并根据文件层次结构，分类归档到对应的 StudyUID、SeriesUID 目录下，并且把这些 StudyUID、SeriesUID、文件名等信息分类输出到对应配置文件下。这样，阅读软件可以花费最小的代价用最快的速度精准加载每一个医学影像文件。为了避免前置机被大量远期的文件堆积，远程传输模块启动定时周期线程，定时清除文件。

最后，系统为了方便用户查询相关信息，系统的信息查询模块采用 B/S 架构开发的 Web 服务。同时为了方便代码后期维护和快速迭代，信息查询模块加入了 SpringMVC 框架、Spring 框架和 Phoenix 框架。

6.2 展望

由于时间和条件有限，本系统还有一些不足之处，需要进一步完善：

1、系统需要一个统计缴费模块，根据存储容量，基层医院可以向数据中心支付一定的维护费用，用来长期存储文件。

2、系统还没有充分利用 Hadoop 组件的优越性，下一步可以对医学影像文件进行更全面的解析，数据中心增加关于医学影像文件的其他服务可以省略这些基础的解析工作。

3、系统在性能测试方面还不够充分，在系统安全方面，面对恶意攻击，还没有更完善的方案。

4、随着分级诊疗的普及，系统的业务快速增长，数据中心可以在现有的基础上开展多租户服务。

5、系统采用 Phoenix 框架实现用 JDBC 访问 HBase 数据库，下一步需要研究 Phoenix 框架与 Mybatis 框架结合，使代码更简洁、统一。

6、系统可以把 Spring 和 SpringMVC 两个框架，升级为 SpringBoot,这样可以进一步降低项目开发难度。

6.3 本章小结

本章总结了医学影像云存储系统设计与实现的主要工作内容，并针对系统的不足，进行展望，以及后期需要开展的内容。

参考文献

[1] 饶克勤.健康中国战略与分级诊疗制度建设[J].卫生经济研究,2018(01):4-6+9

[2] 赵人行,李晓龙.互联网医疗发展环境、目标及展望[J].学术交流,2018(02):127-132.

[3] 郭朝先,胡雨朦.中外云计算产业发展形势与比较[J].经济与管理,2019,(02):86-92.

[4] 王亚玲,李春阳,崔蔚等.基于 Docker 的 PaaS 平台建设[J].计算机系统应用,2016,25(03):72-77.

[5] 刘胜强,王晶.Docker 的 Hadoop 平台架构分析[J].自动化与仪器仪表,2018(10):192-195.

[6] Grace A.Lewis.Cloud Computing[J].Computer,2017,Vol.50(No.5): 8-9

[7] Xin Liu,Wenfeng Shen,Baohua Liu,et al.Research on Large Screen Visualization Based on Docker[J].Journal of Physics:Conference Series,2019,Conference 1(Vol.1169):012052

[8] N Hardi,J Blomer,G Ganis,et al.Making containers lazy with Docker and CernVM-FS[J].Journal of Physics:Conference Series,2018, Vol.1085：032019

[9] Tihfon,Park,Kim,et al.An efficient multi-task PaaS cloud infrastructure based on docker and AWS ECS for application deployment[J].2016,Vol.19(No.3):1585-1597

[10]张羿,胡永华,黄丁.基于 Docker 的电网轻量级 PaaS 平台构建方案[J].信息与电脑(理论版),2017(11):75-78.

[11]方胜吉.基于 DICOM 标准的医学影像数据格式分析及存储方法[J].中国新通信,2017,19(09):148-149.

[12]钟良志.基于 DICOM 标准的医学图像解析与处理系统的设计与实现[D].电子科技大学,2018.

[13]林伟婷.C/S 与 B/S 架构技术比较分析[J].科技资讯,2018,16(13):15-16.

[14]黄江兵,邵亚丽.基于 Spring 框架的汽车租赁系统分析与设计[J].电脑知识与技术,2018,14(19):75-76+78.

[15]张泉森.基于 Spring 框架的高校教学评估系统的设计与实现[D].湖南大学,2018

[16]侯瑞敏.基于 SSM 的电力设备管理系统的设计与实现[D].华北电力大学,2017.

[17]黄素萍,欧阳宏基.基于 Spring MVC 框架的 Java Web 应用[J].计算机与现代化,2018(08):97-101.

[18]管才路,叶刚,耿伟,王立河.基于 Java 的 Mybaits 生成持久层配置文件[J].电子技术与软件工程,2018(22):139.

[19]张哲,王荣,温伟鸽.基于 SSM 框架整合的工服报号统计系统设计[J].电脑知识与技术,2018,14(15):113-115.

[20]刘文东.基于 Phoenix 平台的空间数据索引与查询技术研究[D].西安电子科技大学,2018.

[21]陈大才.基于 Nginx 的高并发访问服务器的研究与应用[D].中国科学院大学(中国科学院沈阳计算技术研究所),2018.

[22]孔艳蓉.基于 Nginx 高并发 Web 服务器负载均衡策略的研究与改进[D].长安大学,2018.

[23]邢硕.基于 SSH 框架的学术在线交流平台设计与实现[D].吉林大学,2018.

[24]周长俊,宗平.Hadoop 备份数据存放策略的改进[J].计算机技术与发展,2019,29(01):11-16.

[25]谭台哲,向云鹏.Hadoop 平台下海量图像处理实现[J].计算机工程与设计,2017,38(04):976-980.

[26]冯心欣,胡淑英,邹其昊,徐艺文.基于 HBase 的车联网海量数据查询[J].福州大学学报(自然科学版),2018,46(04):466-471.

[27]黄炜耀.HBase 二级索引查询引擎实现探讨[J].科技风,2018(32):83.

[28]郭一鸣.支持分布式定时任务调度的 Web 服务的设计与实现[D].北京邮电大学,2018.

[29]邹猛.基于消息中间件的消息系统在云护理平台的应用与实现[D].北京邮电大学,2017.

[30]胡文龙.HDFS 高可用性方案的优化与实现[D].南京邮电大学,2018.

个人简历

，男，1992 年 1 月 3 日生

2014 年 7 月毕业于郑州大学信息工程学院通信工程，获得通信工程专业学士学位。

2016 年 9 月进入郑州大学产业技术研究院，攻读电子与通信工程专业学位研究生。

致谢